MODULE-2 : REQUIREMENT ENGINEERING AND MODELLING

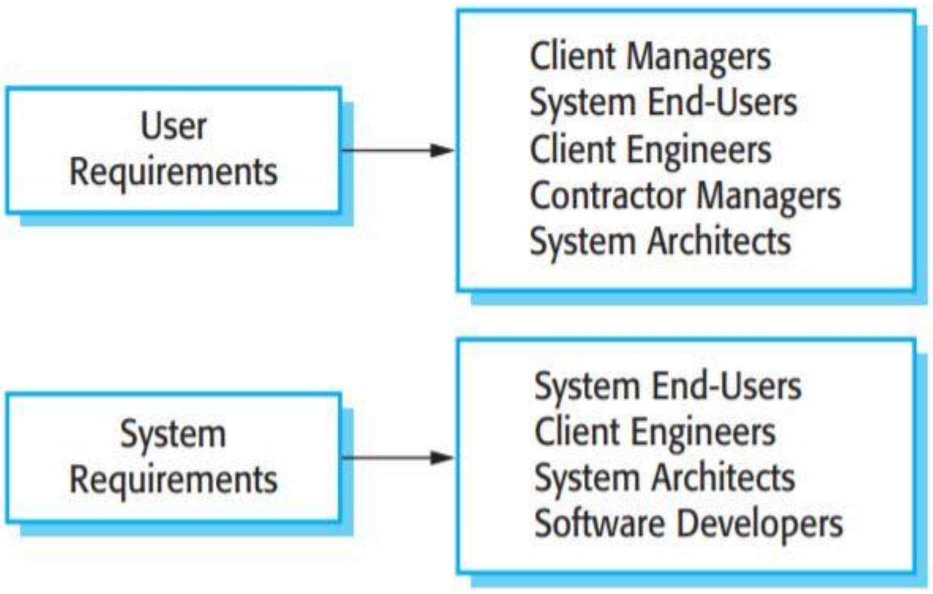
Requirements Engineering: Functional and non-functional requirements, The software requirements document, Requirements specifications, Requirements engineering processes, Requirements elicitation and analysis, Requirements validation, Requirements management.
Requirements Modeling: Requirements Analysis, Data Modeling Concepts, Flow-Oriented Modeling,Scenario based Modeling, UML Models that supplement the Use Case Case study on Requirements modeling for Web and MobileApps.

REQUIREMENTS ENGINEERING

What is Requirement

Requirements engineering is the systematic procedure of determining the specific services that a client demands from a system, as well as the limitations and conditions that govern its operation and development. Requirements are the explicit descriptions of the services and limitations of a system that are developed during the process of requirements engineering.

Different types of Requirement Specification



Functional and non-functional requirements

- **Functional requirements**

Specifications describing the functions the system is expected to carry out, as well as the expected responses to and actions in certain scenarios.

Non-functional requirements

- limitations imposed by the system on the services or capabilities it provides, such as strict deadlines, difficult development processes, stringent quality controls, etc.

Domain requirements

Domain-specific requirements are system requirements that are derived from the application domain and reflect the unique characteristics and functionalities of that domain.

Functional requirements

• Specify the features and services of the system.

• Be contingent upon the nature of the software, its intended audience, and the nature of the system on which it runs.

• The functional system requirements must be specific, in contrast to the functional user requirements which might be more general descriptions of the system's expected behaviour.

Requirements completeness and consistency

It is generally accepted that requirements should be exhaustive and uniform.

Complete

– All services required by the user should be defined.

Consistent

It is impossible to create a comprehensive and consistent requirements document in practise, but it is essential that there be no inconsistencies or conflicts in the descriptions of system facilities.

Non-functional requirements

- The requirements and characteristics of a system are defined by these factors. I/O device limitations, system representations, and similar factors all serve as constraints.

- Non-functional needs can be more crucial than functional requirements. If these conditions are not fulfilled, the system becomes ineffective.
- The implementation of these needs may be spread out across the system. There are two factors contributing to this:

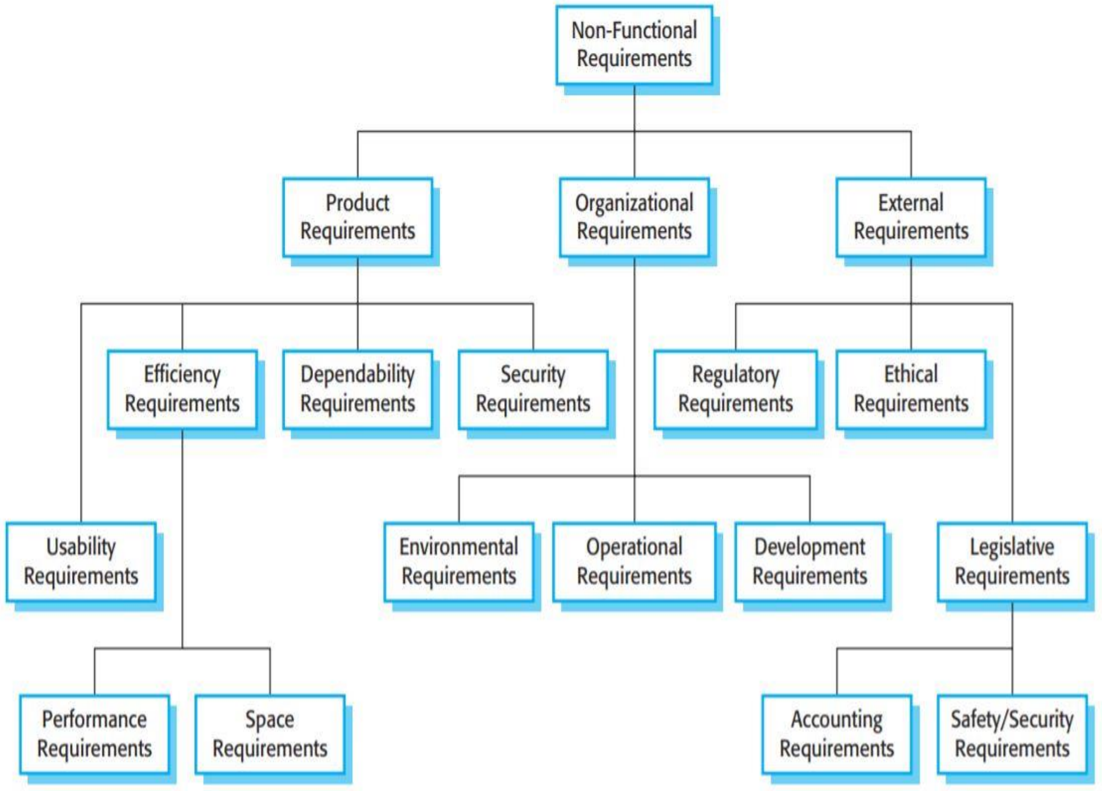
• The non-functional requirements of a system may have more of an impact on the system's architecture as a whole than on its individual components. To fulfil performance criteria, for instance, you might have to organise the system in such a way as to reduce the amount of communication that occurs between its various components.

• It is possible for a single non-functional demand, such as a requirement for system security, to spawn a number of related functional requirements that describe new system services that must be provided. Additionally, it is possible for it to produce requirements that constrain requirements that are already in place.

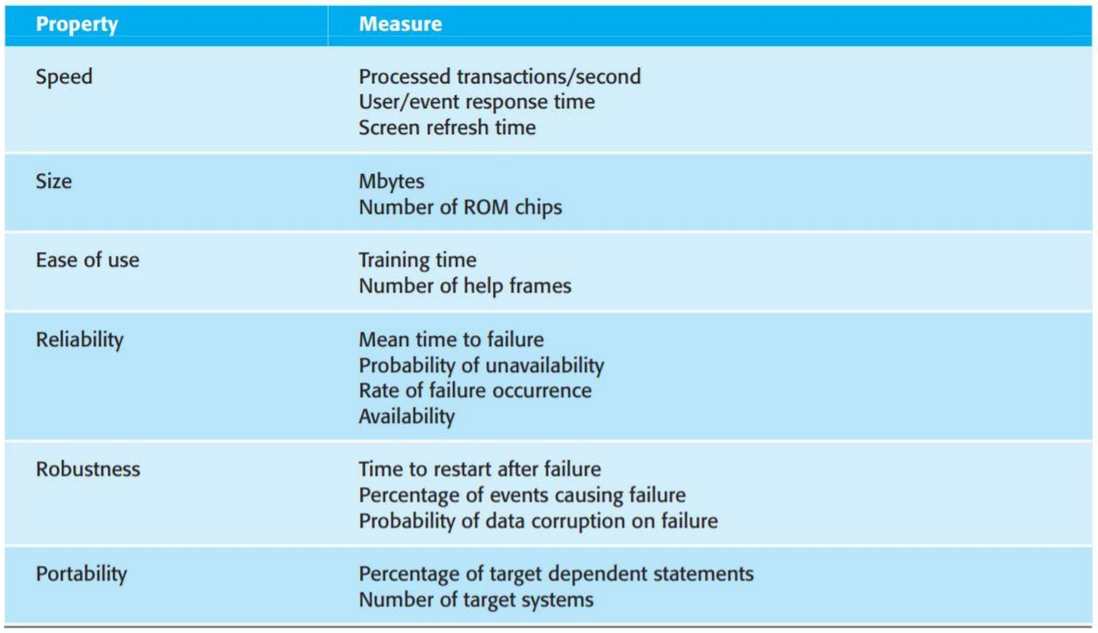
Non-functional classifications

- **Product requirements**
- Prescriptions on how the delivered product must perform, including time to completion, reliability, etc.
- **Organisational requirements**
- Company-specific requirements, such as those for meeting process standards, meeting deadlines, etc., that are a direct outcome of the company's policies and procedures.
- **External requirements**

- Requirements that are imposed on the system and its development process by elements that are not directly related to the system itself, such as interoperability requirements, legislative requirements, and so on.



