Chapter 6

Software Process Model (SPM)

1. Software Project Concept

A Software project is the complete software development process that starts with gathering requirement, analyzing, designing, coding and testing in a given time frame to produce desired software output.

Characteristics of project

- Every project may have its own set of goals.
- A Project is not a day to day operation or a normal task.
- Each project has a beginning and end date.
- A project require sufficient resources in terms of time, manpower, finance, material and knowledge.

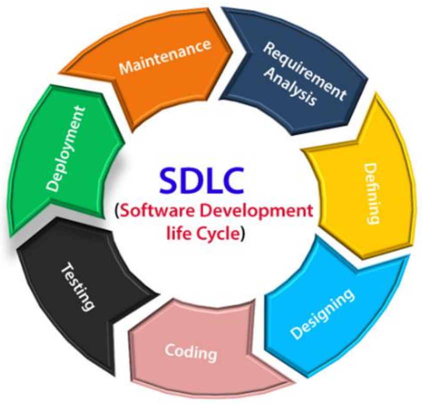
2. Concept of SDLC life cycle

A software life cycle model (also termed process model) is a pictorial and diagrammatic representation of the software life cycle. A life cycle model represents all the methods required to make a software product transit through its life cycle stages. It also captures the structure in which these methods are to be undertaken.

In other words, a life cycle model maps the various activities performed on a software product from its inception to retirement. Different life cycle models may plan the necessary development activities to phases in different ways. Thus, no element which life cycle model is followed, the essential activities are contained in all life cycle models though the action may be carried out in distinct orders in different life cycle models. During any life cycle stage, more than one activity may also be carried out.

There are different phase of software development life cycle:-

a) Requirement Analysis
b) Defining
c) Designing
d) Coding
e) Testing
f) Deployment
g) Maintenance



Stage1: Planning and requirement analysis

Requirement Analysis is the most important and necessary stage in SDLC. The senior members of the team perform it with inputs from all the stakeholders and domain experts or SMEs in the industry. Planning for the quality assurance requirements and identifications of the risks associated with the projects is also done at this stage.

Business analyst and Project organizer set up a meeting with the client to gather all the data like what the customer wants to build, who will be the end user, what is the objective of the product. Before creating a product, a core understanding or knowledge of the product is very necessary.

For Example, A client wants to have an application which concerns money transactions. In this method, the requirement has to be precise like what kind of operations will be done, how it will be done, in which currency it will be done, etc.

Stage2: Defining Requirements

Once the requirement analysis is done, the next stage is to certainly represent and document the software requirements and get them accepted from the project stakeholders. This is accomplished through "SRS"- Software Requirement Specification document which contains all the product requirements to be constructed and developed during the project life cycle.

Stage3: Designing the Software

The next phase is about to bring down all the knowledge of requirements, analysis, and design of the software project. This phase is the product of the last two, like inputs from the customer and requirement gathering.

Stage4: Developing the project

In this phase of SDLC, the actual development begins, and the programming is built. The implementation of design begins concerning writing code. Developers have to follow the coding guidelines described by their management and programming tools like compilers, interpreters, debuggers, etc. are used to develop and implement the code.

Stage5: Testing

After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage. During this stage, unit testing, integration testing, system testing, acceptance testing are done.

Stage6: Deployment

Once the software is certified, and no bugs or errors are stated, then it is deployed. Then based on the assessment, the software may be released as it is or with suggested enhancement in the object segment. After the software is deployed, then its maintenance begins.

Stage7: Maintenance

Once when the client starts using the developed systems, then the real issues come up and requirements to be solved from time to time. This procedure where the care is taken for the developed product is known as maintenance.

3. Feasibility Study:-

Feasibility Study in Software development life cycle is a study to evaluate feasibility of proposed project or system. Feasibility study is one of stage among important four stages of Software Project Management Process. As name suggests feasibility study is the feasibility analysis or it is a measure of the software product in terms of how much beneficial product development will be for the organization in a practical point of view. Feasibility study is carried out based on many purposes to analyze whether software product will be right in terms of development, implantation, contribution of project to the organization etc.

Types of Feasibility Study:

The feasibility study mainly concentrates on below five mentioned areas. Among these Economic Feasibility Study is most important part of the feasibility analysis and Legal Feasibility Study is less considered feasibility analysis.

Technical Feasibility – In Technical Feasibility current resources both hardware software along with required technology are analyzed/assessed to develop project. This technical feasibility study gives report whether there exists correct required resources and technologies which will be used for project development. Along with this, feasibility study also analyzes technical skills and capabilities of technical team, existing technology can be used or not, maintenance and up-gradation is easy or not for chosen technology etc.

Operational Feasibility – In Operational Feasibility degree of providing service to requirements is analyzed along with how much easy product will be to operate and maintenance after deployment. Along with this other operational scopes are determining usability of product, Determining suggested solution by software development team is acceptable or not etc.

Economic Feasibility – In Economic Feasibility study cost and benefit of the project is analyzed. Means under this feasibility study a detail analysis is carried out what will be cost of the project for development which includes all required cost for final development like hardware and software resource required, design and development cost and operational cost and so on. After that it is analyzed whether project will be beneficial in terms of finance for organization or not.

