Cihan University /Sulaymaniya

Software engineering

Computer Science Department

2nd stage

2023-2024

Lecturer: Wafaa Mustafa Hameed

WILEY

Software Engineering definition

- we can define *software engineering* as an engineering branch associated with the development of software product using well-defined scientific principles, methods and procedures. The outcome of software engineering is an efficient and reliable software product.

- IEEE defines software engineering as: The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software.

2

WILEY

Software product

Software engineering techniques

- Software engineering principles use two important techniques to reduce problem complexity:
  - *abstraction*
  - *decomposition*.

WILEY

Abstraction

- The principle of abstraction implies that a problem can be simplified by omitting irrelevant details. Abstraction is a powerful way of reducing the complexity of the problem.

WILEY

Decomposition

- In this technique, a complex problem is divided into several smaller problems and then the smaller problems are solved one by one.

- `A good decomposition of a problem should minimize interactions among various components. If the different subcomponents are interrelated, then the different components cannot be solved separately and the desired reduction in complexity will not be realized.

6

NEED OF SOFTWARE ENGINEERING
(Why Software Engineering)

The need of software engineering arises because of higher rate of change in user requirements and environment on which the software is working.

- **Large software** - It is easier to build a wall than to a house or building, likewise, as the size of software become large engineering has to step to give it a scientific process.

WILEY

- **Scalability**- If the software process were not based on scientific and engineering concepts, it would be easier to re-create new software than to scale an existing one.

- **Cost-** As hardware industry has shown its skills and huge manufacturing has lower down the price of computer and electronic hardware. But the cost of software remains high if proper process is not adapted.

- **Dynamic Nature-** The always growing and adapting nature of software hugely depends upon the environment in which the user works. If the nature of software is always changing, new enhancements need to be done in the existing one. This is where software engineering plays a good role.

Quality Management- Better process of software development provides better and quality software product.

WILEY

Importance of Software Engineering
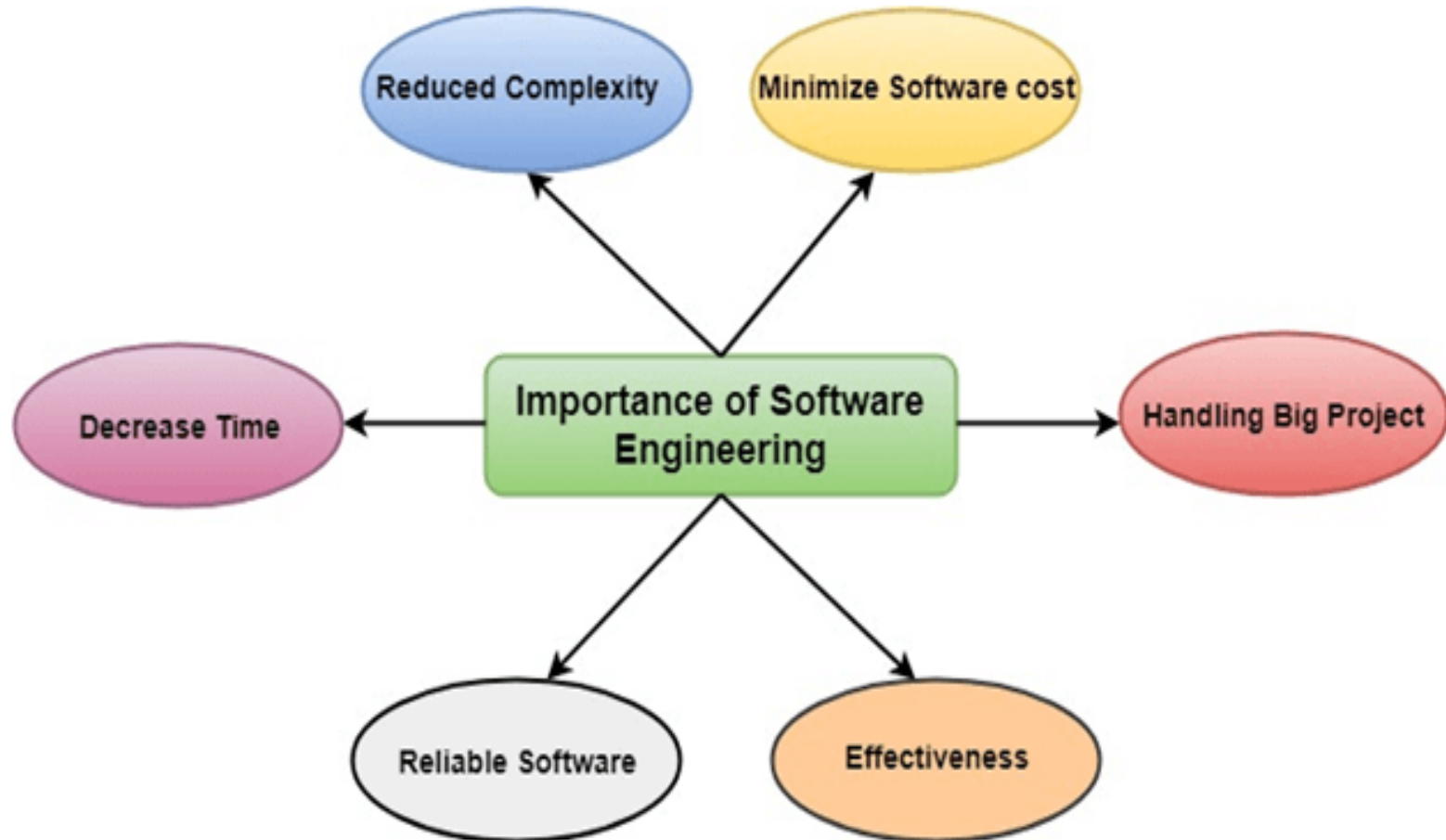
Importance of Software Engineering:

- **Reduces complexity:** Big software is always complicated and challenging to progress. Software engineering has a great solution to reduce the complication of any project. Software engineering divides big problems into various small issues. And then start solving each small issue one by one. All these small problems are solved independently to each other.

- **To minimize software cost**: Software needs a lot of hard work and software engineers are highly paid experts. A lot of manpower is required to develop software with a large number of codes. But in software engineering, programmers project everything and decrease all those things that are not needed. In turn, the cost for software productions becomes less as compared to any software that does not use software engineering method.

WILEY

- **To decrease time:** Anything that is not made according to the project always wastes time. And if you are making great software, then you may need to run many codes to get the definitive running code. This is a very time-consuming procedure, and if it is not well handled, then this can take a lot of time. So if you are making your software according to the software engineering method, then it will decrease a lot of time.

- **Handling big projects:** Big projects are not done in a couple of days, and they need lots of patience, planning, and management. And to invest six and seven months of any company, it requires heaps of planning, direction, testing, and maintenance. No one can say that he has given four months of a company to the task, and the project is still in its first stage. Because the company has provided many resources to the plan and it should be completed. So to handle a big project without any problem, the company has to go for a software engineering method.

WILEY

- **Reliable software:** Software should be secure, means if you have delivered the software, then it should work for at least its given time or subscription. And if any bugs come in the software, the company is responsible for solving all these bugs. Because in software engineering, testing and maintenance are given, so there is no worry of its reliability.

- **Effectiveness:** Effectiveness comes if anything has made according to the standards. Software standards are the big target of companies to make it more effective. So Software becomes more effective in the act with the help of software engineering.

WILEY

CHARACTERESTICS OF GOOD SOFTWARE

A software product can be judged by what it offers and how well it can be used. This software must satisfy on the following grounds:

- Operational

- Transitional

- Maintenance

Operational

This tells us how well software works in operations. It can be measured on:

- Budget
- Usability
- Efficiency
- Correctness
- Functionality
- Dependability
- Security
- Safety

WILEY

Transitional

This aspect is important when the software is moved from one platform to another:

- **Portability** (Portability deals with moving the component from one environment to another. Example: A game running on Windows XP **is** said to be portable if the same game can be run on Windows 7 without any change in the behavior of the game).

- Interoperability (Interoperability is an ability of one system to interact with another system. This interaction is between 2 different systems or 2 different applications all together. ... Like – **MS Word and Calculator** are 2 different application and they perform their expected behavior independently in the same operating system)

WILEY

- **Reusability** (In computer science and software engineering, reusability is **the use of existing assets in some form within the software product development process**; these assets are products of the software development life cycle and include code, software components, test suites, designs and documentation.)

- **Adaptability** (An adaptable software system can tolerate changes in its environment without external intervention. For example, a **dual-mode cell phone can find out by itself if any one of the two wireless standards it supports is available at its current location** and if so it starts using that standard.)

WILEY

Maintenance

This aspect briefs about how well a software has the capabilities to maintain itself in the ever-changing environment:

Modularity (Software modularity is the decomposition of a program into smaller programs with standardized interfaces.)

- **Maintainability** (Maintainability is defined as the probability of performing a successful repair action within a given time. ...

For example, if it is said that a particular component has a **90% maintainability for one hour**, this means that there is a 90% probability that the component will be repaired within an hour.)

WILEY

- **Flexibility** It creates new opportunities: Flexible software **frees businesses from hardware and software restrictions of the past**. For instance, suppose a business needs an industrial scanner to scan new inventory into their system. Older software might limit that company's options to one or two expensive, industrial scanners.

- **Scalability** For example, an **application program would be scalable if** it could be moved from a smaller to a larger operating system and take full advantage of the larger operating system in terms of performance and the larger number of users that could be handled.

- In short, Software engineering is a branch of computer science, which uses well-defined engineering concepts required to produce efficient, durable, scalable, in-budget and on-time software products

Software Applications

No clear breakdown of application types, following are some generally accepted overlapping categories

- **System software**

  It is a software designed to provide a platform for other software.