Metrics for specifying Non functional Requirements



Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

The Software Requirements Document

A formal statement of what should be implemented by the system developers, the software requirements document (also known as the software requirements specification or SRS).

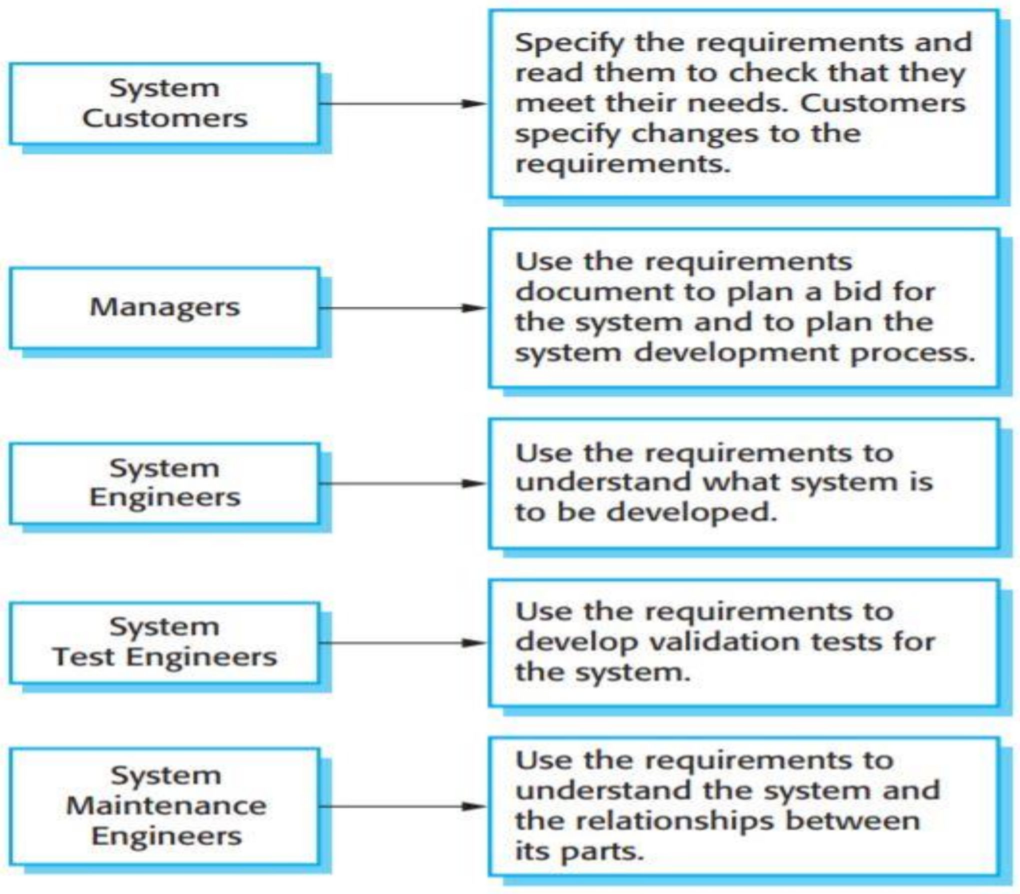
User needs and a thorough description of the system's specifications should both be included.

• The precise system requirements may be presented in a different document if there are several of them.

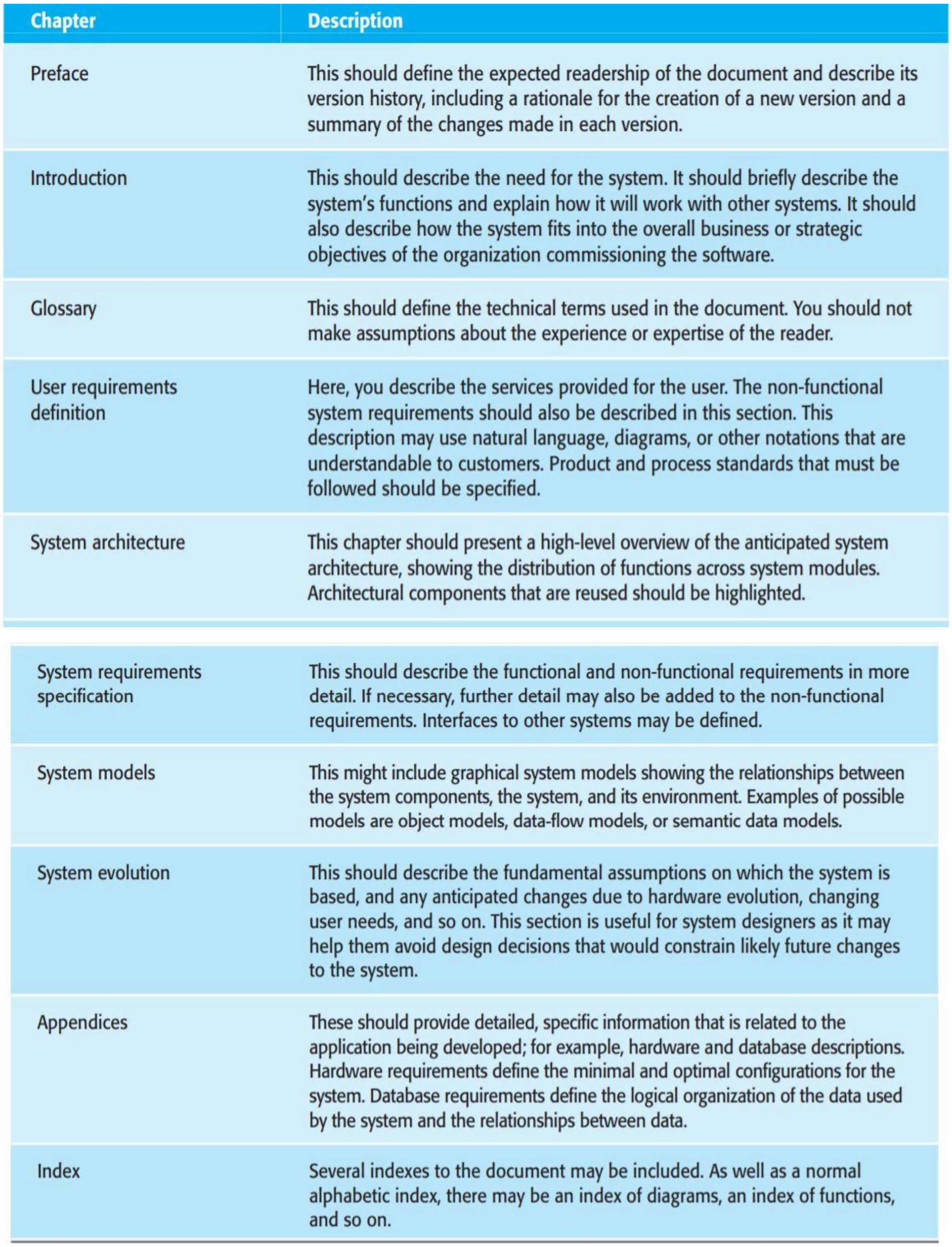
If you're having a third-party firm create your software, "requirements documents" are a must.

• In Agile development where the requirements change fast, SRS gets out of date. Therefore, Extreme Programming-style models gradually gather user needs.

Users of Requirements Document


System Customers
Specify the requirements and read them to check that they meet their needs. Customers specify changes to the requirements.
Managers
Use the requirements document to plan a bid for the system and to plan the system development process.
System Engineers
Use the requirements to understand what system is to be developed.
System Test Engineers
Use the requirements to develop validation tests for the system.
System Maintenance Engineers
Use the requirements to understand the system and the relationships between its parts.