Legal Feasibility – In Legal Feasibility study project is analyzed in legality point of view. This includes analyzing barriers of legal implementation of project, data protection acts or social media laws, project certificate, license, copyright etc. Overall it can be said that Legal Feasibility Study is study to know if proposed project conform legal and ethical requirements.

Schedule Feasibility – In Schedule Feasibility Study mainly timelines/deadlines is analyzed for proposed project which includes how many times teams will take to complete final project which has a great impact on the organization as purpose of project may fail if it can’t be completed on time.

- **Economic:** Can we complete the project within the budget or not?
- **Legal:** Can we handle this project as cyber law and other regulatory framework/compliances?
- **Operation feasibility:** Can we create operations which is expected by the client?
- **Technical:** Need to check whether the current computer system can support the software
- Schedule: Decide that the project can be completed within the given schedule or not.

4. System Analyst Vs. Software Engineer

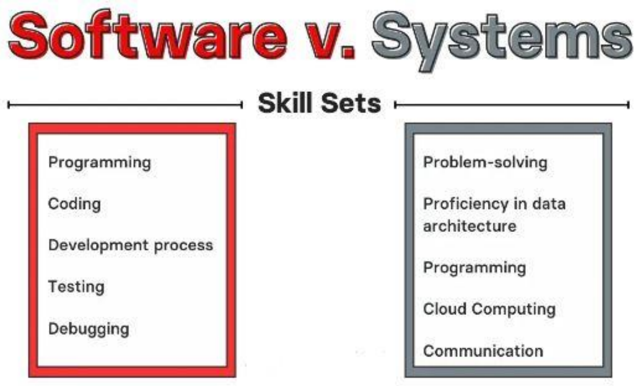
A systems analyst is an IT professional who works on a high level in an organization to ensure that systems, infrastructures and computer systems are functioning as effectively and efficiently as possible. System analysts carry the responsibilities of researching problems, finding solutions, recommending courses of actions and coordinating with stakeholders in order to meet specified requirements. They study the current system, procedures and business processes of a company and create action plans based on the requirements set.

The job's responsibilities may be summarized as follows:

- Communicate with customers and stakeholders to learn and document requirements in order to create a technical specification
- Interact and coordinate with developers and implementers
- Help perform system testing
- Deploy the system
- Help with technical documentation like manuals

Software Engineer

Software developers are the creative force behind computer programs of all kinds. They design and write the code used to build everything from operating systems to apps to video games. A software engineer designs, develops, tests, and maintains software applications and systems. They use their expertise in programming languages, software development methodologies, and tools to build and deliver software products that meet the needs of businesses, organizations, or end-users. Software engineers typically work in teams and collaborate with other professionals, such as project managers, quality assurance testers, and designers, to ensure that software products are of high quality, reliable, and user-friendly. They may also work on a variety of software systems, including web applications, mobile applications, desktop applications, and operating systems.



5. Requirement collection methods

Requirements gathering is the process of understanding what you are trying to build and why you are building it. Requirements gathering is often regarded as a part of developing software applications. It is the process of generating a list of functional, technical and systematic requirements from several project collaborators, such as clients, information technology staff, product users or vendors. This list often includes features, activities and tasks for a team to finish to achieve the goals of a project.

The following are some methods for requirement collections:-

Brainstorming

Brainstorming is used in requirement gathering to get as many ideas as possible from group of people. Generally used to identify possible solutions to problems, and clarify details of opportunities.

Document Analysis

Reviewing the documentation of an existing system can help when creating AS–IS process document, as well as driving gap analysis for scoping of migration projects. In an ideal world, we would even be reviewing the requirements that drove creation of the existing system – a starting point for documenting current requirements. Nuggets of information are often buried in existing documents that help us ask questions as part of validating requirement completeness.

Focus Group

A focus group is a gathering of people who are representative of the users or customers of a product to get feedback. The feedback can be gathered about needs/opportunities/ problems to identify requirements, or can be gathered to validate and refine already elicited requirements. This form of market research is distinct from brainstorming in that it is a managed process with specific participants.

Interface analysis

Interfaces for a software product can be human or machine. Integration with external systems and devices is just another interface. User centric design approaches are very effective at making sure that we create usable software. Interface analysis – reviewing the touch points with other external systems is important to make sure we don’t overlook requirements that aren’t immediately visible to users.

Interview

Interviews of stakeholders and users are critical to creating the great software. Without understanding the goals and expectations of the users and stakeholders, we are very unlikely to satisfy them. We also have to recognize the perspective of each interviewee, so that, we can properly weigh and address their inputs. Listening is the skill that helps a great analyst to get more value from an interview than an average analyst.

Observation

By observing users, an analyst can identify a process flow, steps, pain points and opportunities for improvement. Observations can be passive or active (asking questions while observing). Passive observation is better for getting feedback on a prototype (to refine requirements), where active observation is more effective at getting an understanding of an existing business process. Either approach can be used.