  Examples of system software include **operating systems like macOS, Linux, Android and Microsoft Windows, computational science software, game engines, search engines, industrial automation**, and software as a service applications.

- **Real-time software**

- Videoconference applications.

- Online gaming.

- Some e-commerce transactions.

- Chatting.

- **Business information software**
  - ☑ word processing programs.
  - ☑ accounts software.
  - ☑ billing software.
  - ☑ payroll software.
  - ☑ database software.
  - ☑ asset management software.

  - **Engineering & scientific software**
  - This software satisfies the needs of a scientific or engineering user to perform specific tasks. Such software is written for specific applications using principles, techniques, and formulae specific to that field. Examples are **software like MATLAB, AUTOCAD, ORCAD, etc**.

- Embedded software

Image processing systems found in medical imaging equipment.

Fly-by-wire control systems found in aircraft.

Motion detection systems in security cameras.

Traffic control systems found in traffic lights.

Timing and automation systems found in smart home devices.

- Personal application software
  - Microsoft suite of products (Office, Excel, Word, PowerPoint, Outlook, etc.)
  - Internet browsers like Firefox, Safari, and Chrome.
  - Mobile pieces of software such as Pandora (for music appreciation), Skype (for real-time online communication), and Slack (for team collaboration)

WILEY

- Communication software
- The best defined examples of communication software are **file transfer protocol (FTP), messaging software (**Skype, Messenge, whatsup**) and email**.

Software Processes

The term **software** specifies to the set of computer programs, procedures and associated documents (Flowcharts, manuals, etc.) that describe the program and how they are to be used.
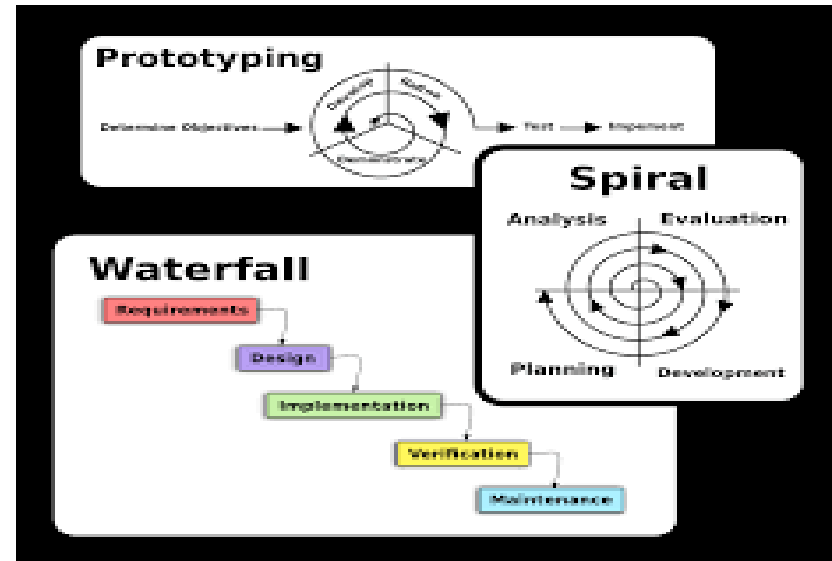
A software process is the set of activities and associated outcome that produce a software product.

WILEY

- Software engineers mostly carry out these activities. These are four key process activities, which are common to all software processes. These activities are:

  - **Software specifications:** The functionality of the software and constraints on its operation must be defined.

  - **Software development:** The software must be produced. to meet the requirement

  - **Software validation:** The software must be validated to ensure that it does what the customer wants.

  - **Software evolution:** The software must evolve to meet changing client needs.

The Software Process Model

- A software process model is an abstraction of the actual process. It can also be defined as

  a simplified representation of a software process. Each model represents a process from a specific perspective.

- Process models may contain activities, which are part of the software process, software product, and the roles of people involved in software engineering.

WILEY

Some **examples** of the types of software process models that may be produced

1. A workflow model: This shows the series of activities in the process along with their inputs, outputs and dependencies. The activities in this model perform human actions.
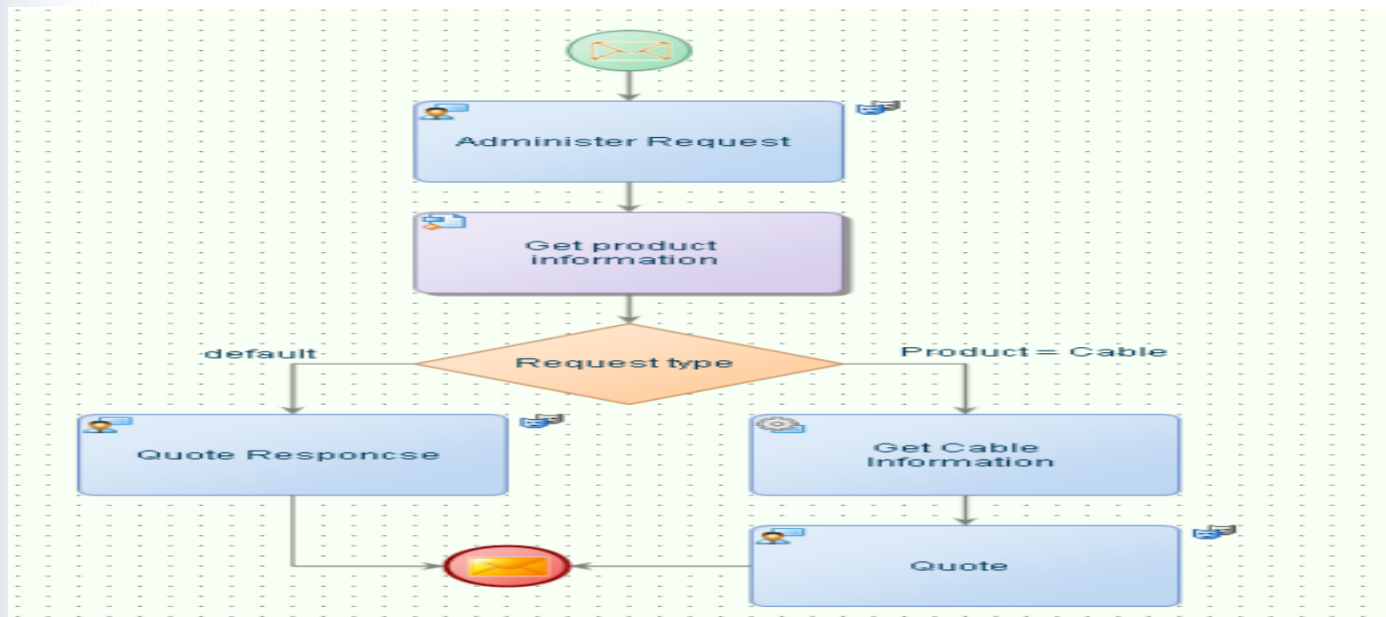
2. A dataflow or activity model: This represents the process as a set of activities, each of which carries out some data transformations. It shows how the input to the process, such as a specification is converted to an output such as a design. The activities here may be at a lower level than activities in a workflow model. They may perform transformations carried out by people or by computers.
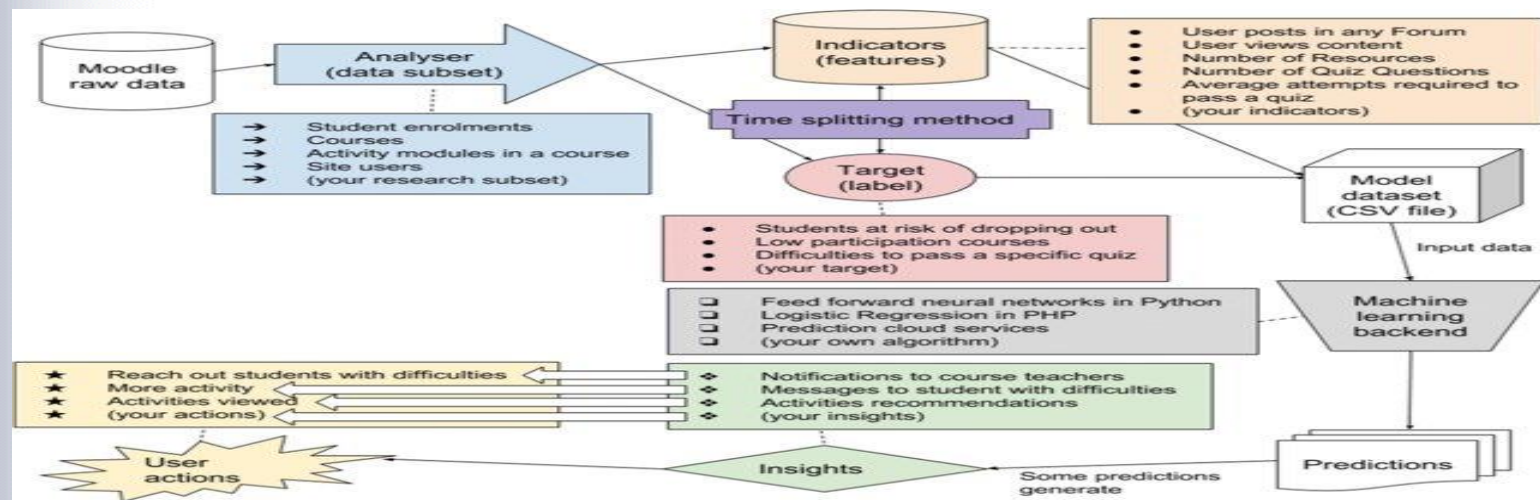
3. A role/action model: This means the roles of the people involved in the software process and the activities for which they are responsible.
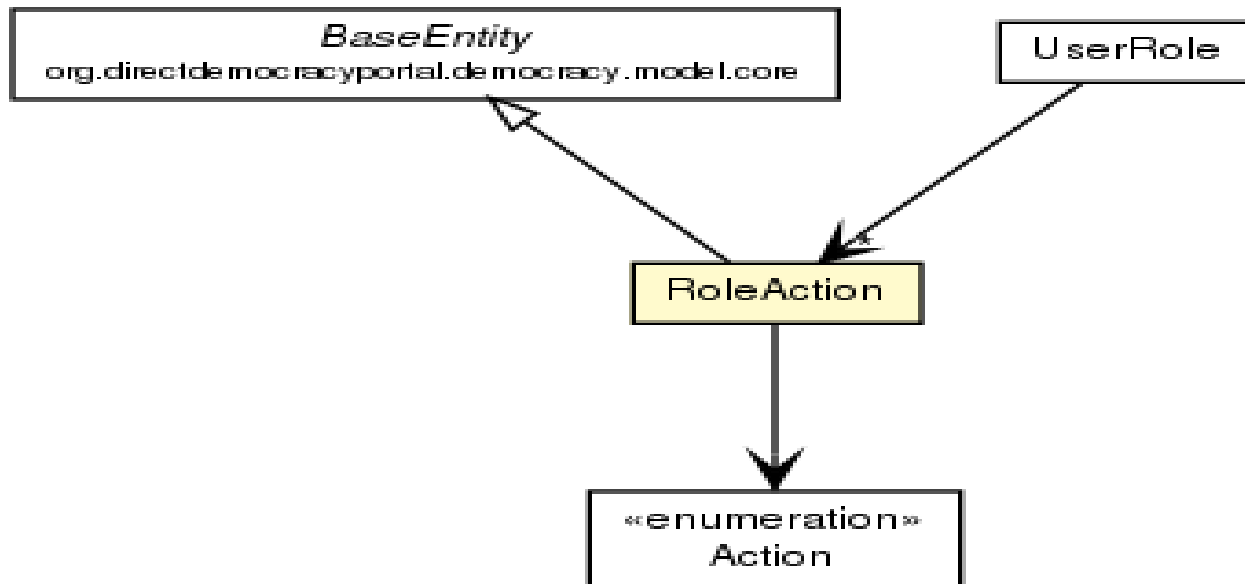
Why we need software Process?