The structure of a requirements document



Chapter	Description
Preface	This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.
User requirements definition	Here, you describe the services provided for the user. The non-functional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.
System requirements specification	This should describe the functional and non-functional requirements in more detail. If necessary, further detail may also be added to the non-functional requirements. Interfaces to other systems may be defined.
System models	This might include graphical system models showing the relationships between the system components, the system, and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.
Appendices	These should provide detailed, specific information that is related to the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used by the system and the relationships between data.
Index	Several indexes to the document may be included. As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, and so on.

Requirements specification

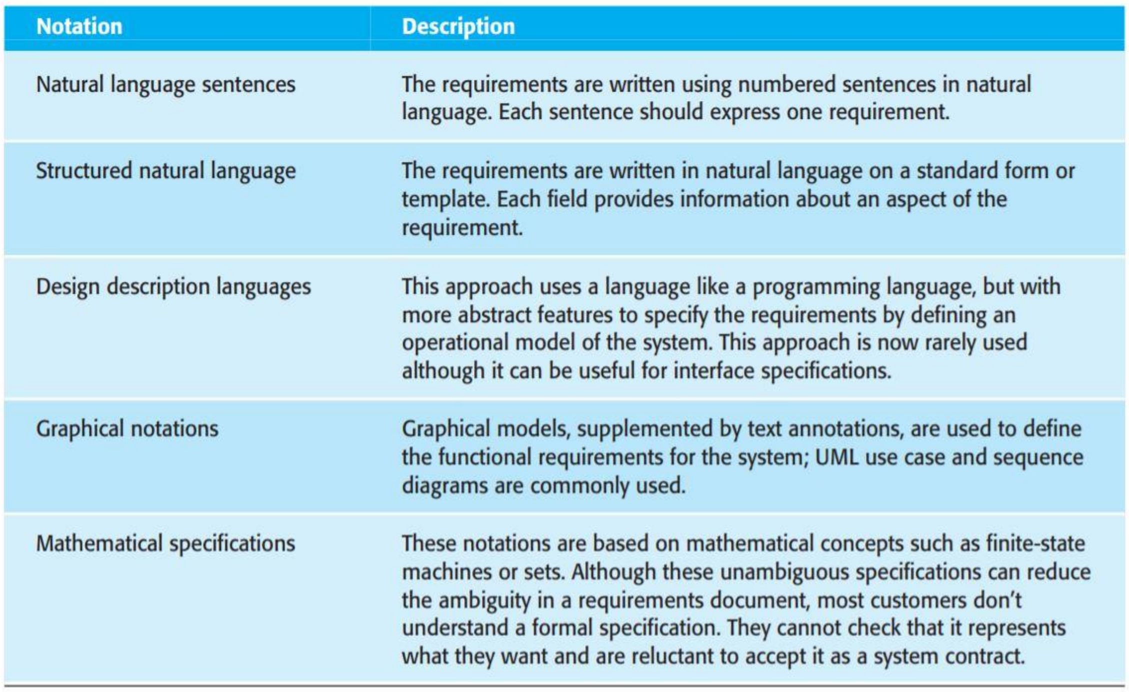
- An ideal requirements definition would result in user and system needs that are crystal obvious, unambiguous, simple to grasp, exhaustive, consistent, and well-organized into a requirements document.

- System users without extensive technical knowledge should be able to grasp the user requirements for a system if they accurately represent the system's functional and nonfunctional requirements.
- The system's external behaviour and operational restrictions should be simply described in the requirements.
- They shouldn't worry about the system's design or implementation.

All design information must be included, and that's not doable. This is due to a number of factors:

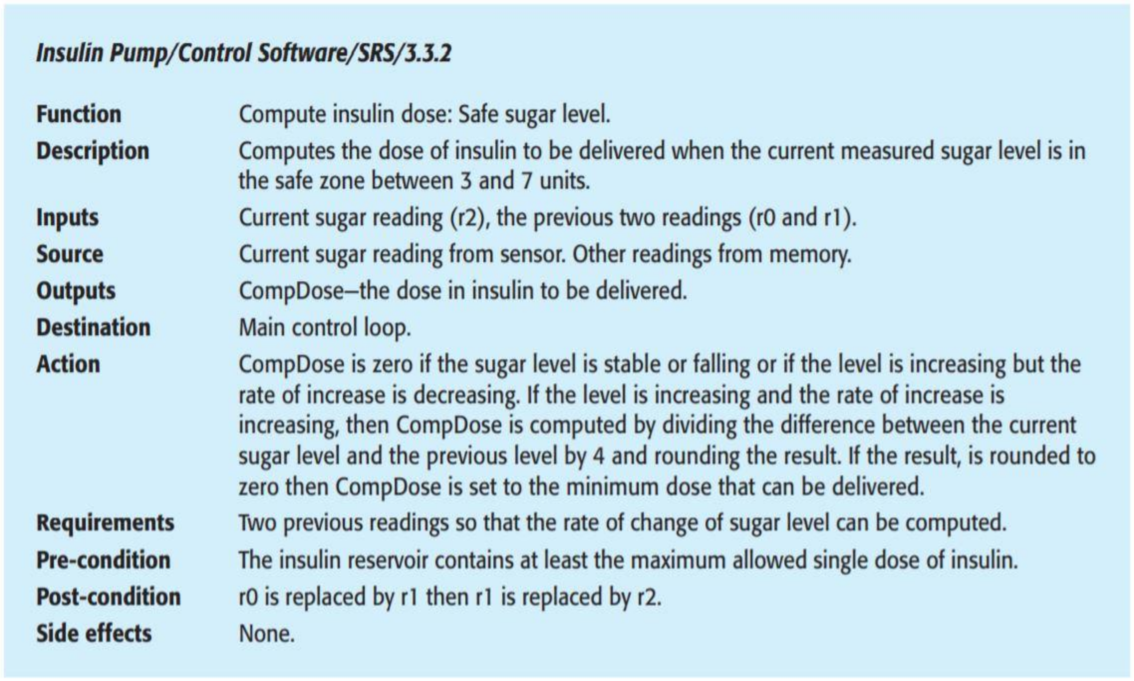
1. To better organise the requirements specification, you may need to create an initial architecture of the system. Requirements are categorised by the several subsystems that make up the whole.
2. Most systems need to communicate with one another, which might place limitations on the design and additional demands on the new system.
3. The usage of a specific architecture to satisfy non-functional criteria may be necessary. If an outside authority has to verify the system's security, they can insist on using a layout that has already been approved.

Ways of writing a system requirements specification



Notation	Description
Natural language sentences	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
Design description languages	This approach uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model of the system. This approach is now rarely used although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want and are reluctant to accept it as a system contract.

A structured specification of a requirement for an insulin pump



Insulin Pump/Control Software/SRS/3.3.2

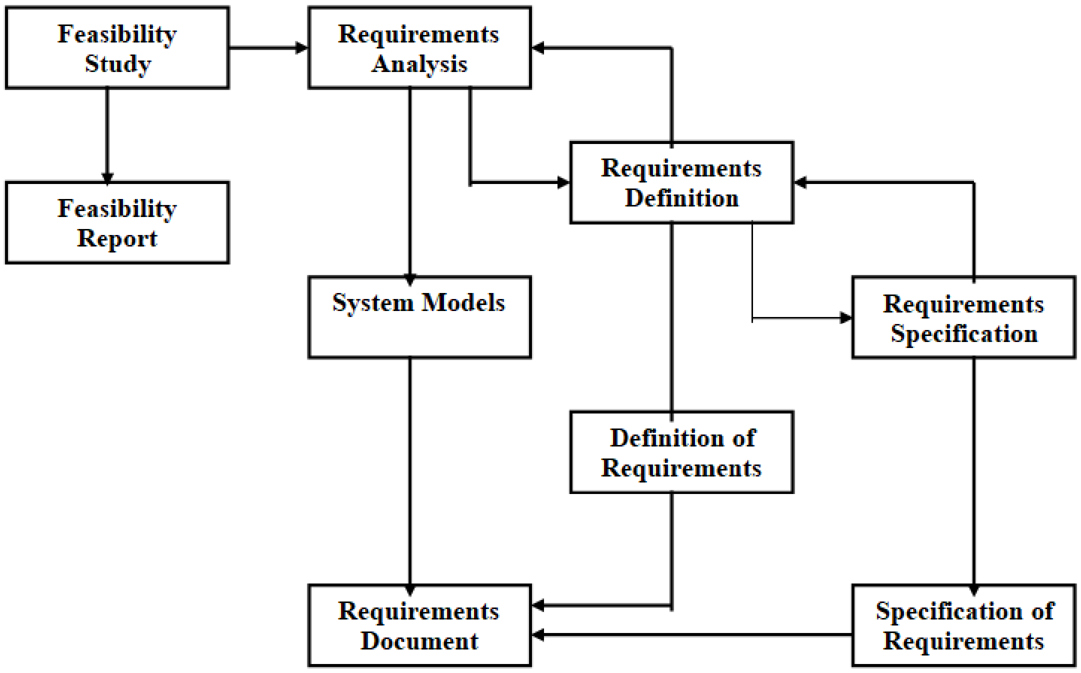
Function	Compute insulin dose: Safe sugar level.
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1).
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose–the dose in insulin to be delivered.
Destination	Main control loop.
Action	CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.
Requirements	Two previous readings so that the rate of change of sugar level can be computed.
Pre-condition	The insulin reservoir contains at least the maximum allowed single dose of insulin.
Post-condition	r0 is replaced by r1 then r1 is replaced by r2.
Side effects	None.

