# Software Engineering

Introduction to software engineering, Evolution of Software Engineering (Content beyond Syllabus)

# Syllabus:

https://docs.google.com/document/d/11HgCM72iRdIJGMLEB_n3rpQgXu9yyPxW/edit?usp=sharing&ouid=107776947515336695274&rtpof=true&sd=true

# Activity

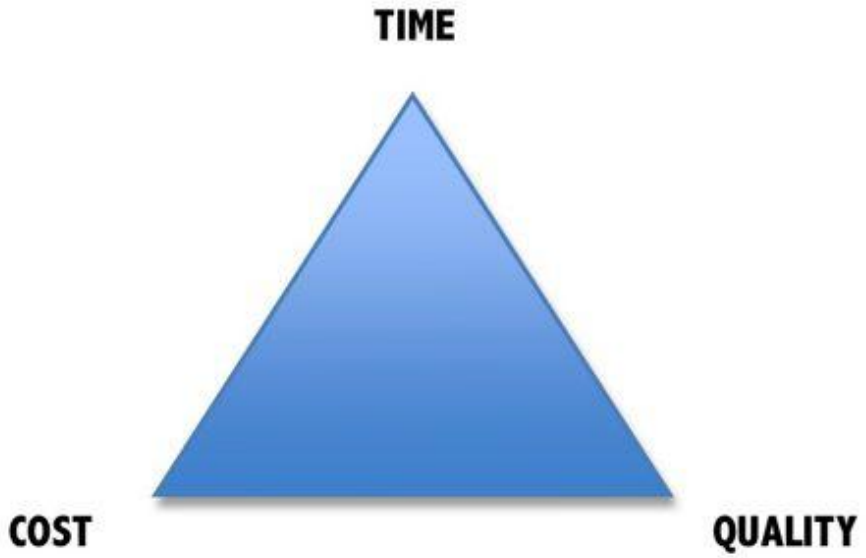
Think about all the devices and systems that you encounter in your everyday life …which have software controlling them

List as many as you can

Virtually all countries depend on complex computer-based .systems

# Why is Software Engineering important?

Complex systems need a disciplined approach for designing, developing and managing them.



# Software Development Crises

**Projects were:**

- Late.
- Over budget.
- Unreliable.
- Difficult to maintain.
- Performed poorly.

# Software errors…..*the cost*

Errors in computer software can have devastating effects.

# Software Crisis

**Example 1: 2009,Computer glitch delays flights**

**Saturday 3rd October 2009-London, England (CNN)**

- Dozens of flights from the UK were delayed Saturday after a glitch in an air traffic control system in Scotland, but the problem was fixed a few hours later.
- The agency said it reverted to backup equipment as engineering worked on the system.
- The problem did not create a safety issue but could cause delays in flights.

# Software Crisis

**Example 2: Ariane 5 Explosion**

- European Space Agency spent 10 years and $7 billion to produce Ariane 5.
- Crash after 36.7 seconds.
- Caused by an overflow error. Trying to store a 64-bit number into a 16-bit space.
- Watch the video: http://www.youtube.com/watch?v=z-r9cYp3tTE

# Software Crisis

**Example 3: 1992, London Ambulance Service**


- Considered the largest ambulance service in the world.
- Overloaded problem.
- It was unable to keep track of the ambulances and their statuses. Sending multiple units to some locations and no units to other locations.
- Generates many exceptions messages.
- 46 deaths.