Prototyping

Prototyping is a relatively modern technique for gathering requirements. In this approach, you gather preliminary requirements that you use to build an initial version of the solution - a prototype. You show this to the client, who then gives you additional requirements. You change the application and cycle around with the client again. This repetitive process continues until the product meets the critical mass of business needs or for an agreed number of iterations.

Requirement Workshops

Workshops can be very effective for gathering requirements. More structured than a brainstorming session, involved parties collaborate to document requirements. One way to capture the collaboration is with creation of domain-model artifacts (like static diagrams, activity diagrams). A workshop will be more effective with two analysts than with one.

Reverse Engineering

When a migration project does not have access to sufficient documentation of the existing system, reverse engineering will identify what the system does. It will not identify what the system should do, and will not identify when the system does the wrong thing.

Survey/Questionnaire

When collecting information from many people – too many to interview with budget and time constraints – a survey or questionnaire can be used. The survey can force users to select from choices, rate something ("Agree strongly, agree…"), or have open ended questions allowing free-form responses. Survey design is hard – questions can bias the respondents.

6. Concept of System Design

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well-running system. The tools which are used to design the system are known as system design tools. Following are the tools which are used to design the system.

- **Algorithm**

The algorithm is a set of steps that are followed in order to solve a mathematical problem or to complete a computer process. The word "algorithm" is derived from the name of the

Arabian mathematician Al-Khwarizmi, which means a method, technique, or procedure. It is the most commonly used tool by the programmer for planning the program and solving the problems. It is an effective method for solving a problem.

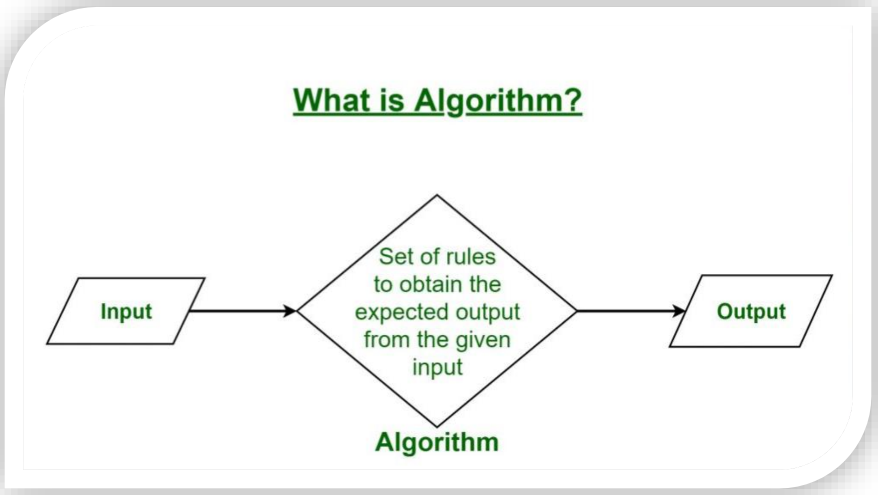
An algorithm is a specific set of instruction or rules for carrying out a procedure or solving a particular problem.

a. **Algorithm to calculate the area of a rectangular box.**

```
Step 1: Start
Step 2: Read the length of a box, say L
Step 3: Read breadth of a box, say B
Step 4: Area = L*B
Step 5: Display Area
Step 6: Stop
```

B. to Display the sum of even numbers from 1 to 100.

```
Step 1: Start
Step 2: Suppose n=1: Sum = 0
Step 3: Divide n by 2, If remainder is 0 then Sum = sum + n
Step 4: Increase n by 1
Step 5: If n<=100 then go to step 3 else go to step 6
Step 6: Show the sum
Step 7: Stop
```

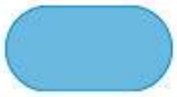
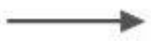
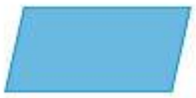
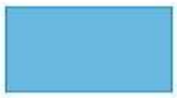
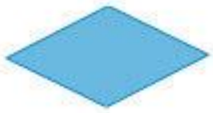

What is Algorithm?
Input
Set of rules to obtain the expected output from the given input
Algorithm
Output

- **Flowchart**

A flowchart is simply a graphical representation of steps. It shows steps in sequential order and is widely used in presenting the flow of algorithms, workflow or processes. Typically, a flowchart shows the steps as boxes of various kinds, and their order by connecting them with arrows.