Requirements Engineering Processes

Definition:

* The requirements definition and requirements specification are the end results of the requirements engineering process.

The software specification and reports on the system's viability are also included.



There are four principal stages in this process:

Feasibility Study:

* It is estimated whether the identified user need can be met with current software and hardware technology.

The study will also decide if the suggested system is cost-effective and if it can be built within the current budget.

The result should help decide if a more in-depth study should be done or not.

A viability study shouldn't cost too much and shouldn't take too long.

Requirements Analysis:

obtaining the system requirements by performing activities such as analysing tasks, talking to possible users, evaluating existing systems, etc. The analyst gains a better understanding of the system that needs to be detailed as a result of this. In order to acquire a clearer picture of the requirements, system prototypes are often developed.

Requirements Definition:

The activity of transforming the information acquired during the analysis activity into a document that describes a set of requirements is known as the requirements definition activity.

* This document needs to be created in such a way that it can be comprehended by the end-user as well as the system customer.

Requirements Specification:

The development of this document is conducted concurrently with the high-level design process. Deficiencies in requirement specification are identified during the process of document development, necessitating modifications to rectify these issues.

* A comprehensive and accurate specification of the system requirements is established, serving as the foundation for the contractual agreement between the client and the software developer.

* The system requirements statement serves as the foundation for the contractual agreement between the client and the software developer.

Requirement Elicitation:

* It is a procedure that involves asking review questions with the client, the user, and other people to inquire about

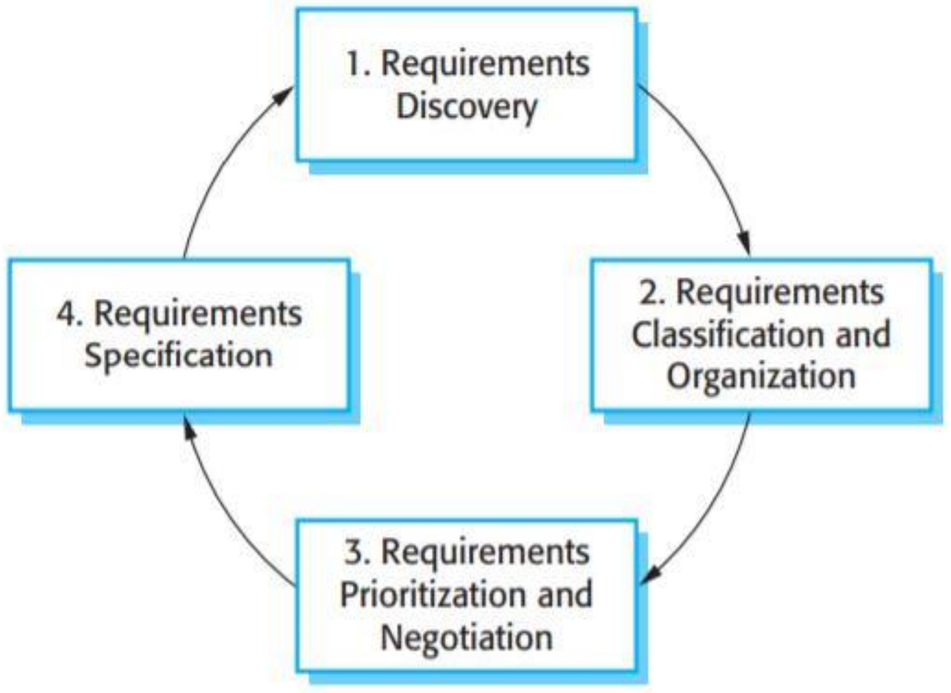
=> What the goal of the system is

.=> What exactly is it that has to be done?

=> How does the system accommodate the requirements of the organization?

=> What is the recommended way to use the product or the system on a day-to-day basis

The requirements elicitation and analysis process


1. Requirements Discovery
2. Requirements Classification and Organization
3. Requirements Prioritization and Negotiation
4. Requirements Specification

The process activities are:

1. **Requirements discovery** This is the process of discovering the requirements of the stakeholders of the system through interaction with those stakeholders. During the course of this process, domain needs from the stakeholders and the documentation are also found.

2. **Requirements classification and organization** This action takes the unstructured collection of requirements and organises them into coherent clusters by grouping together requirements that are connected to one another. Using a model of the system architecture to identify sub-systems and to associate requirements with each sub-system is the most popular method for grouping requirements. This is also the most effective method. In

actual practise, the processes of requirements engineering and architectural design simply cannot be kept entirely apart from one another.

3. **Requirements prioritization and negotiation** When there are several stakeholders involved, it is inevitable that their requirements may conflict with one another. Within the scope of this activity are the processes of prioritising requirements, discovering conflicts between requirements, and resolving those conflicts through negotiation. In most cases, the parties involved are required to get together in order to reconcile their differences and reach an agreement on the necessary concessions.

4. **Requirements specification** The specifications are written down and included into the subsequent iteration of the spiral. There is potential for the production of formal or informal requirements documents.

Viewpoints

- Viewpoints serve as a method for organizing requirements in order to accurately represent the many perspectives of different stakeholders. Stakeholders can be categorized based on several perspectives.
- A multi-perspective study is crucial because there is no definitive method for analyzing system requirements.

Types of viewpoint

- Interactor viewpoints
    - Individuals or other entities that directly engage with the system. In an ATM, the customer's information and the account database are interconnected virtual private networks.
- Indirect viewpoints
    - Individuals who are not directly involved in the system's operation but who have an impact on its requirements. Both the management and the security staff at an ATM are considered to be indirect opinions.
- Domain viewpoints
    - The needs are influenced by the characteristics and constraints of the domain. In the context of an Automated Teller Machine (ATM), an illustration would be the protocols and guidelines governing the exchange of information between different banks.

Interviewing

• During either a formal or a casual interview, the RE team will ask stakeholders questions about the system that they currently use as well as the system that will be constructed.
• There are two different kinds of interviews: closed interviews, in which participants answer a series of questions that have been determined in advance, and open interviews.

— Interviews with no set agenda, in which a wide variety of topics are discussed with various stakeholders; these are open interviews.

Scenarios
• Scenarios are real-life examples of how a system might be utilised. • Scenarios should include the following: a description of the beginning situation; a description of the goal of the scenario.

- A description of the typical order in which things occur;
- A discussion of the various things that could go wrong;
- Information regarding other activities taking place at the same time;
- A description of the situation that exists once the scenario has been completed.

Requirements checking

- <text>Validity.</text> <text>Does the system offer the functions that most effectively meet the customer's requirements?</text>
- Consistency. Are there any conflicts arising from requirements?
- Completeness. Does the customer's requirements encompass all necessary functions?
- Realism. Is it feasible to achieve the requirements within the constraints of the existing budget and technology?
- Verifiability. Could you verify the requirements?

Requirements validation techniques

- Requirements reviews
  - A methodical manual analysis of the specifications.
- Prototyping

  - Verifying requirements with an executable model of the system.
- Test-case generation
  - Creating tests to verify the testability of requirements.

Requirements reviews

- While the requirements definition is being developed, regular reviews ought to be conducted.
- Staff from the contractor and the client should participate in reviews.
- Reviews might be informal or formal, requiring completed papers. Early problem solving is possible when developers, clients, and users have effective communication.