Therefore…

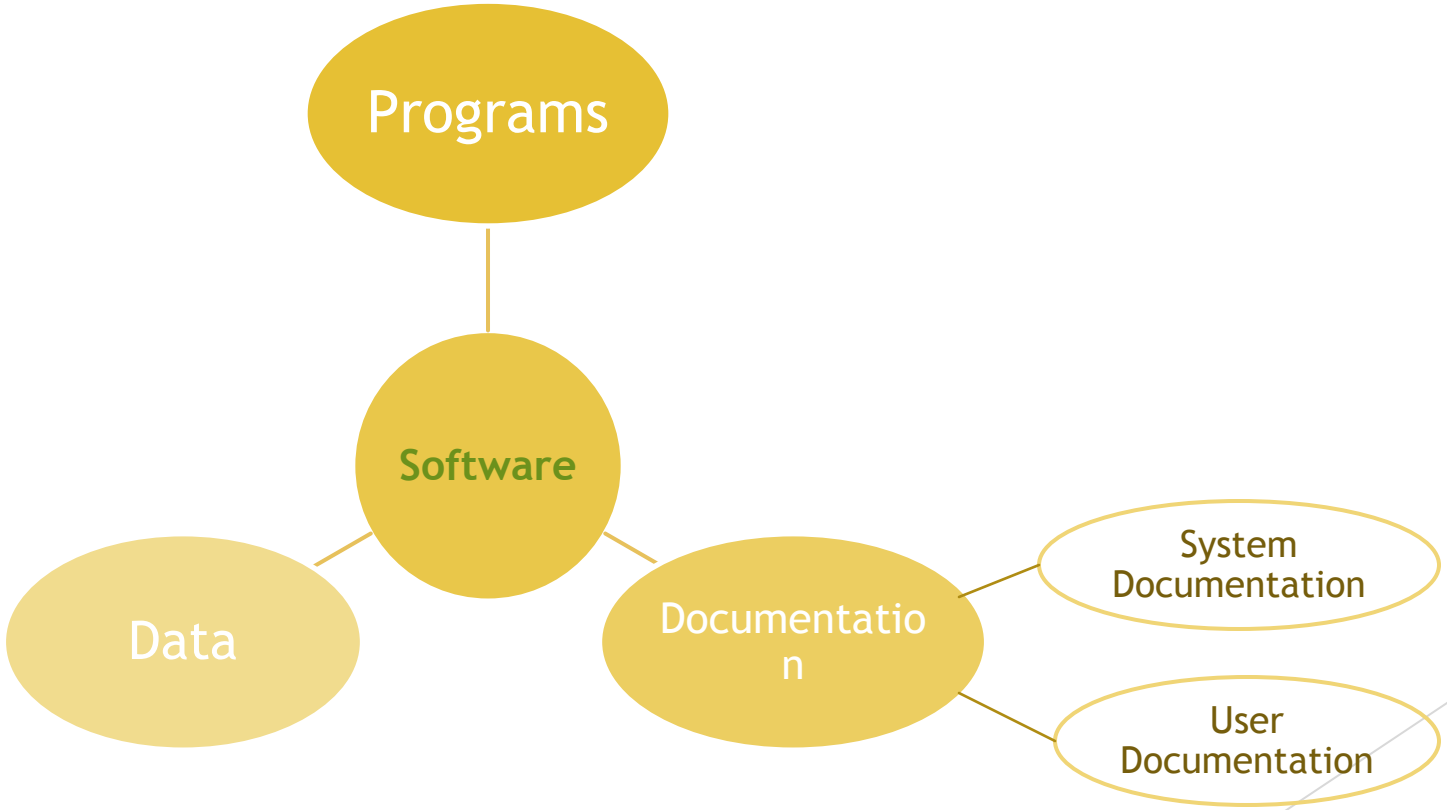
A well-disciplined approach to software development and management is necessary. This is called engineering.

# Software Engineering

- The term *software engineering* first appeared in the **1968** NATO Software Engineering Conference and was meant to provoke thought regarding what was then called the **"software crisis"..**

- ".. An engineering discipline that is concerned **with all aspects of software production** from the **early stages** of system specification to **maintaining the system after it has gone into use.**" *Sommerville, pg.7*

# What is Software?



# Types of Software

- Generic products.
  - Stand-alone systems that are marketed and sold to any customer who wishes to buy them.
  - Examples – PC software such as graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.
  - The specification of what the software should do is owned by the software developer and decisions on software change are made by the developer.
- Customized or bespoke products.
  - Software that is commissioned by a specific customer to meet their own needs.
  - Examples – embedded control systems, air traffic control software, traffic monitoring systems.
  - The specification of what the software should do is owned by the customer for the software and they make decisions on software changes that are required.