Flowchart Symbols

Different flowchart shapes have different conventional meanings. The meanings of some of the more common shapes are as follows:

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectagle represents a process
	Decision	A diamond indicates a decision

1. **Draw a flowchart to find the greatest number among the 2 numbers.**

Algorithm:

1. Start

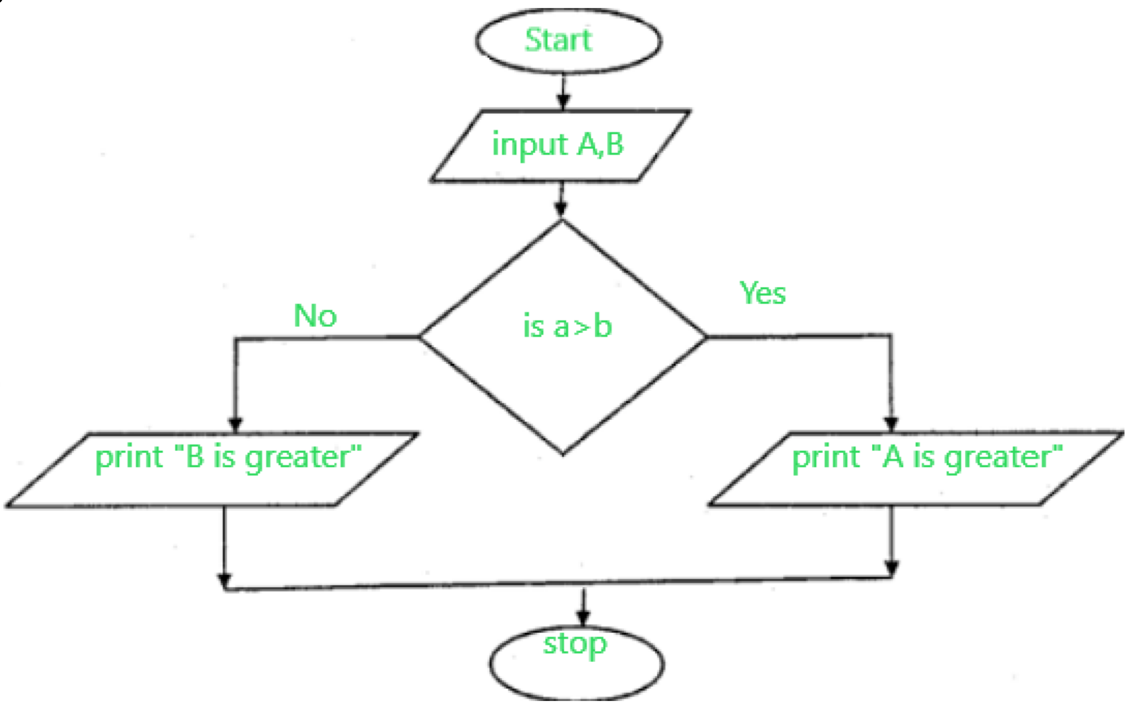
2. Input 2 variables from user

3. Now check the condition If a > b, goto step 4, else goto step 5.

4. Print a is greater, goto step 6

5. Print b is greater

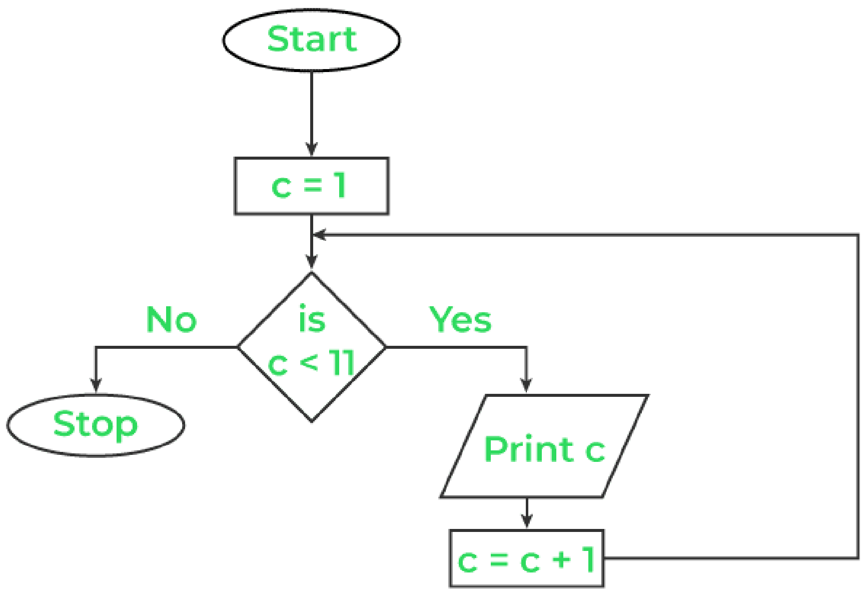
6. Stop



2. Draw a flowchart to print numbers from 1 to 10.

Algorithm:

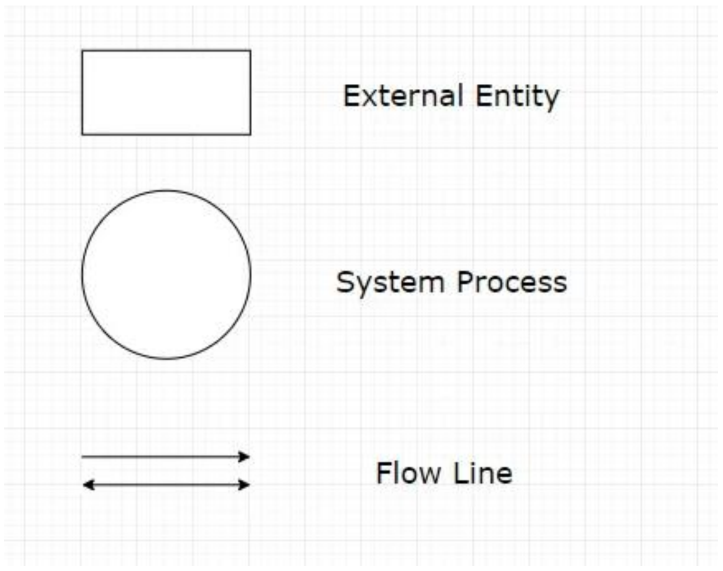
1. Start
2. Now initialize c = 1
3. Now we check the condition if c < 11, then goto step 4 otherwise goto step 6.
4. Print c
5. c = c + 1 then go to step 3
6. Stop