- Common understanding of the activities, resources and constraints involved in software development.
- Creating processes helps
  - Find inconsistencies,
  - Redundancies; and
  - Omissions

35

WILEY

Software Crisis

- **Size:** Software is becoming more expensive and more complex with the growing complexity and expectation out of software. For example, the code in the consumer product is doubling every couple of years.

- **Quality:** Many software products have poor quality, i.e., the software products defects after putting into use due to ineffective testing technique. For example, Software testing typically finds 25 errors per 1000 lines of code.

- **Cost:** Software development is costly i.e. in terms of time taken to develop and the money involved

- **Delayed Delivery:** Serious schedule overruns are common. Very often the software takes longer than the estimated time to develop, which in turn leads to cost shooting up. For example, one in four large-scale development projects is never completed.

WILEY

Requirement Engineering

- **Requirements engineering (RE)** refers to the process of defining, documenting, and maintaining requirements in the engineering design process. Requirement engineering provides the appropriate mechanism to understand what the customer desires, analyzing the need, and assessing feasibility, negotiating a reasonable solution, specifying the solution clearly, validating the specifications and managing the requirements as they are transformed into a working system.

- Thus, requirement engineering is the disciplined application of proven principles, methods, tools, and notation to describe a proposed system's intended behavior and its associated constraints.

WILEY

Requirement Engineering Process

It is a five-step process, which includes –

- ☑ Feasibility Study
- ☑ Requirement Elicitation and Analysis
- ☑ Software Requirement Specification
- ☑ Software Requirement Validation
- ☑ Software Requirement Management

WILEY
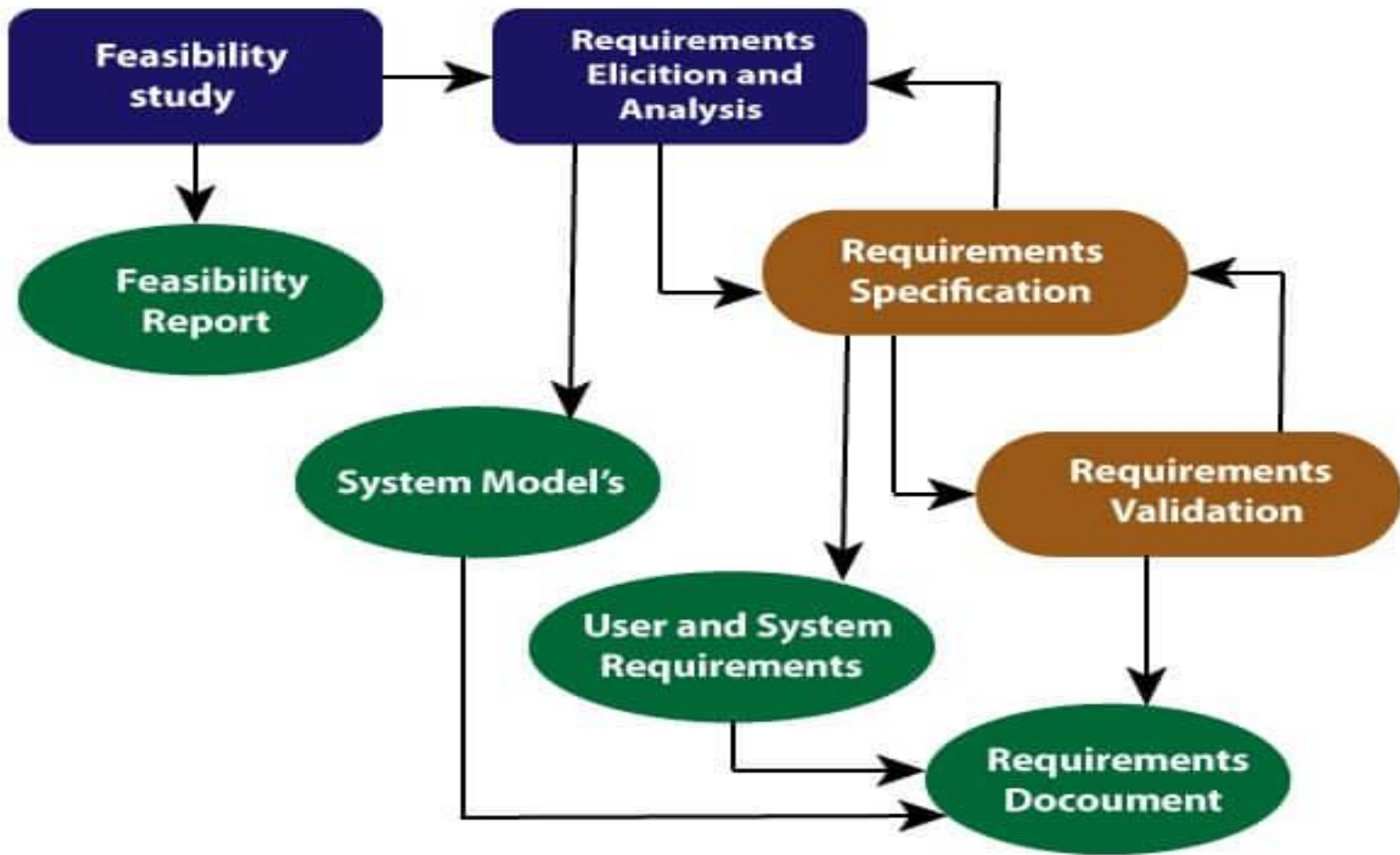
Requirement Engineering Process

1. Feasibility study

- The main aim of feasibility study is to determine whether it would be financially and technically feasible to develop the product.

- At first project managers or team leaders try to have a rough understanding of what is required to be done by visiting the client side. They study different input data to the system and output data to be produced by the system. They study what kind of processing is needed to be done on these data and they look at the various constraints on the behavior of the system.

- The objective behind the feasibility study is to create the reasons for developing the software that is acceptable to users, flexible to change and conformable to established standards.

WILEY

- After they have an overall understanding of the problem they investigate the different solutions that are possible. Then they examine each of the solutions in terms of what kind of resources required, what would be the cost of development and what would be the development time for each solution.

- Based on this analysis they pick the best solution and determine whether the solution is feasible financially and technically. They check whether the customer budget would meet the cost of the product and whether they have sufficient technical expertise in the area of development.

WILEY

Types of Feasibility:

- **Technical Feasibility** - Technical feasibility evaluates the current technologies, which are needed to accomplish customer requirements within the time and budget.

- **Operational Feasibility** - Operational feasibility assesses the range in which the required software performs a series of levels to solve business problems and customer requirements.

- **Economic Feasibility** - Economic feasibility decides whether the necessary software can generate financial profits for an organization.

42

WILEY

2. Requirement Elicitation and Analysis:

- This is also known as the **gathering of requirements**. Here, requirements are identified with the help of customers and existing systems processes, if available.

- Analysis of requirements starts with requirement elicitation. The requirements are analyzed to identify inconsistencies, defects, omission, etc. analystes describe requirements in terms of relationships and also resolve conflicts if any.

- The goal of the requirements gathering activity is to collect all relevant information from the customer regarding the product to be developed. This is done to clearly understand the customer requirements so that incompleteness and inconsistencies are removed.

- After all ambiguities, inconsistencies, and incompleteness have been resolved and all the requirements properly understood, the requirements specification activity can start. During this activity, the user requirements are systematically organized into a Software Requirements Specification (SRS) document.

WILEY

How to write a good SRS for your Project

☑ **What is SRS?**

A software requirements specification (SRS) is a description of a software system to be developed.