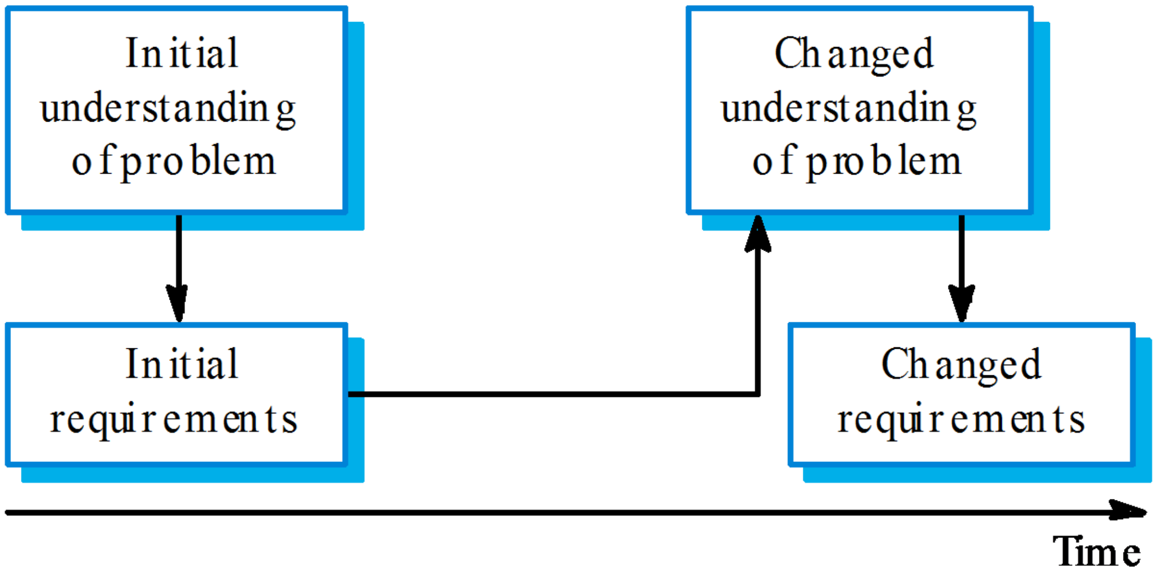
Review checks

- Verifiability. Can the requirement be tested in a practical way?
- Comprehensibility. Does the requirement make sense to you?
- Traceability. Does the requirement clearly identify where it came from?
- Adaptability. Is it possible to modify the requirement without significantly affecting other requirements?

Requirements Management

- Managing evolving needs during requirements engineering and system development is known as requirements management.
- There will always be incomplete and inconsistent requirements;
  - As business needs evolve and a deeper understanding of the system is created, new requirements will always arise;
  - Diverse perspectives will result in diverse requirements, many of which are contradictory.

Requirements evolution


Initial understanding of problem
Changed understanding of problem
Initial requirements
Changed requirements
Time

Requirements classification

Requirement Type	Description
Mutable requirements	Requirements that change because of changes to the environment in which the organisation is operating. For example, in hospital systems, the funding of patient care may change and thus require different treatment information to be collected.
Emergent requirements	Requirements that emerge as the customer's understanding of the system develops during the system development. The design process may reveal new emergent requirements.
Consequential requirements	Requirements that result from the introduction of the computer system. Introducing the computer system may change the organisations processes and open up new ways of working which generate new system requirements
Compatibility requirements	Requirements that depend on the particular systems or business processes within an organisation. As these change, the compatibility requirements on the commissioned or delivered system may also have to evolve.

Requirements Management Planning

• During the process of requirements engineering, you are required to plan for the following: – Requirements identification

• The procedure that is followed when assessing a change in the requirements; – An approach to the management of change

• The method by which individual needs are determined

- Policies regarding traceability

• The quantity of information that is retained regarding the links between the needs;

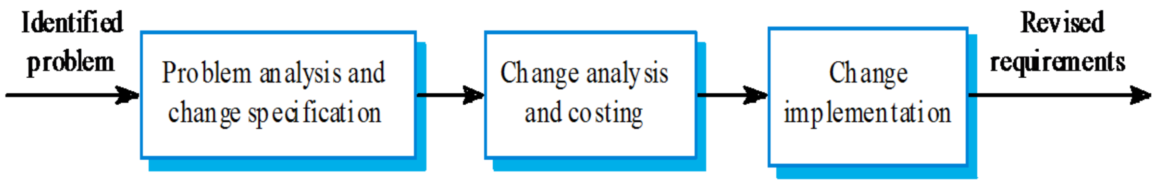
– Support for CASE-based tools

• The tool support necessary to assist with the management of changing requirements;

Requirements change management

- Must be applicable to any suggested modifications to the requirements.
- Principal stages
  - Problem analysis. Examine the requirements issue and suggest a solution;
  - Change analysis and costing. Analyze how a modification will affect other requirements;
  - Change implementation. Changes should be reflected in the requirements document and other documents.

Change management


Identified problem
Problem analysis and change specification
Change analysis and costing
Change implementation
Revised requirements

Requirements Modeling

Requirements Analysis

- Models of the following kinds are produced as a result of the requirements modelling process:
- Requirements models based on scenarios, including input from a wide range of system "actors."
- Class-oriented models capture the interplay between classes in an object-oriented framework and the properties and operations they use to fulfil system needs.
- Pattern- and behavior-based models that show how the programme responds to outside "events."
- Data models representing the problem's information space.
- Models focused on data flow that depict the system's functional components and the transformations they enact on data as it moves through the system.
- There are three main goals that the requirements model has to accomplish:

(1) describing the needs of the customer;

(2) providing a foundation for the software architecture; and

(3) defining a set of requirements that can be verified once the software is developed.

Analysis Rules of Thumb

• The model should be centred on the obvious requirements of the problem or business domain. It is advised that a relatively high level of abstraction be used.

• The information domain, the function, and the behaviour of the system are all aspects of the system that could benefit from the insights provided by the requirements model. That's the case if we want to say this model works.

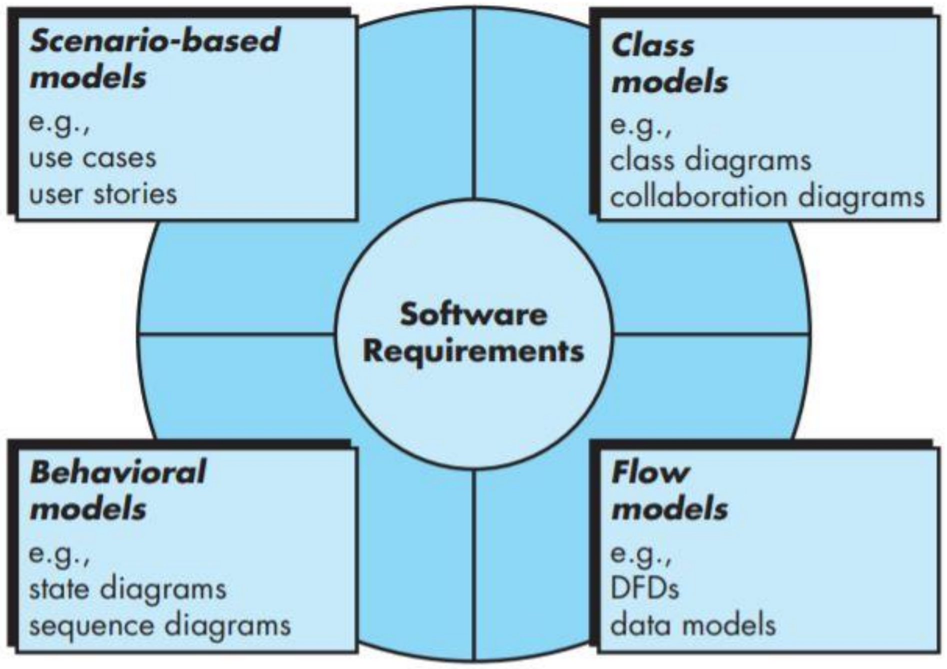
• Consideration of infrastructure and other non-functional models should be postponed until after design has been completed. That is to say, it is possible that a database will be necessary; nevertheless, the classes necessary to implement it, the functions required to access it, and the behaviour that will be exhibited as it is used should not be addressed until after the issue domain analysis has been finished.

• Reduce the amount of connection that exists throughout the system. It is essential to accurately portray the connections that exist between classes and functions. However, if the degree of "interconnectedness" is already at an excessively high level, there is a need for attempts to bring it down.

• You should make sure that the requirements model offers something of value to all of the stakeholders. The model can be put to a variety of different uses depending on the constituency. For instance, business stakeholders should use the model to validate the requirements; designers should use the model as a basis for design; and quality assurance experts should use the model to assist in the planning of acceptance tests.

• Try to keep the model as straightforward as possible. It is not necessary to provide additional diagrams if they do not contribute any new information. Avoid using complicated notational forms wherever possible and stick to using lists instead.