- **Context Diagram**

A context diagram is a visual representation of the relationship between data and business processes. It is also referred to as the Level O Data Flow Diagram, the Context diagram is the highest level in a Data Flow Diagram. It is a tool popular among Business Analysts who use it to understand the details and boundaries of the system to be designed in a project. It points out the flow of information between the system and external components.

Symbol Used In context Diagram



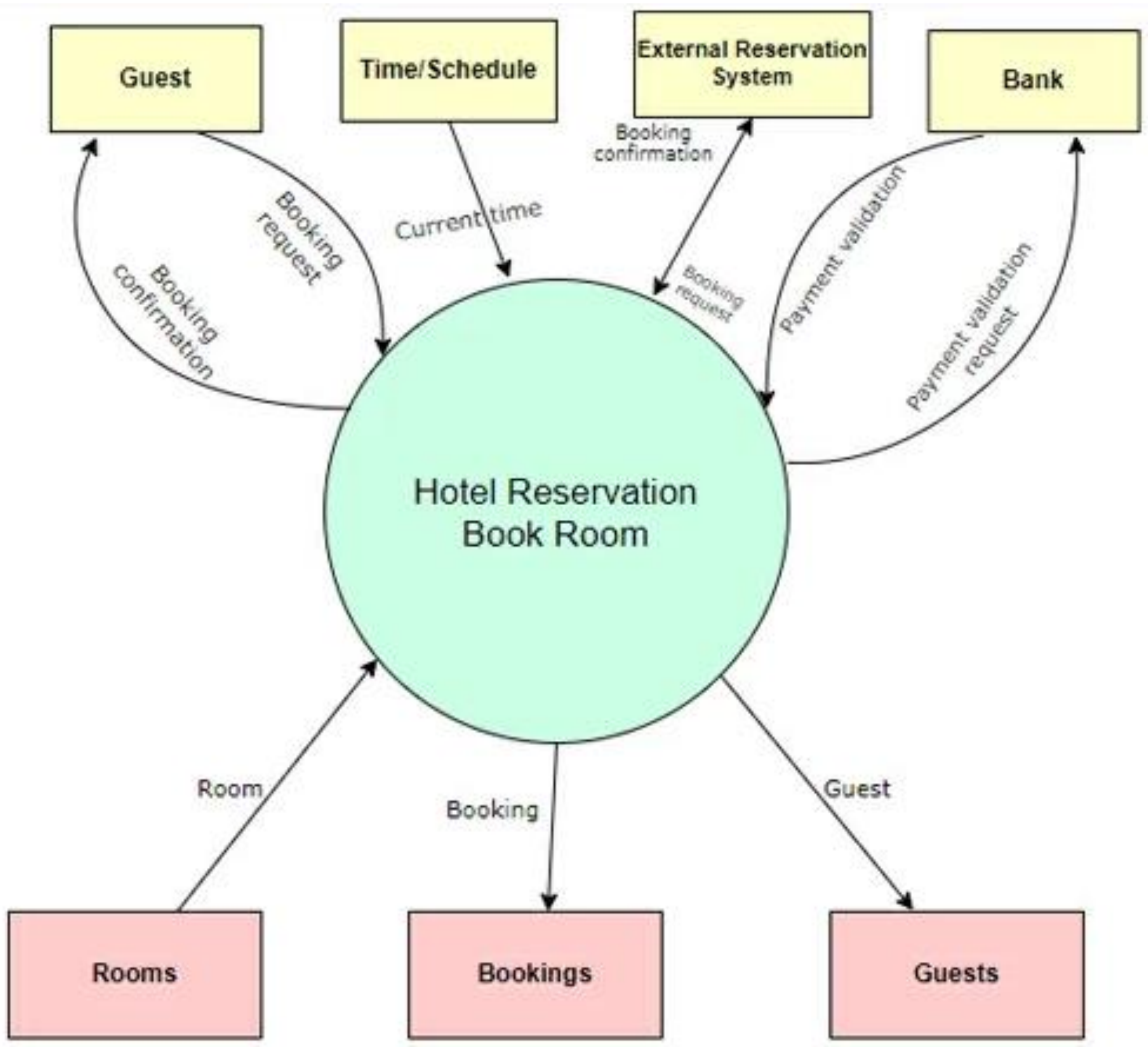
External Entity- an element in the system diagram that inputs data into the information system and retrieves processed data.

Process- refers to the entire process of the system. This is responsible for processing and distributing information to the entities of the system context diagram.