It lays out functional and non-functional requirements and may include a set of use cases that describe user interactions that the software must provide.

☑ **Why SRS?**

In order to fully understand one's project, it is very important that they come up with an SRS listing out their requirements, how are they going to meet them and how will they complete the project. It helps the team to save upon their time as they are able to comprehend how are going to go about the project. Doing this also enables the team to find out about the limitations and risks

45

WILEY

Requirements specification

- The customer requirements identified during the requirements gathering and analysis activity are organized into a SRS document. The important components of this document are functional requirements, the nonfunctional requirements, and the goals of implementation.

WILEY

Problems of Elicitation and Analysis

- Getting all, while only, the right people should be involved.
- Stakeholders often don't know what they want.
- Stakeholders express requirements in their terms.
- Stakeholders may have conflicting requirements.
- Requirement change during the analysis process.
- Organizational and political factors may influence system requirements.

WILEY

3. Software Requirement Specification:

- Software requirement specification is a kind of document which is created by a software analyst after the requirements collected from the various sources .

- the requirement received by the customer written in ordinary language. It is the job of the analyst to write the requirement in technical language so that they can be understood and beneficial by the development team.

WILEY

The models used at this stage include ER diagrams, data flow diagrams (DFDs), function decomposition diagrams (FDDs), data dictionaries.

- **Data Flow Diagrams:** Data Flow Diagrams (DFDs) are used widely for modeling the requirements.

- DFD shows the flow of data through a system.

- The system may be a company, an organization, a set of procedures, a computer hardware system, a software system, or any combination of the preceding.

- The DFD is also known as a data flow graph or bubble chart.

Data Flow Diagrams
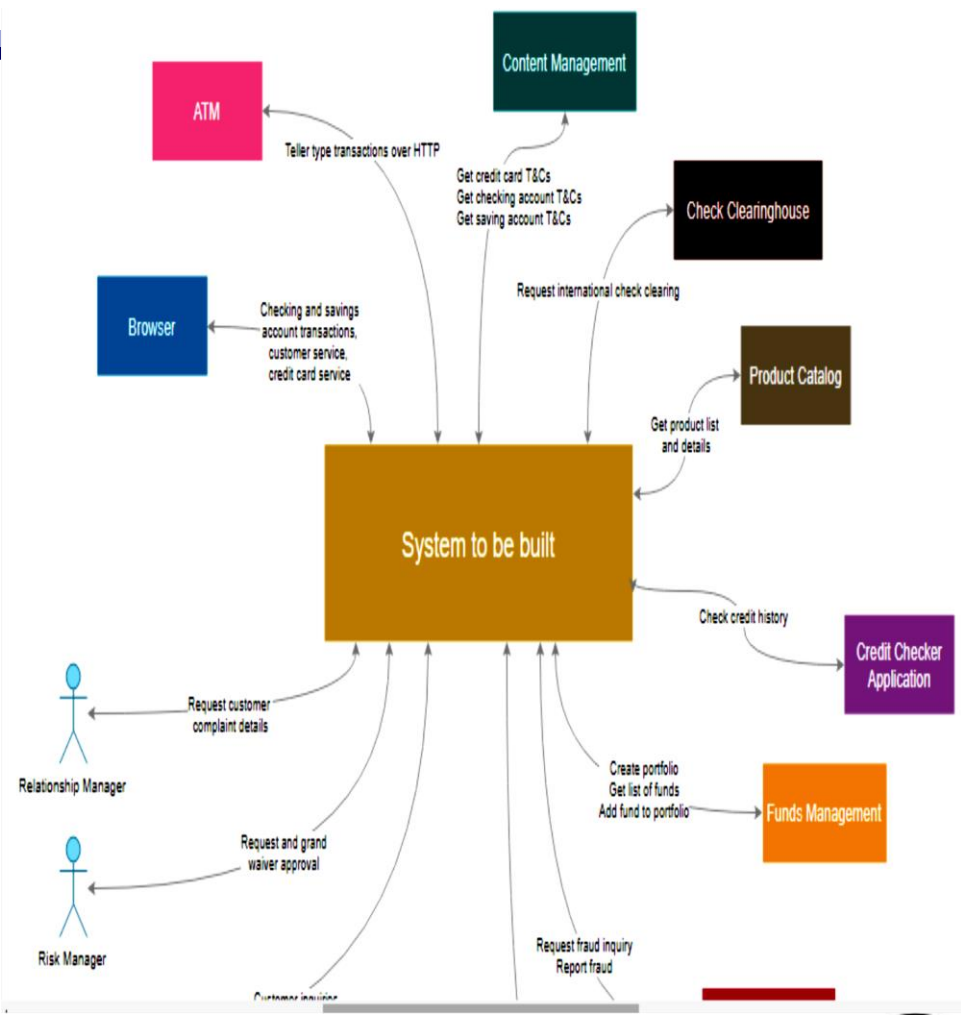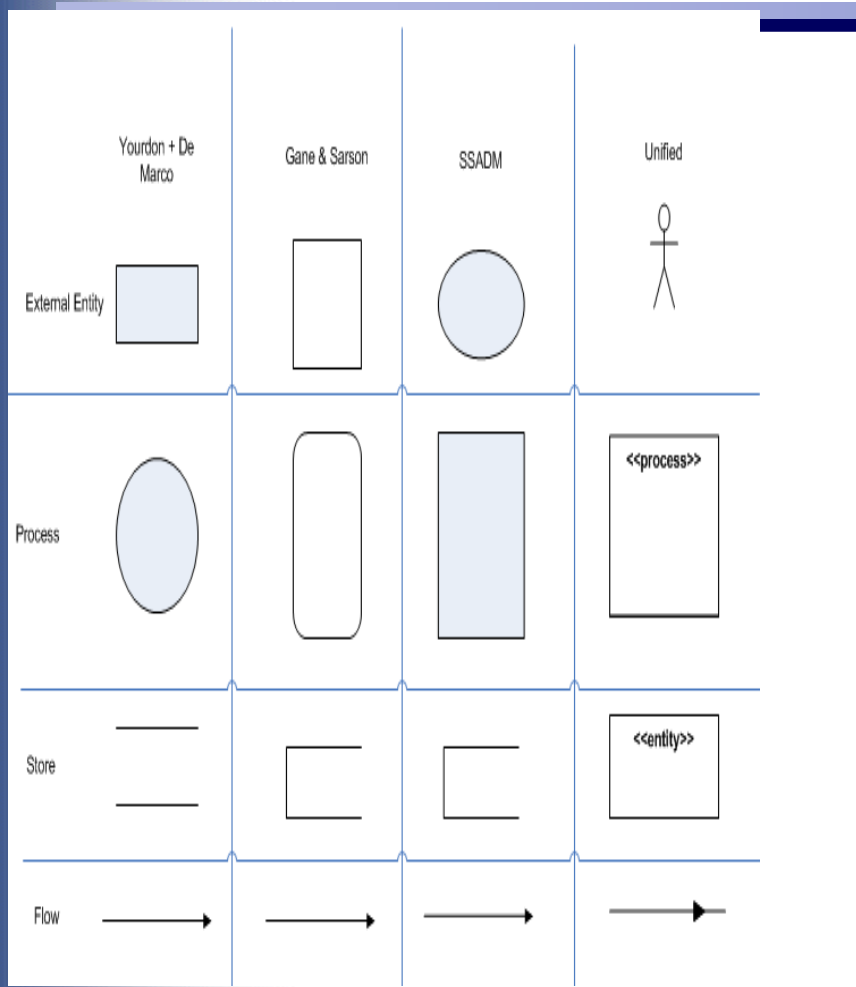
- **Data Dictionaries:** Data Dictionaries are simply repositories to store information about all data items defined in DFDs.

- At the requirements stage, the data dictionary should at least define customer data items, to ensure that the customer and developers use the same definition and terminologies.

DATA

emlployee_id	first_name	last_name	nin	dept_id
44	Simon	Martinez	HH 45 09 73 D	1
45	Thomas	Goldstein	SA 75 35 42 B	2
46	Eugene	Cornelsen	NE 22 63 82	2
47	Andrew	Petculescu	XY 29 87 61 A	1
48	Ruth	Stadick	MA 12 89 36 A	15
49	Barry	Scardelis	AT 20 73 18	2
50	Sidney	Hunter	HW 12 94 21 C	6
51	Jeffrey	Evans	LX 13 26 39 B	6
52	Doris	Berndt	YA 49 88 11 A	3
53	Diane	Eaton	BE 08 74 68 A	1

DATA DICTIONARY (METADATA)

Column	Data Type	Description
emlployee_id	int	Primary key of a table
first_name	nvarchar(50)	Employee first name
last_name	nvarchar(50)	Employee last name
nin	nvarchar(15)	National Identification Number
position	nvarchar(50)	Current postion title, e.g. Secretary
dept_id	int	Employee depamtnet. Ref: Departmetns
gender	char(1)	M = Male, F = Female, Null = unknown
employment_start_date	date	Start date of employment in organization.
employment_end_date	date	Employment end date.

Datædo

WILEY

- **Entity-Relationship Diagrams:** Another tool for requirement specification is the entity-relationship diagram, often called an "*E-R diagram*."