Elements of the analysis model

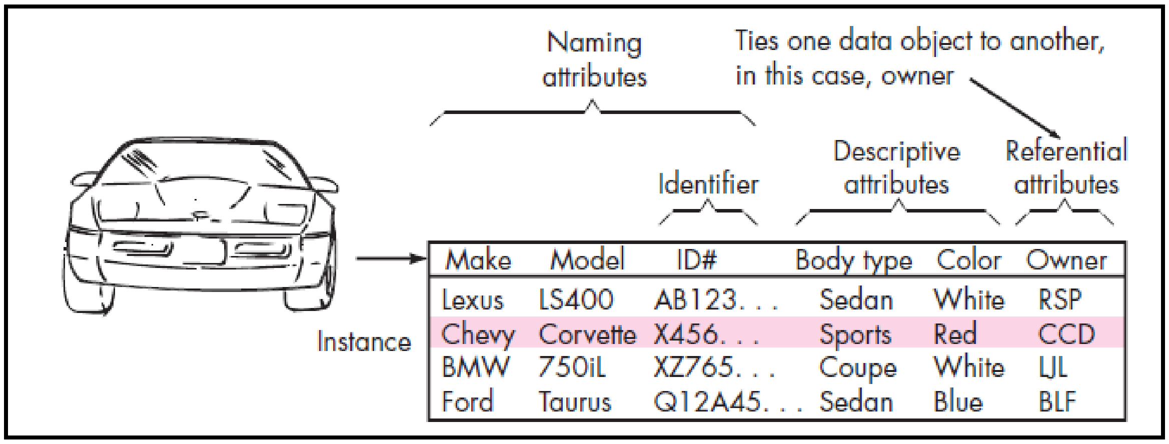


DATA MODELING CONCEPTS

- Data Objects
- Data Attributes
- Relationships

Data Objects

- A data object is a computer-understandable representation of complex data.
- • Anything that meets the definition of a data object (such as anything that generates or uses information) can be considered a data object.
- something (such a report or a display, for example)
- an occurrence (like a phone call), • an event (like an alarm), or both
- a function (such as a salesperson),
- a department within an organisation (such as accountancy), a location (such as a warehouse), or a structure (such as a file) are all examples of organisational units.



Make	Model	ID#	Body type	Color	Owner
Lexus	LS400	AB123. . .	Sedan	White	RSP
Chevy	Corvette	X456. . .	Sports	Red	CCD
BMW	750iL	XZ765. . .	Coupe	White	LJL
Ford	Taurus	Q12A45. . .	Sedan	Blue	BLF

Figure. Tabular representation of data objects

Data Attributes

The properties of a data entity are referred to as its data attributes, and these attributes can have one of three distinct qualities. They have three different functions:

(1) they can be used to name an instance of the data object;

(2) they can be used to describe the instance; and

(3) they can be used to make a reference to another instance that is located in another table.

Relationships

- There are various methods in which data objects are related to one another.

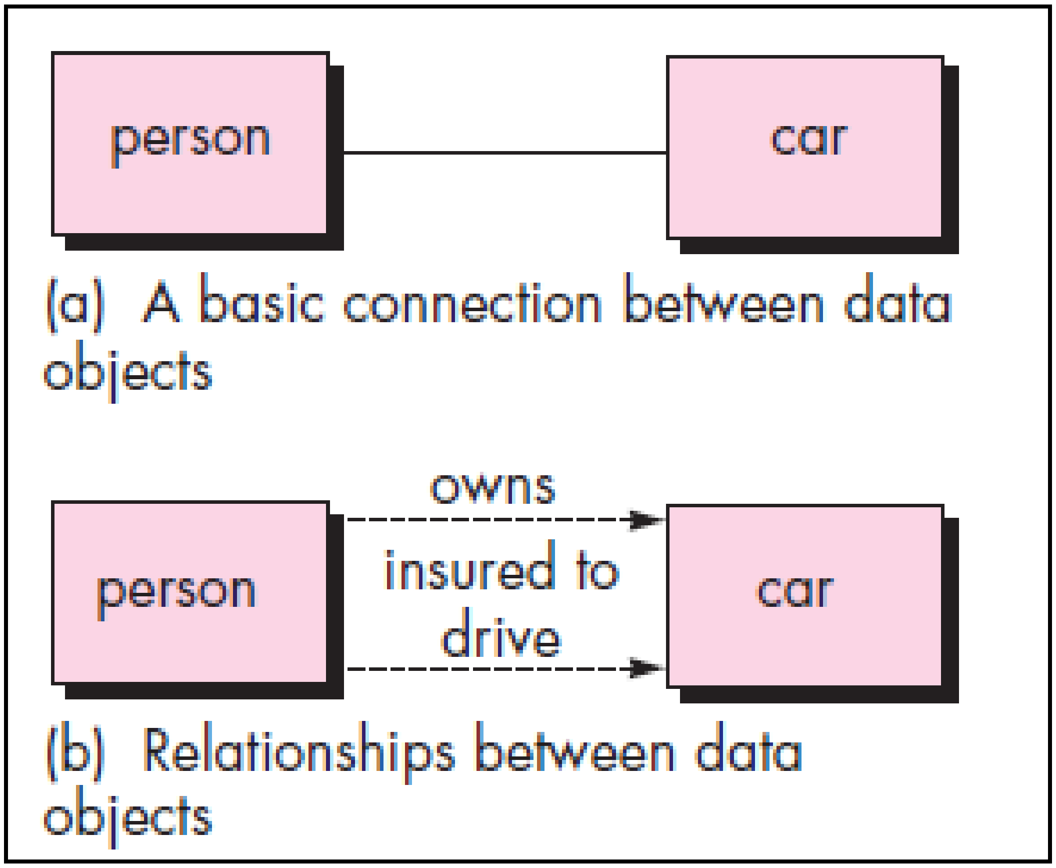

person
car
(a) A basic connection between data objects
person
owns
insured to drive
car
(b) Relationships between data objects

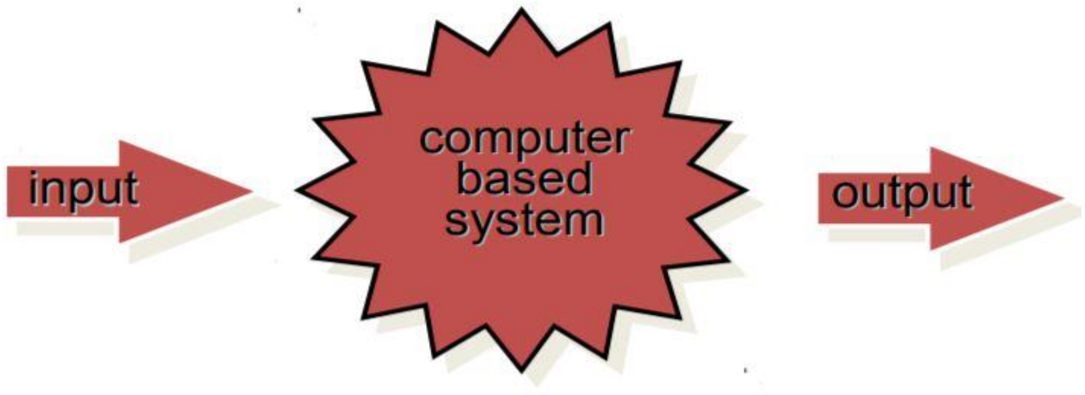
Figure. Relationships between data objects

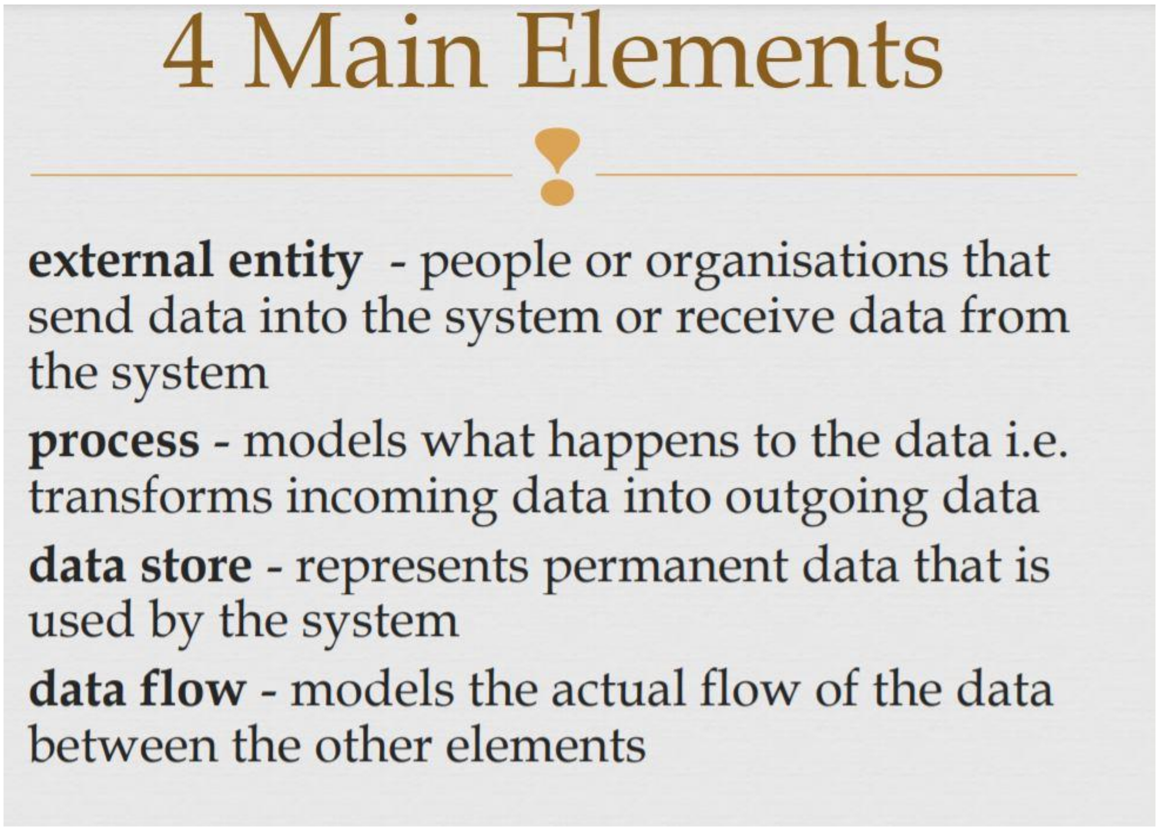
Flow Oriented Modelling

- It illustrates the transformations that occur to data objects as they pass through the system.
- DFDs explain how information enters and exits a system and what the system does with the data.
- A data flow diagram (DFD) shows how information moves through a system.