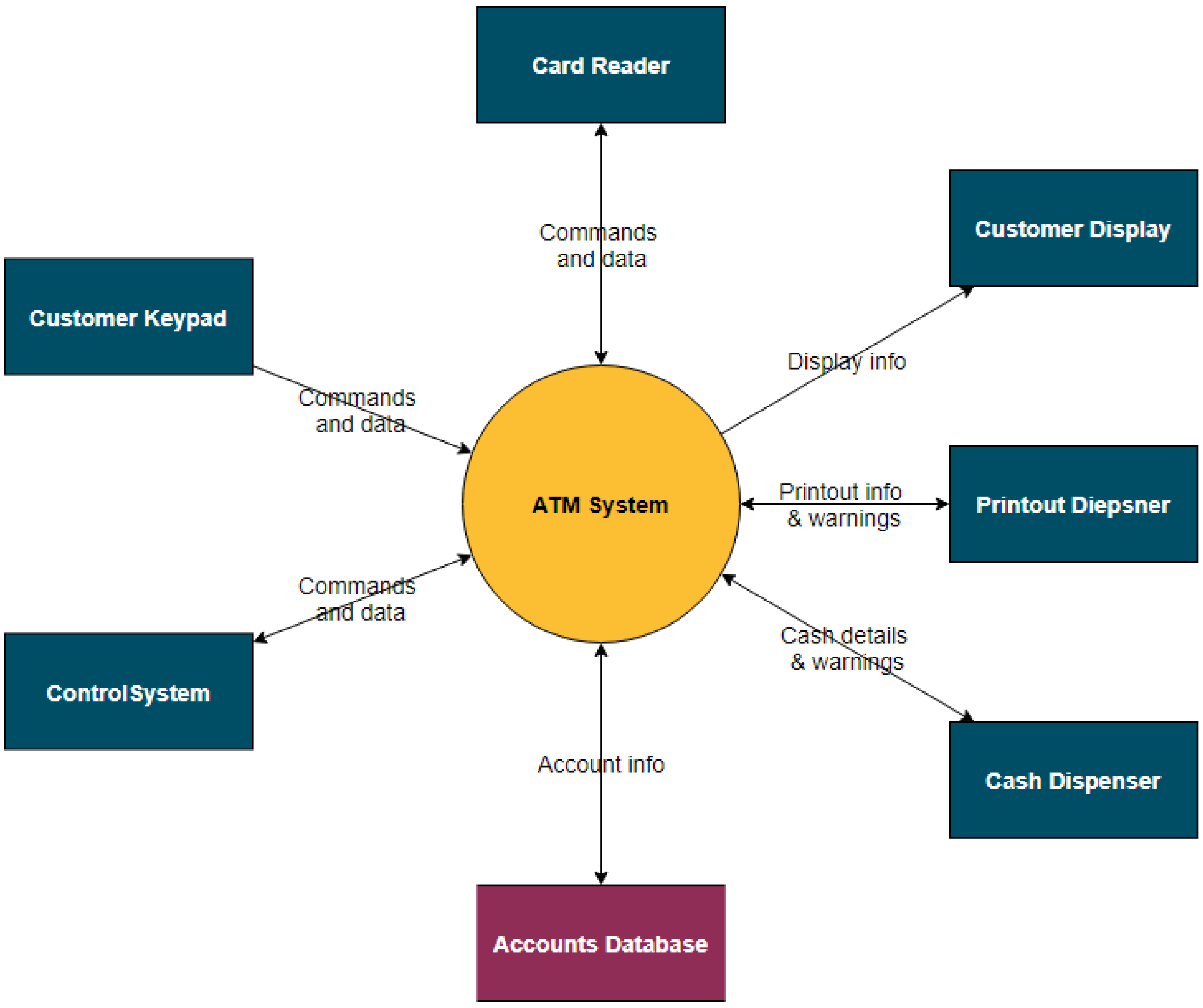
Flow Line- this element depicts the flow of the data within the system. It is supported by text to show what type of data is being sent.

Example of Context Diagram

Hotel Reservation System



Context Diagram of ATM System



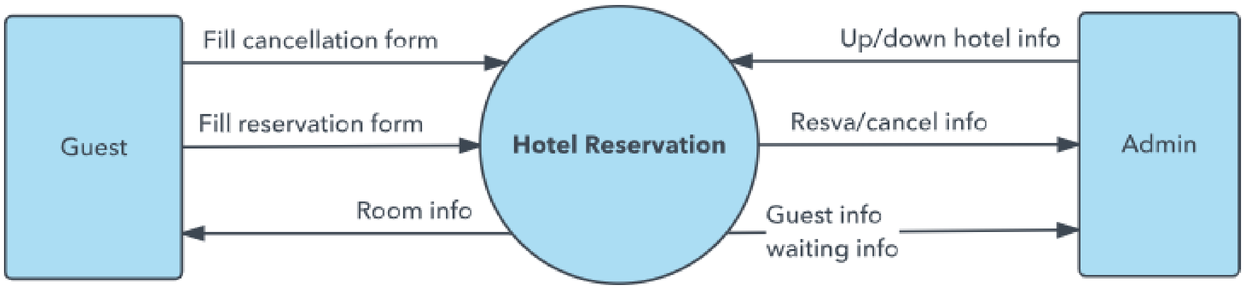
- ## Data Flow Diagram (DFD)

A data flow diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on the flow of information, where data comes from, where it goes and how it gets stored.