- It is a detailed logical representation of the data for the organization and uses three main constructs i.e. data entities, relationships, and their associated attributes.
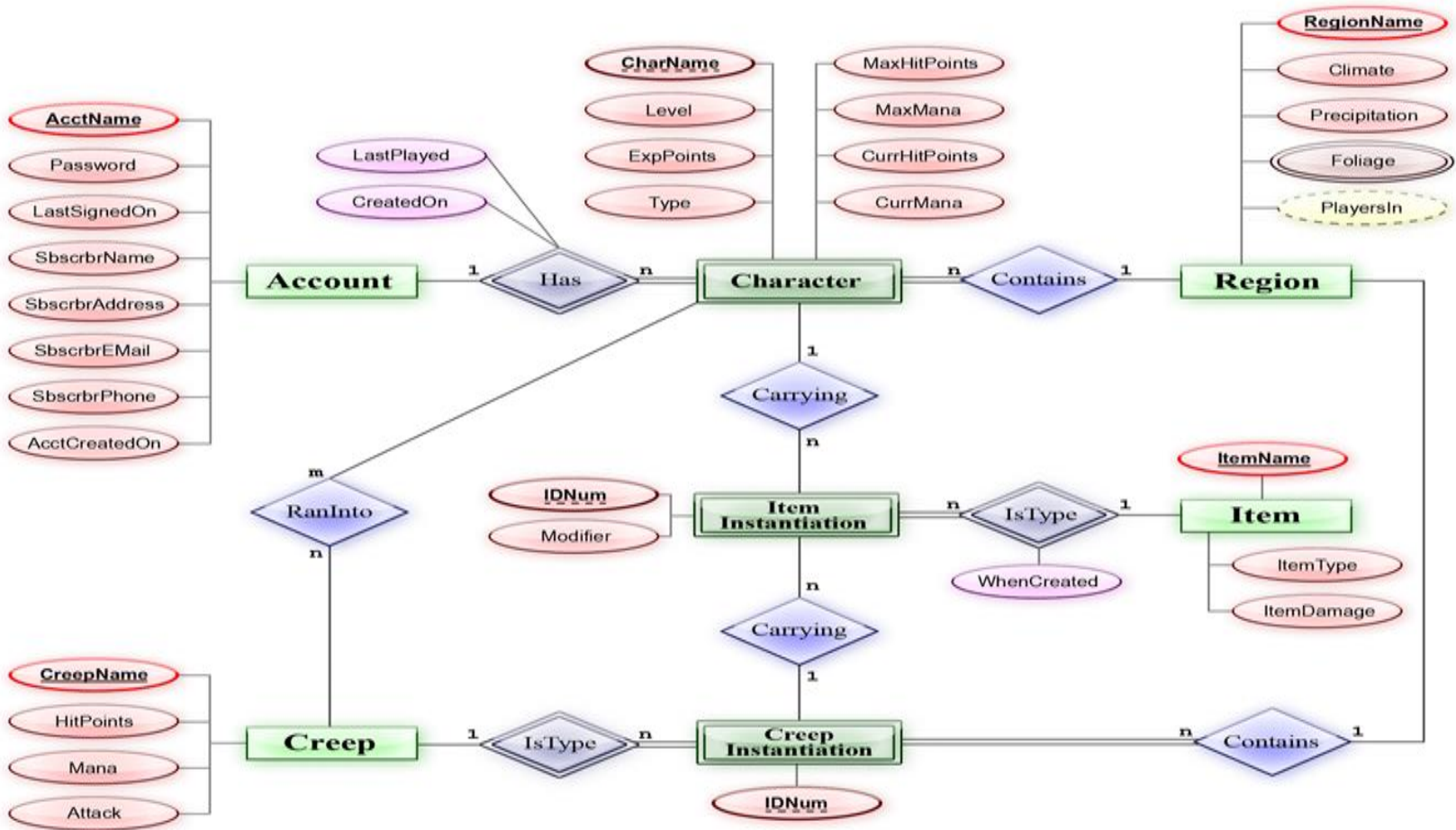
Functional Decomposition Diagram (FDD) is **a business planning tool** that depicts the hierarchy of business functions, processes, and sub processes within an organization that are later described in detail using process models
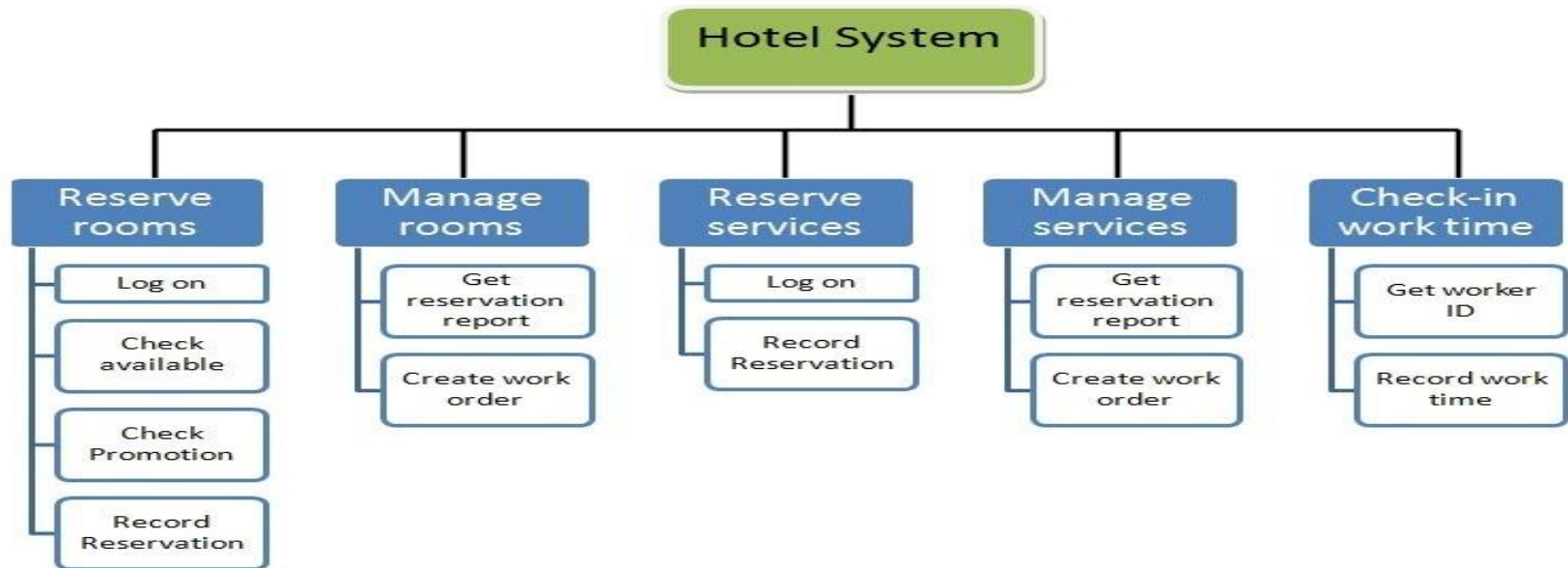
Prerequisite of Software requirements

- Collection of software requirements is the basis of the entire software development project. A complete Software Requirement Specifications should be:

- Clear
- Correct
- Consistent
- Coherent
- Comprehensible
- Modifiable
- Verifiable

- Prioritized
- Unambiguous
- Traceable
- Credible source

WILEY

Software Requirements: Largely software requirements must be categorized into two categories:

- **Functional Requirements:** Functional requirements define a function that a system or system element must be qualified to perform and must be documented in different forms. The functional requirements are describing the behavior of the system as it correlates to the system's functionality.

- Functional requirements define the basic system behavior. Essentially, they are **what the system does or must not do**, and can be thought of in terms of how the system responds to inputs. Functional requirements usually define if/then behaviors and include calculations, data input, and business processes.

WILEY

- **Non-functional Requirements:** This can be the necessities that specify the criteria that can be used to decide the operation instead of specific behaviors of the system.

- Nonfunctional Requirements (NFRs) define system attributes. They serve as constraints or restrictions on the design of the system across the different backlogs.

Non-functional requirements are divided into two main categories:

- **Execution qualities** like security and usability, which are observable at run time.

- **Evolution qualities** like testability, maintainability, extensibility, and scalability that embodied in the static structure of the software system.

4. Software Requirement Validation:

- After requirement specifications developed, the requirements discussed in this document are validated. The user might demand illegal, impossible solution or experts may misinterpret the needs.

- In the requirements validation process, different type of test to check the requirements mentioned need to perform :-

- If they can practically implement

- If they are correct and as per the functionality and specially of software

- If there are any ambiguities

- If they can describe all the requirements clearly.

A software requirements specification (SRS) can be

use to implement the validity .

WILEY

Requirements Validation Techniques:

- **Requirements reviews/inspections:** systematic manual analysis of the requirements.

- **Prototyping:** Using an executable model of the system to check requirements.

- **Test-case generation:** Developing tests for requirements to check testability.(**Customer** is responsible for acceptance testing.)

- **Automated consistency analysis:** checking for the consistency of structured requirements descriptions; for automatic detection of errors, such as type errors, nun determinism, missing cases, and circular definitions, in requirements specifications. .

WILEY

LIFE CYCLE MODEL

A software life cycle model (also called process model) is a descriptive and diagrammatic representation of the software life cycle. A life cycle model represents all the activities required to make a software product transit through its life cycle phases. It also captures the order in which these activities are to be undertaken.