The Flow Model

Every computer-based system is an information transform ....





4 Main Elements

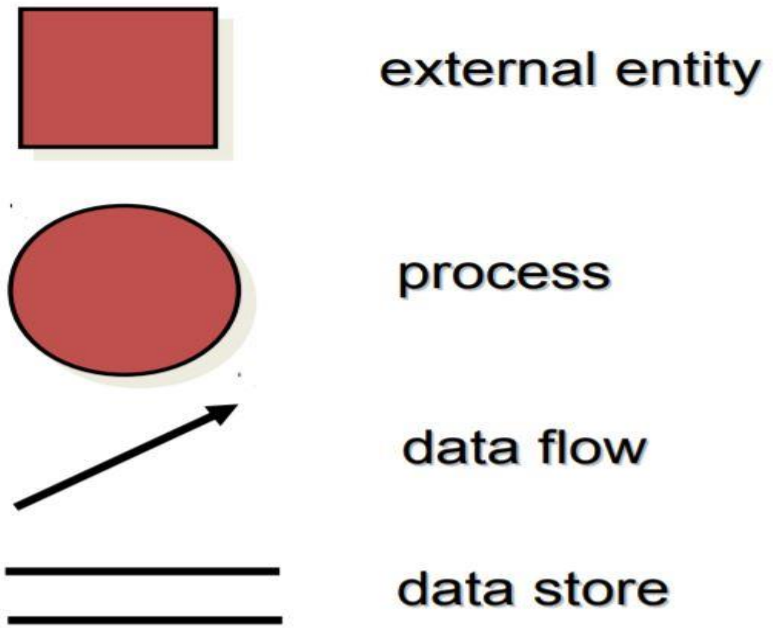
external entity - people or organisations that send data into the system or receive data from the system

process - models what happens to the data i.e. transforms incoming data into outgoing data

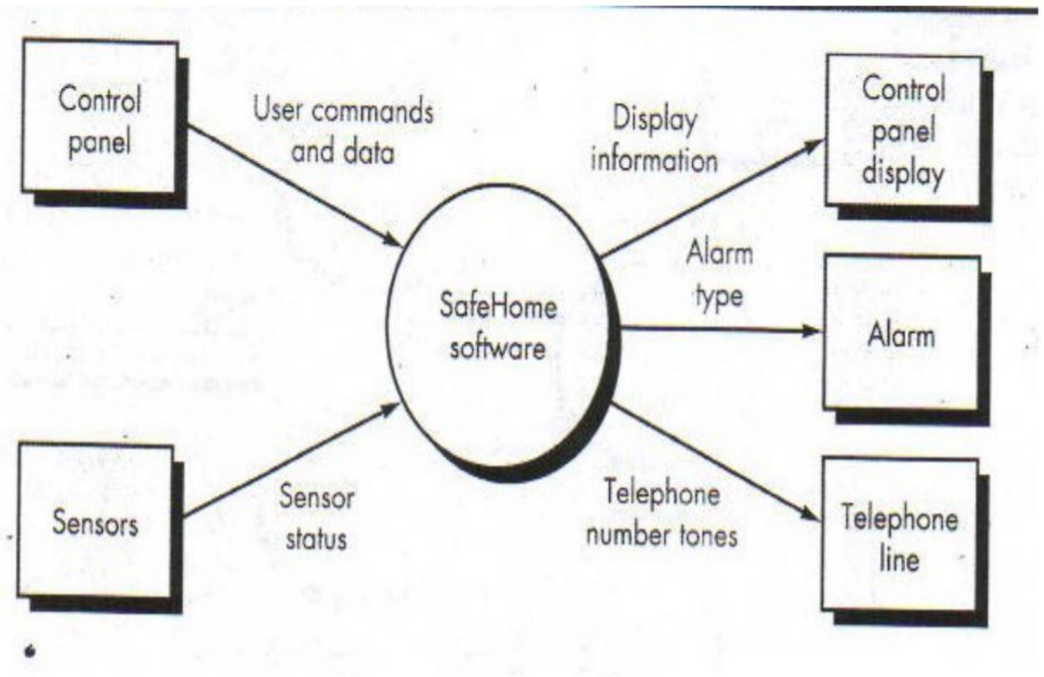
data store - represents permanent data that is used by the system

data flow - models the actual flow of the data between the other elements

Flow Modeling Notation



DFD for SafeHome system



SCENARIO-BASED MODELING

Firstly, what topic should you write on?

Secondly, What is the appropriate amount of writing to do about it?
Thirdly, what level of information is appropriate for your description?
And finally, what order should the description be in?

These are the problems that need to be addressed in order for use cases to be a useful tool for requirements modeling.

The SafeHome home surveillance function that are performed by the homeowner actor:

• Choose the camera you want to view.
• Make sure thumbnails are requested from all of the cameras.
• Views of the camera can be seen in a window on your computer.
• Manage the camera's pan and zoom settings individually.
• Record the output of the camera in a selectable manner.
• Play back the output from the camera.
• Use the Internet to access the video surveillance cameras.

Use case: Access camera surveillance via the Internet—display camera views
Actor: homeowner

1. The homeowner visits the SafeHome Products website and checks in to their account.
2. The user ID of the homeowner is entered into the system.
3. The homeowner is required to enter two passwords, each of which must be at least eight characters long.
4. The system presents buttons for all of the primary functions.
5. The homeowner presses the button labelled "surveillance" to access the system's primary functions.
6. The homeowner then chooses the option to "pick a camera."
7. The system will show you the layout of the house's floor plan.
8. The homeowner chooses an icon for a camera from the floor layout.
9. The homeowner clicks the "view" button on their computer screen.

Refining a Preliminary Use Case

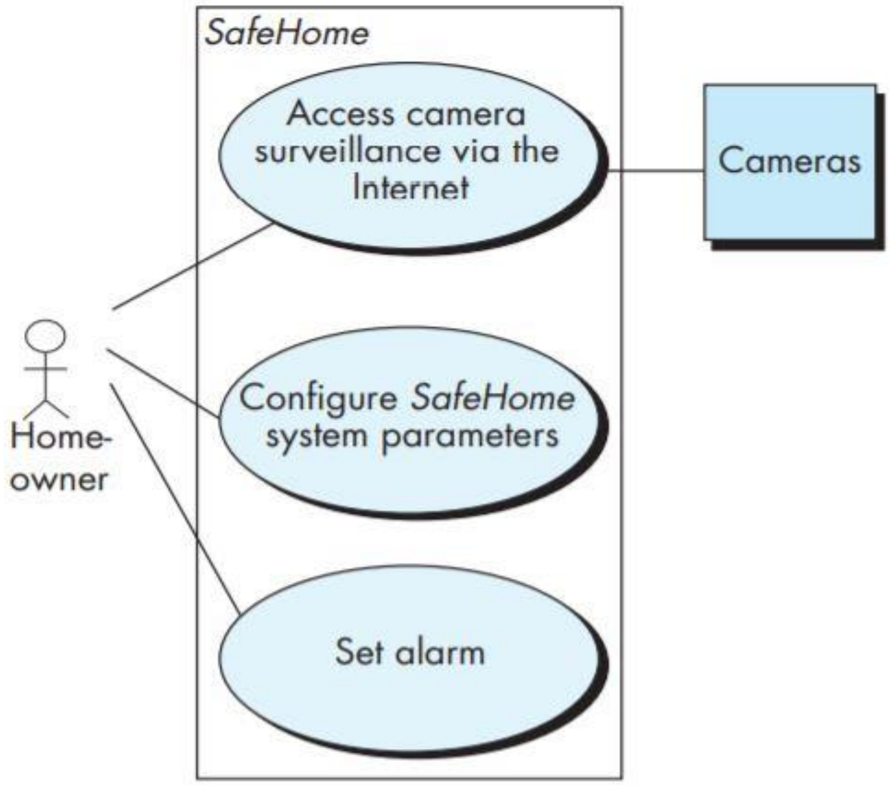
Therefore, in order to evaluate each phase that makes up the major scenario, the following questions will be asked.