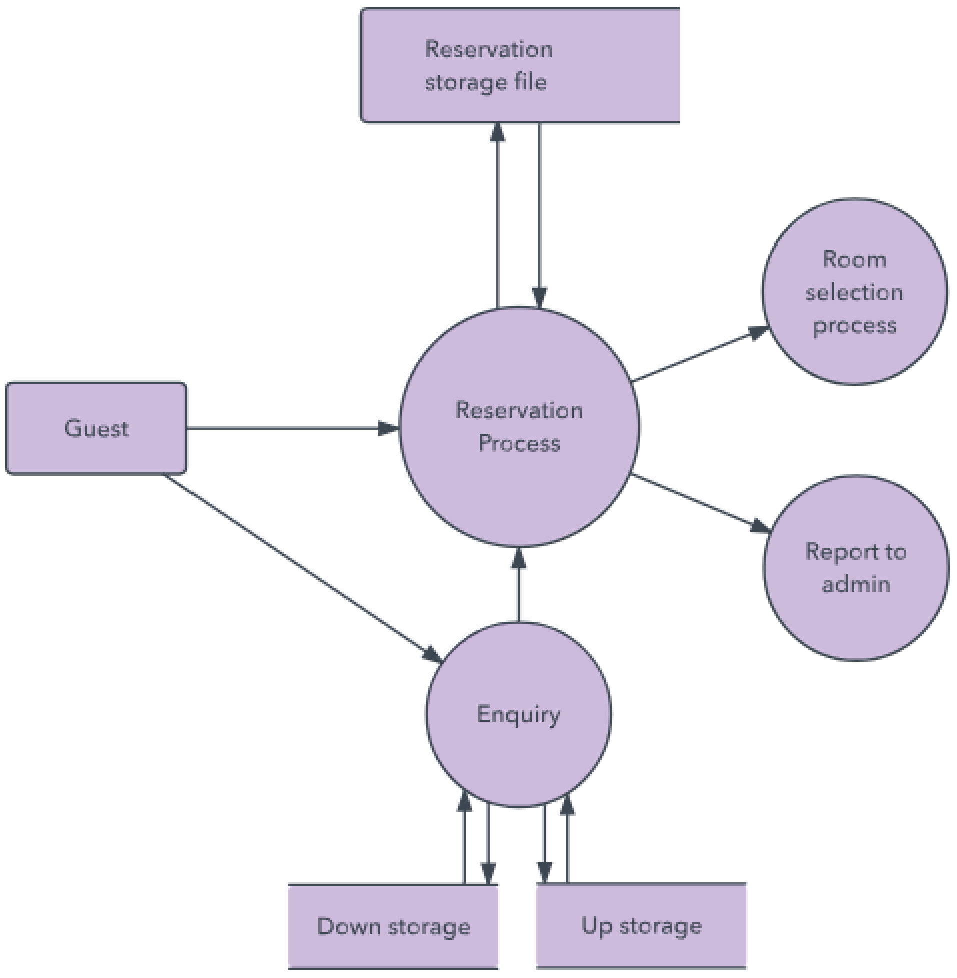
A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years.

DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish.

- DFD Level 0 is also called a Context Diagram. It’s a basic overview of the whole system or process being analyzed or modeled. It’s designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.



- DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its sub processes.

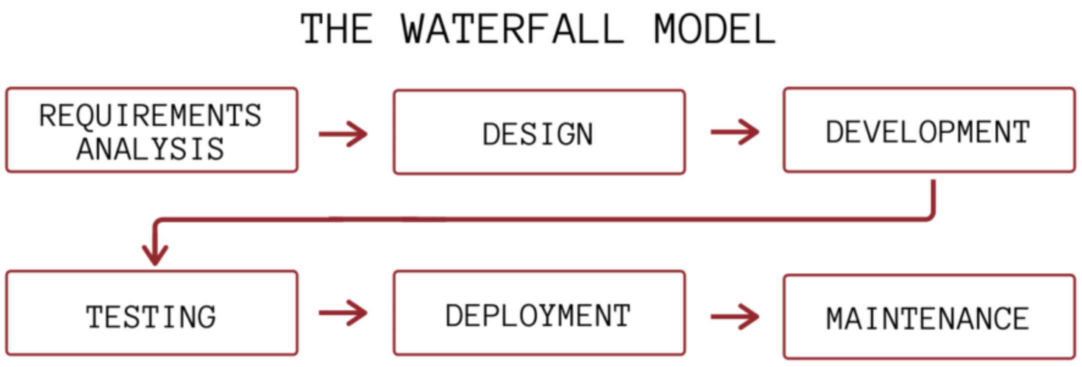


Software Development models

There are different software development life cycle models specify and design, which are followed during the software development phase. These models are also called "Software Development Process Models." Each process model follows a series of phase unique to its type to ensure success in the step of software development. Today, let’s cover some of the most popular software development models.