WILEY

Different software life cycle models

- Many life cycle models have been proposed so far. Each of them has some advantages as well as some disadvantages. A few important and commonly used life cycle models are as follows:

  - **Classical Waterfall Model**
  - **Iterative Waterfall Model**
  - **Prototyping Model**
  - **Evolutionary Model**
  - **Spiral Model**

WILEY

1.CLASSICAL WATERFALL MODEL


The classic waterfall development model
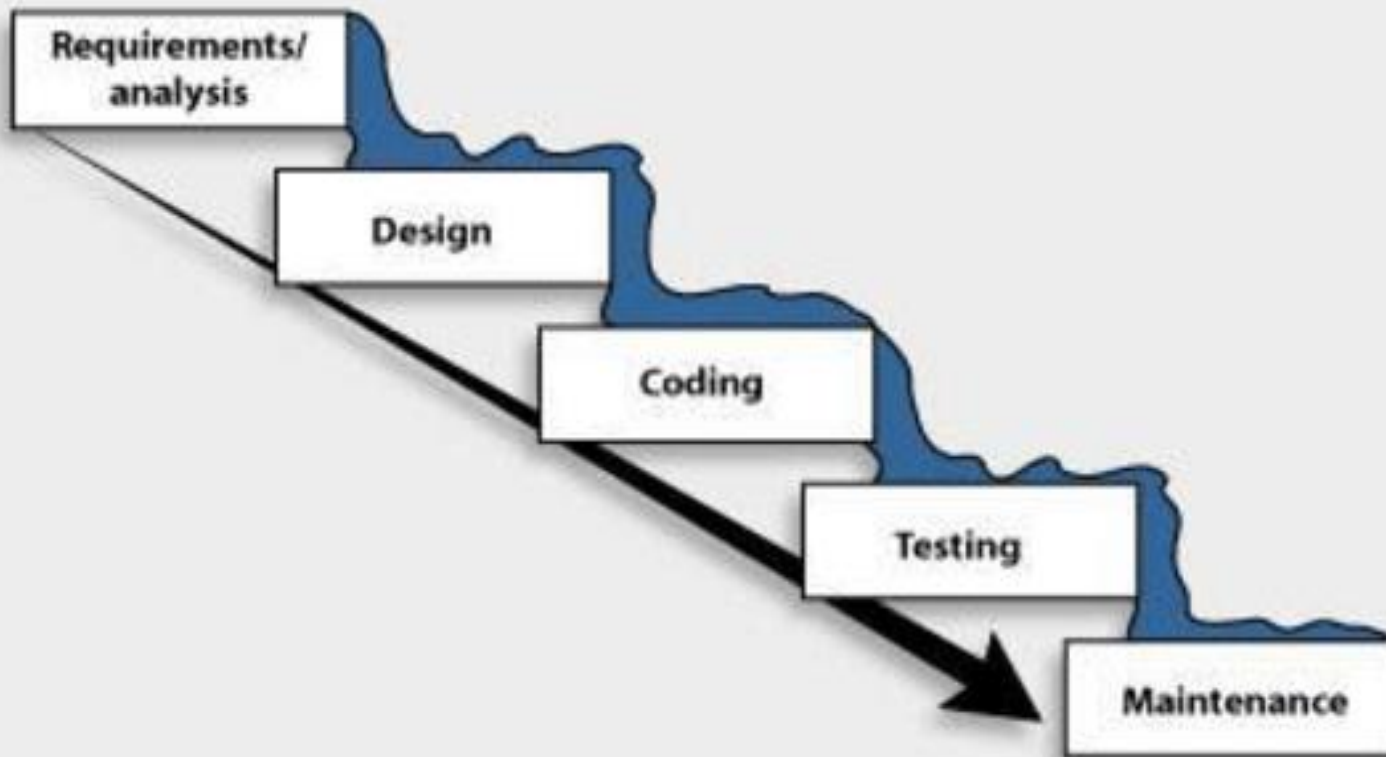
- The classical waterfall model is the most obvious way to develop software. Though the classical waterfall model is elegant and intuitively obvious, it is not a practical model in the sense that it cannot be used in actual software development projects. Thus, this model can be considered to be a theoretical way of developing software. But all other life cycle models are essentially derived from the classical waterfall model. So, in order to be able to appreciate other life cycle models it is necessary to learn the classical waterfall model.

- The classical waterfall model is an idealistic one since it assumes that no development error is ever committed by the engineers during any of the life cycle phases.

- The source of the defects can be many: oversight, wrong assumptions, use of inappropriate technology, communication gap among the project engineers, etc.

These defects usually get detected much later in the life cycle. For example, a design defect might go unnoticed till we reach the coding or testing phase. Once a defect is detected, the engineers need to go back to the phase where the defect had occurred and redo some of the work done during that phase and the subsequent phases to correct the defect and its effect on the later phases.
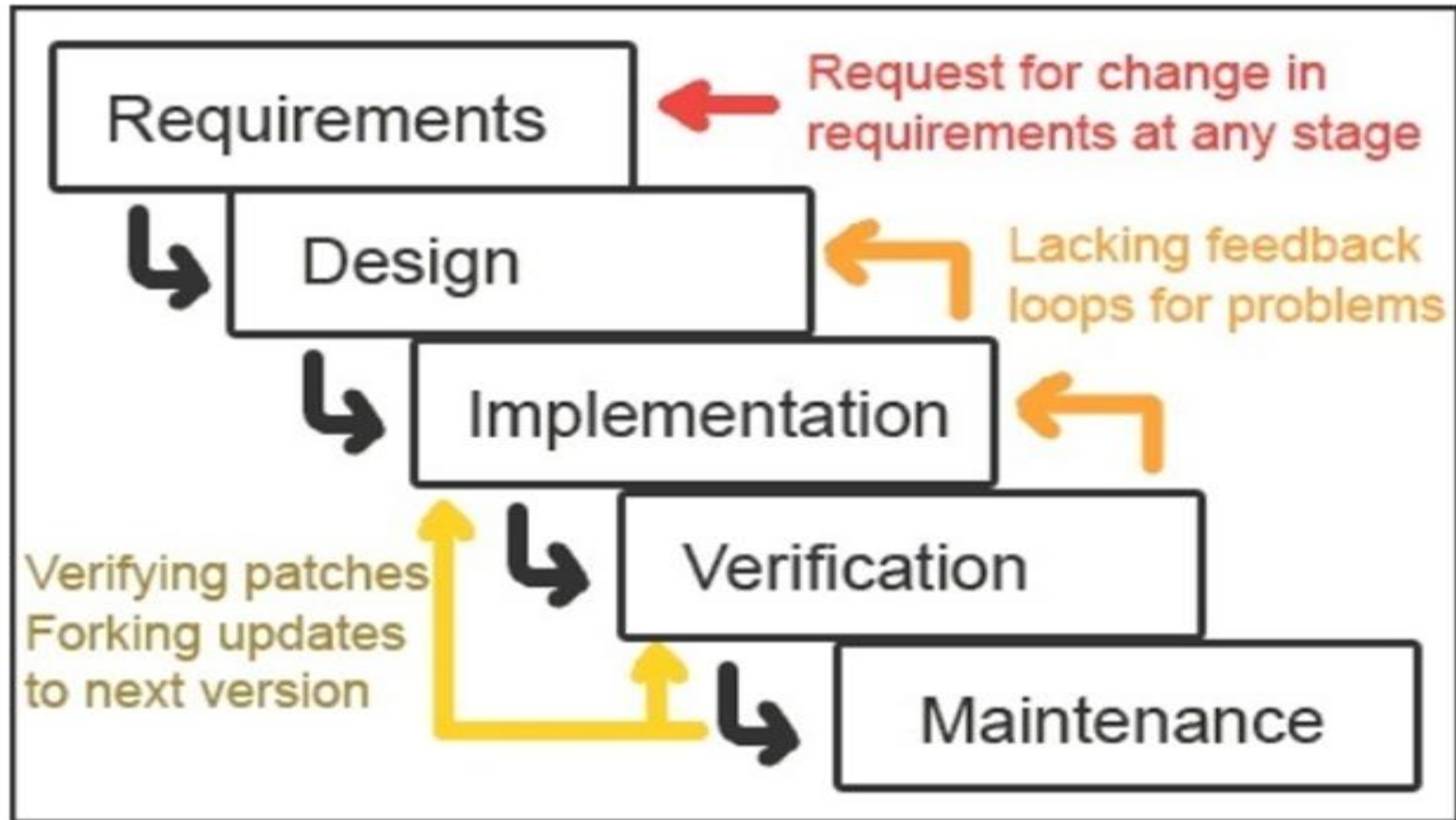
☑ Iterative waterfall model provide feedback paths for error correction as when detected later in a phase. Though errors are inevitable, but it is desirable to detect them in the same phase in which they occur. If so, this can reduce the effort to correct the bug.

Advantages and disadvantages of iterative waterfall model

- The advantage of this model is that there is a working model of the system at a very early stage of development which makes it easier to find functional or design flaws. Finding issues at an early stage of development enables to take corrective measures in a limited budget.

- The disadvantage with this SDLC (software development life cycle model is that it is applicable only to large and bulky software development projects. This is because it is hard to break a small software system into further small serviceable increments/modules.