• Is there any other course of action that the actor can take at this point?

• Is it feasible that the actor will experience some kind of error circumstance at this particular juncture? If that's the case, what could it be?

• Is it likely that the actor will come across another behaviour at this point, such as a behaviour that is triggered by an event that is not under the actor's control? If that's the case, what could it be?

Preliminary use case diagram for the SafeHome system

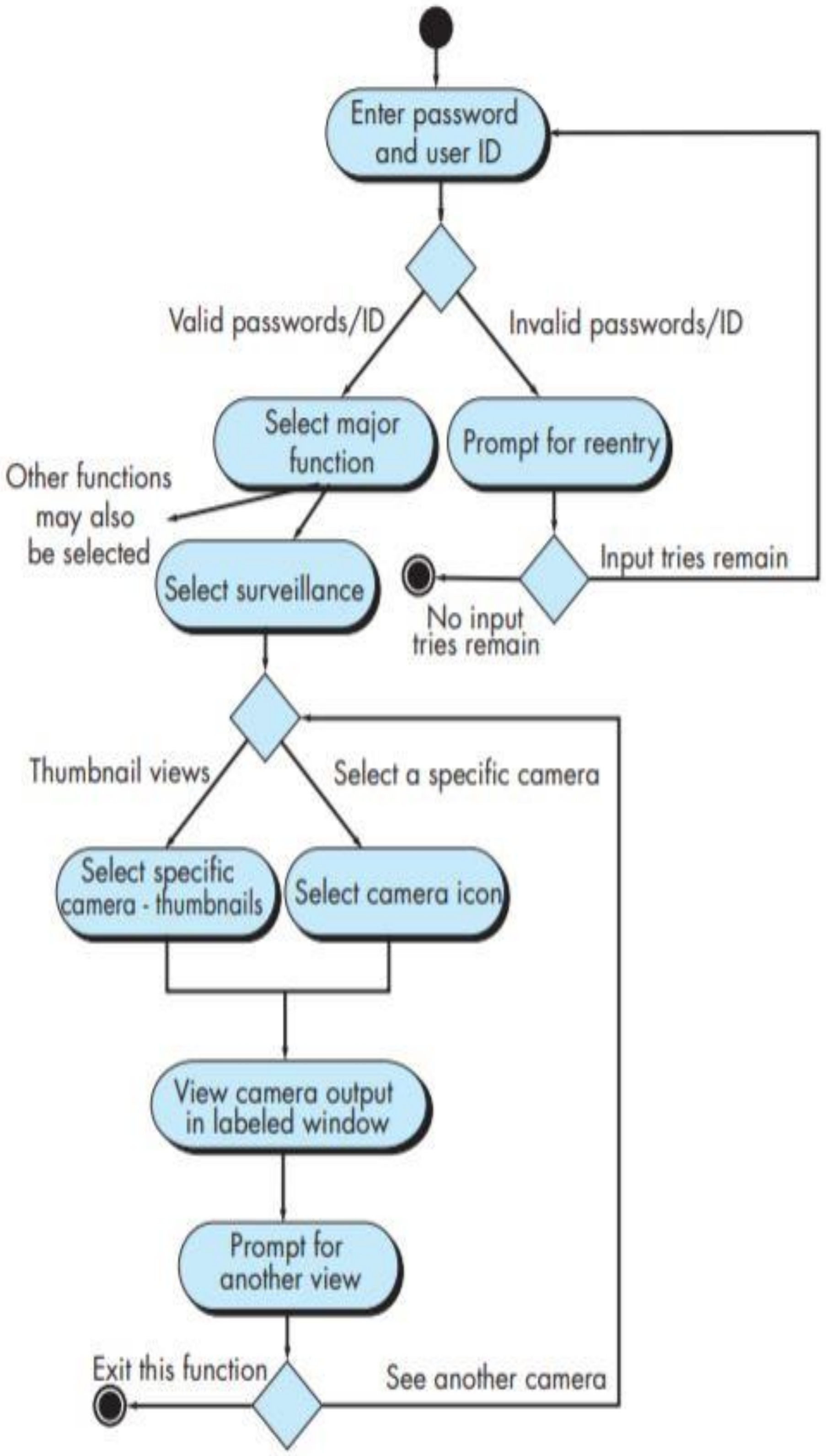

SafeHome
Access camera surveillance via the Internet
Cameras
Home-owner
Configure SafeHome system parameters
Set alarm

UML MODELS THAT SUPPLEMENT THE USE CASE

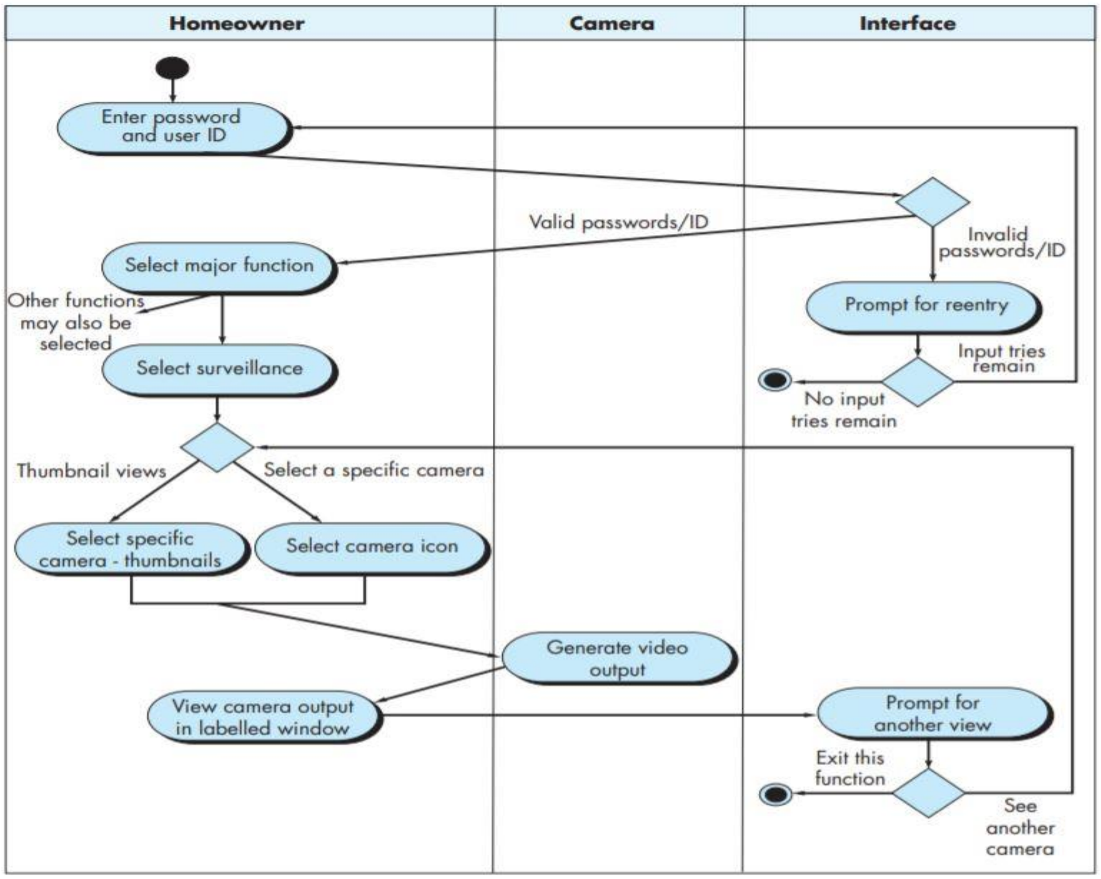
Even if it's something as straightforward as a use case, there are a lot of requirements modelling scenarios in which a text-based model might not be able to convey information in a way that's both clear and succinct. In situations like this, you have access to a vast library of UML graphical models to select from.

- Developing an Activity Diagram
- Swimlane Diagrams

Activity diagram for Access camera surveillance via the Internet— display camera views function.



Swimlane diagram for Access camera surveillance via the Internet—display camera views function.



REQUIREMENTS MODELING FOR WEB AND MOBILE APPS

The following size-related variables dictate how much focus is placed on requirements modeling for Web and mobile applications:

(1) The scope and intricacy of the application increment;

(2) The quantity of stakeholders (analysis can assist in identifying conflicting requirements originating from various sources);

(3) The size of the app development team;

(4) The extent to which team members have collaborated previously (analysis can aid in creating a shared understanding of the project); and

(5) The duration elapsed since the team's last collaboration.