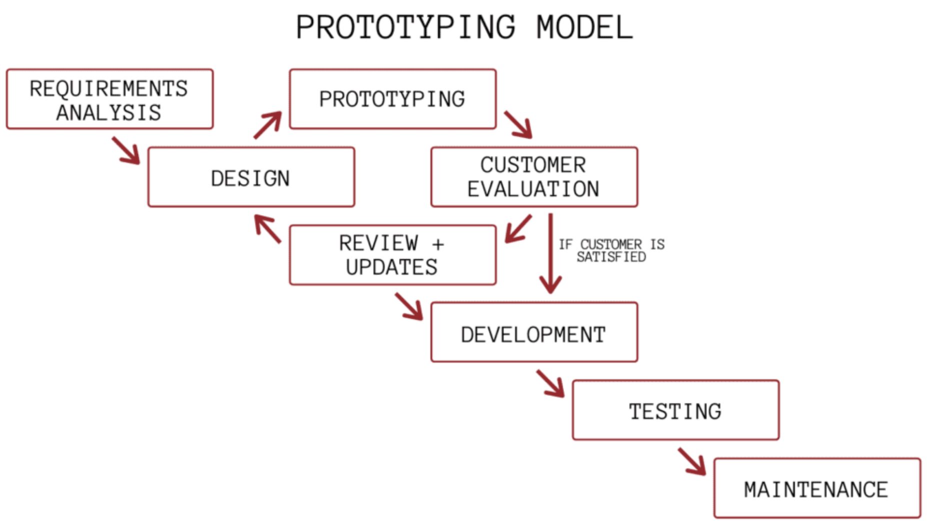
- **Waterfall Model**

The Waterfall model was the first approach to software development. As the name indicates, the process involves moving down through the linear development stages in order: analysis, design, development, testing, deployment, and maintenance. Every stage is well-defined with specific deliverables and milestones.



- **Prototyping Model**

The Prototyping model is centered around increasing the development team's understanding of the customer's wants/needs by creating prototypes. By creating a working small-scale replica of the desired software program, miscommunications or misunderstandings can be resolved before full development occurs.
Before developers begin working on the final product, they create a prototype of what they believe the customer wants. The prototype is developed, tested, and refined according to customer feedback. Once the prototype is accepted, the team begins developing the final product.



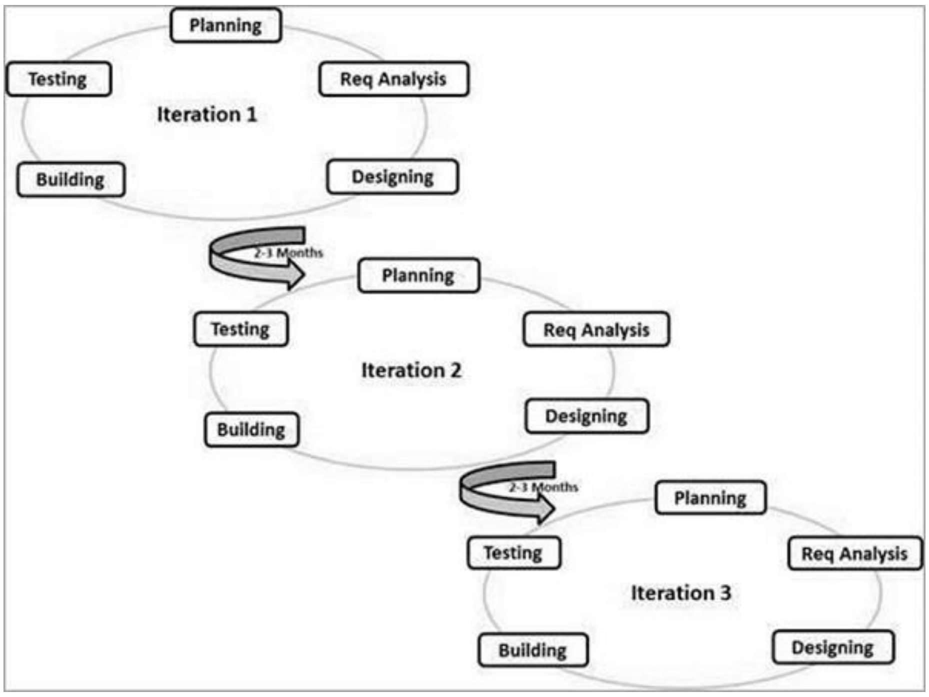
- **Agile Model**

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like −

- Planning
- Requirements Analysis
- Design

- Coding
- Unit Testing and
- Acceptance Testing.

At the end of the iteration, a working product is displayed to the customer and important stakeholders.


Planning
Testing
Req Analysis
Iteration 1
Building
Designing
2-3 Months
Planning
Testing
Req Analysis
Iteration 2
Building
Designing
2-3 Months
Planning
Testing
Req Analysis
Iteration 3
Building
Designing