WILEY

3. PRTOTYPING MODEL
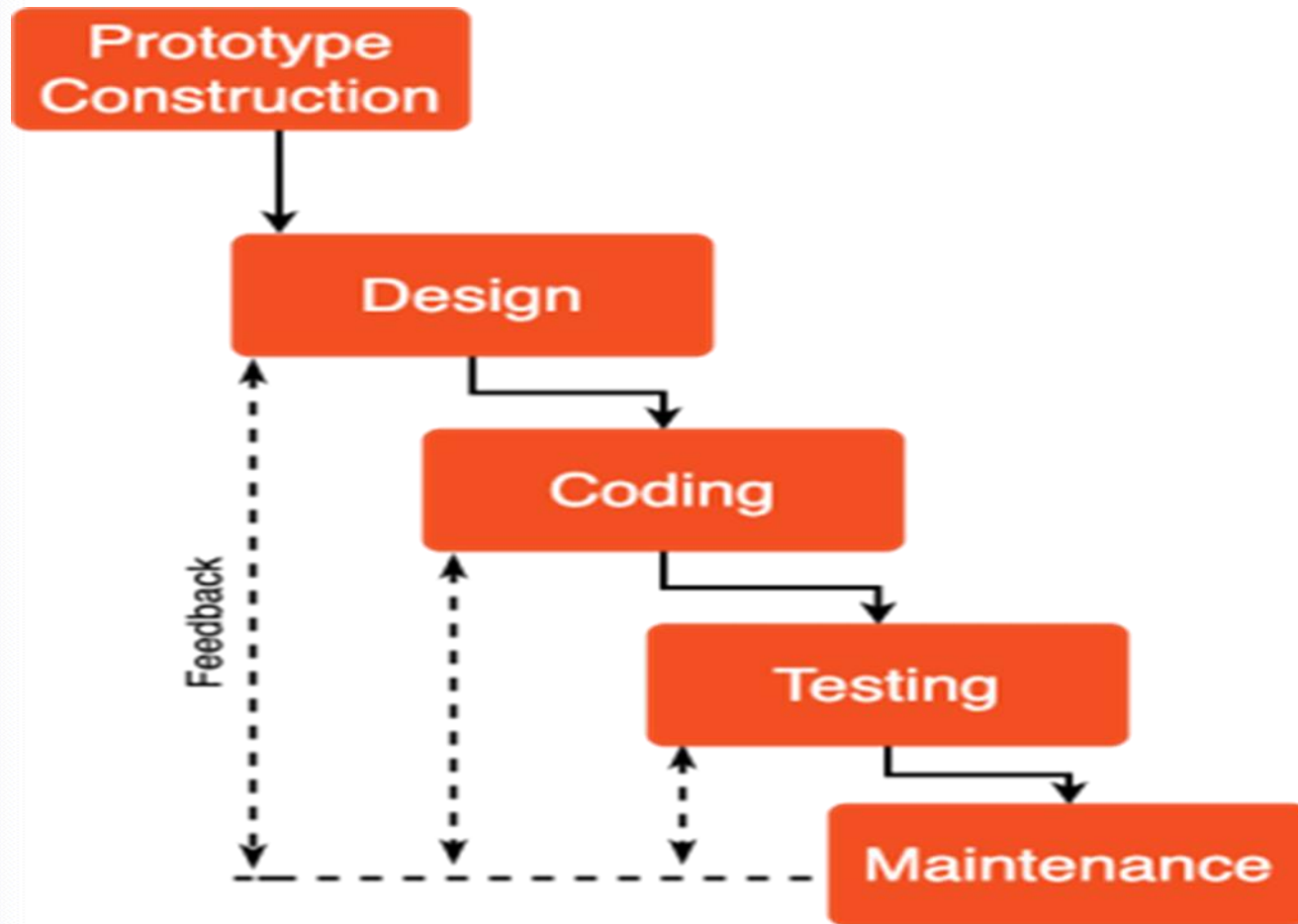
Prototyping Model - notepub.io

- A prototype is a toy implementation of the system.
  - A prototype usually exhibits limited functional capabilities,
  - low reliability, and inefficient performance compared to the actual software.
  - A prototype is usually built using several shortcuts.
- The shortcuts might involve using inefficient, inaccurate, or dummy functions. The shortcut implementation of a function, for example, may produce the desired results by using a table look-up instead of performing the actual computations. A prototype usually turns out to be a very crude version of the actual system.

WILEY

Need for a prototype in software development

An important purpose is to illustrate the input data formats, messages, reports, and the interactive dialogues to the customer. This is a valuable mechanism for gaining better understanding of the customer's needs:

- how the screens might look like
- how the user interface would behave
- how the system would produce outputs

WILEY

- Another reason for developing a prototype is that it is impossible to get the perfect product in the first attempt.

- Many researchers and engineers advocate that if you want to develop a good product you must plan to throw away the first version. The experience gained in developing the prototype can be used to develop the final product.

- A prototyping model can be used when technical solutions are unclear to the development team.

WILEY

- A developed prototype can help engineers to critically examine the technical issues associated with the product development. Often, major design decisions depend on issues like the response time of a hardware controller, or the efficiency of a sorting algorithm, etc. In such circumstances, a prototype may be the best or the only way to resolve the technical issues.

- A prototype of the actual product is preferred in situations such as:

  - User requirements are not complete
  - Technical issues are not clear

4. EVOLUTIONARY MODEL

- It is also called *successive versions model* or *incremental model*. At first, a simple working model is built. Subsequently it undergoes functional improvements & we keep on adding new functions till the desired system is built.

WILEY

EVOLUTIONARY MODEL

Iterations   Major releases
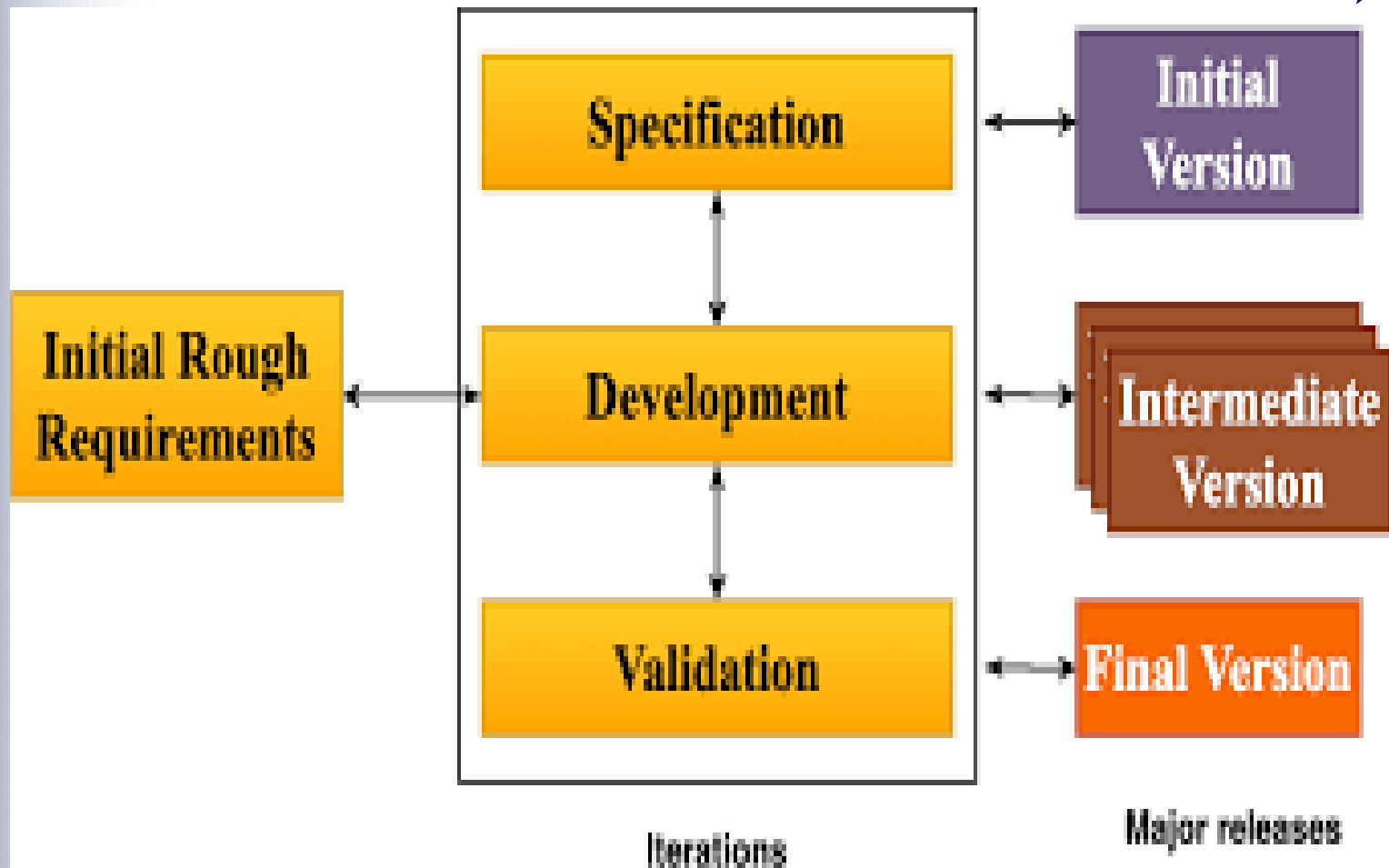
EVOLUTIONARY MODEL APPLICATIONS

- Large projects where you can easily find modules for incremental implementation. Often used when the customer wants to start using the core features rather than waiting for the full software.

- Also used in object oriented software development because the system can be easily portioned into units in terms of objects.

WILEY

Advantages and disadvantages of EVOLUTIONARY MODEL

Advantages:

- User gets a chance to experiment partially developed system

- Reduce the error because the core modules get tested thoroughly.

Disadvantages:

- It is difficult to divide the problem into several versions that would be acceptable to the customer which can be incrementally implemented & delivered.

WILEY

5.SPIRAL MODEL

The diagrammatic representation of this model appears like a spiral with many loops. The exact number of loops in the spiral is not fixed. Each loop of the spiral represents a phase of the software process.

For example, the inner most loop might be concerned with feasibility study, the next loop with requirements specification, the next one with design, and so on. Each phase in this model is split into four sectors (or quadrants)

SPIRAL MODEL
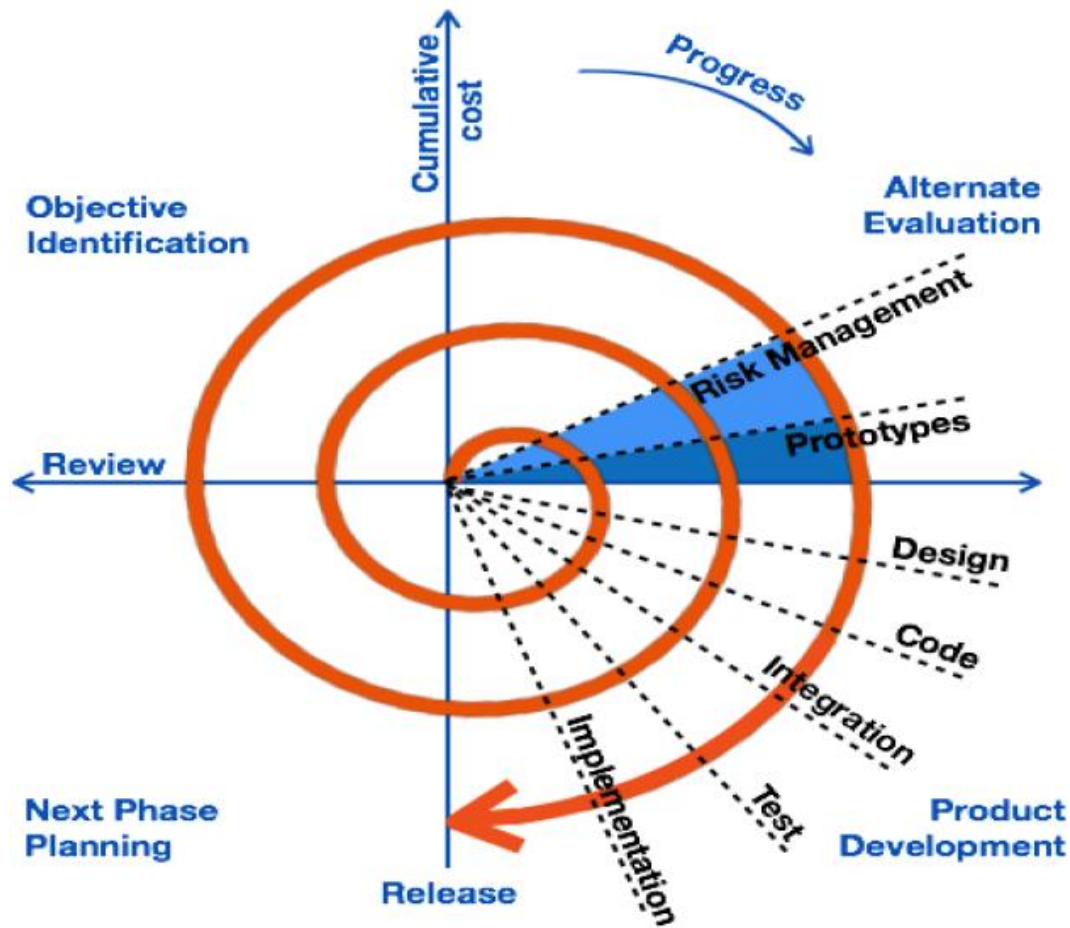
ACTIVITIES CARRIED OUT DURING EACH PHASE OF A SPIRAL MODEL.

- **First quadrant (Objective Setting)**
  - During the first quadrant, it is needed to identify the objectives of the phase.
  - Examine the risks associated with these objectives.
- **Second Quadrant (Risk Assessment and Reduction)**
  - A detailed analysis is carried out for each identified project risk.
  - Steps are taken to reduce the risks. For example, if there is a risk that the requirements are inappropriate, a prototype system may be developed.

WILEY

- **Third Quadrant (Development and Validation)**

  - Develop and validate the next level of the product after resolving the identified risks.

- **Fourth Quadrant (Review and Planning)**

  - Review the results achieved so far with the customer and plan the next iteration around the spiral.

  - Progressively more complete version of the software gets built with each iteration around the spiral.

Circumstances to use spiral model

- The spiral model is called a meta model since it encompasses all other life cycle models. Risk handling is inherently built into this model. The spiral model is suitable for development of technically challenging software products that are prone to several kinds of risks. However, this model is much more complex than the other models – this is probably a factor deterring its use in ordinary projects.

WILEY

Comparison of different life-cycle models

The classical waterfall model can be considered as the basic model and all other life cycle models as embellishments of this model. However, the classical waterfall model cannot be used in practical development projects, since this model supports no mechanism to handle the errors committed during any of the phases.

WILEY

- This problem is overcome in the iterative waterfall model. The iterative waterfall model is probably the most widely used software development model evolved so far. This model is simple to understand and use. However this model is suitable only for well-understood problems; it is not suitable for very large projects and for projects that are subject to many risks

- The prototyping model is suitable for projects for which either the user requirements or the underlying technical aspects are not well understood. This model is especially popular for development of the user-interface part of the projects.

- The evolutionary approach is suitable for large problems which can be decomposed into a set of modules for incremental development and delivery. This model is also widely used for object-oriented development projects. Of course, this model can only be used if the incremental delivery of the system is acceptable to the customer.

The spiral model is called a meta model since it encompasses all other life cycle models. Risk handling is inherently built into this model. The spiral model is suitable for development of technically challenging software products that are prone to several kinds of risks. However, this model is much more complex than the other models – this is probably a factor deterring its use in ordinary projects.

WILEY

Design:

- The goal of the design phase is to transform the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language. In technical terms, during the design phase the software architecture is derived from the **SRS document.**

- Two distinctly different approaches are available:
  - **The traditional design approach**
  - **Object-oriented design approach.**

WILEY

- **A Software Requirements Specification** (SRS)

is a document that describes the nature of a project, software or application. In simple words, SRS document is a manual of a project provided, it is prepared before you kick-start a project/application.

This document is also known by the names SRS report, software document.

1.Traditional design approach

- Traditional design consists of two different activities;

- first a structured analysis of the requirements specification is carried out where the detailed structure of the problem is examined. This is followed by

- a structured design activity, During structured design, the results of structured analysis are transformed into the software design.

WILEY

2. **Object-oriented design approach**

- In this technique, various objects that occur in the problem domain and the solution domain are first identified, and the different relationships that exist among these objects are identified. The object structure is further refined to obtain the detailed design.

Coding and unit testing

- The purpose of the coding phase (sometimes called the implementation phase) of software development is to translate the software design into source code.

- Each component of the design is implemented as a program module. The end-product of this phase is a set of program modules that have been individually tested.

- During this phase, each module is unit tested to determine the correct working of all the individual modules. It involves testing each module in isolation as this is the most efficient way to debug the errors identified at this stage.

WILEY

Integration and system testing

- Integration of different modules is undertaken once they have been coded and unit tested.

- During the integration and system testing phase, the modules are integrated in a planned manner. The different modules making up a software product are almost never integrated in one shot.

- Integration is normally carried out incrementally over a number of steps. During each integration step, the partially integrated system is tested and a set of previously planned modules are added to it.

- Finally, when all the modules have been successfully integrated and tested, system testing is carried out. The goal of system testing is to ensure that the developed

System Testing Stages

System testing usually consists of three different kinds of testing activities:

- α – testing(unit testing): It is the system testing performed by the development team, Individual components are tested independently; Components may be functions or objects or coherent groupings of these entities.

- β –testing(system testing): It is the system testing performed by a friendly set of customers. Testing of the system as a whole, Testing of emergent properties is particularly important.

  Acceptance testing: It is the system testing performed by the customer himself after the product delivery to determine whether to accept or reject the delivered product.

WILEY

Maintenance:

- Maintenance of a typical software product requires much more than the effort necessary to develop the product itself.

- Many studies carried out in the past confirm this and indicate that the relative effort of development of a typical software product to its maintenance effort is roughly in the 40:60 ratios

WILEY

Maintenance Activities

- Maintenance involves performing any one or more of the following three kinds of activities:
  - Correcting errors that were not discovered during the product development phase. This is called corrective maintenance.
  - Improving the implementation of the system, and enhancing the functionalities of the system according to the customer's requirements. This is called perfective maintenance.
  - Porting the software to work in a new environment. For example, porting may be required to get the software to work on a new computer platform or with a new operating system. This is called adaptive maintenance.

WILEY

Software reuse

WILEY

Introduction of Software Reuse

- Software reuse is the process of creating software systems from existing software rather than building software systems from scratch.

- Something that was originally written for a different project and implementation will usually be recognized as reuse.

- Code reuse is the idea that a partial or complete computer program written at one time can be used in another program written at a later time.

- The reuse of programming code is a common technique which attempts to save time and energy by reducing redundant work.

Why Software Reuse

- A good software reuse process facilitates the increase of productivity, quality, and reliability, performance and decrease of costs, effort, risk and implementation time.

- An initial investment is required to start a software reuse process, but that investment pays for itself in a few reuses.

WILEY

Software reuse

- In most engineering disciplines, systems are designed by composing existing components that have been used in other systems.

- Software engineering has been more focused on original development but it is now recognized that to achieve better software, more quickly and at lower cost, we need to adopt a design process that is based on systematic software reuse.

WILEY

Reuse-based software engineering

- **Application system reuse**

  The whole of an application system may be reused either by incorporating it without change into other systems (COTS reuse) or by developing application families.

- **Component reuse**

  Components of an application from sub-systems to single objects may be reused.

- **Object and function reuse**

  Software components that implement a single well-defined object or function may be reused.

WILEY

Benefits

- Increased dependability
- Reduced process risk
- Standards compliance
- Accelerated development

WILEY

Problems

- Increased maintenance costs
- Lack of tool support
- Not-invented-here syndrome
- Finding, understanding and Adapting reusable components

WILEY

Reuse planning factors

- The development schedule for the software.

- The expected software lifetime.

- The background, skills and experience of the development team.

- The criticality of the software and its non-functional requirements.

- The application domain.

- The execution platform for the software.

WILEY

Concept reuse

- When you reuse program or design components, you have to follow the Design decisions made by the original developer of the component.

- This may limit the opportunities for reuse.

- However, a more abstract form of reuse is concept reuse when a Particular approach is described in an implementation independent way And an implementation is then developed.

- The two main approaches to concept reuse are:

  - Design patterns
  - Generative programming