# Software Engineering vs. Computer Science

| Computer Science | Software Engineering |
| --- | --- |
| • Theory.<br>• Fundamentals. | • Practicalities of software design, development and delivery. |

# Software Engineering vs. Systems Engineering

**Systems Engineering:**

- Interdisciplinary engineering field (computer, software, and process eng.).
- Focuses on how complex engineering projects should be designed and managed.

| Systems Engineering | Software Engineering |
| --- | --- |
| • All aspects of computer-based systems development: HW + SW + Process.<br>• Older than SWE. | • Deals with the design, development and delivery of SW.<br>• Is part of Systems Engineering. |

# Frequently asked questions about software engineering

| Question | Answer |
| --- | --- |
| What is software? | Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market. |
| What are the attributes of good software? | Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable. |
| What is software engineering? | Software engineering is an engineering discipline that is concerned with all aspects of software production. |
| What are the fundamental software engineering activities? | Software specification, software development, software validation and software evolution. |
| What is the difference between software engineering and computer science? | Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software. |
| What is the difference between software engineering and system engineering? | System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process. |

# Frequently asked questions about software engineering

| Question | Answer |
| --- | --- |
| What are the key challenges facing software engineering? | Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software. |
| What are the costs of software engineering? | Roughly 60% of software costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs. |
| What are the best software engineering techniques and methods? | While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. You can't, therefore, say that one method is better than another. |
| What differences has the web made to software engineering? | The web has led to the availability of software services and the possibility of developing highly distributed service-based systems. Web-based systems development has led to important advances in programming languages and software reuse. |

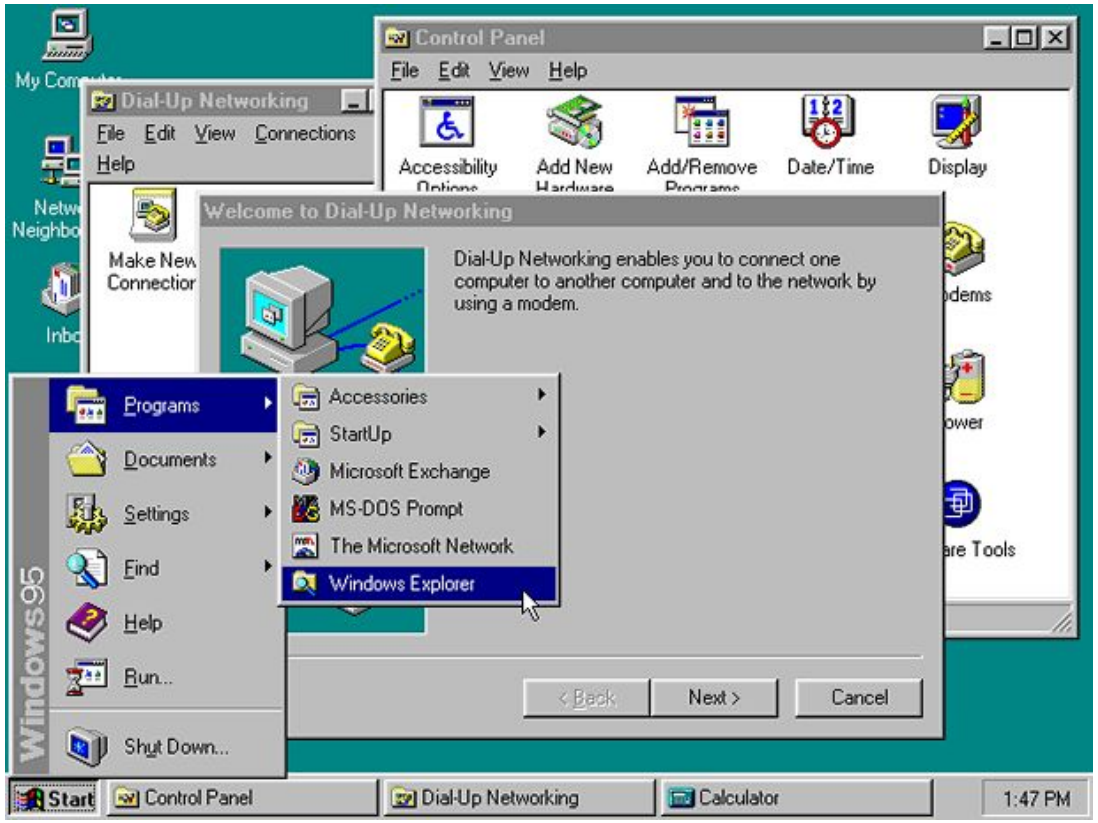
# Formal Definition of Software Engineering:

- Software engineering is the application of principles used in the field of engineering, which usually deals with physical systems, to the design, development, testing, deployment and management of software systems.

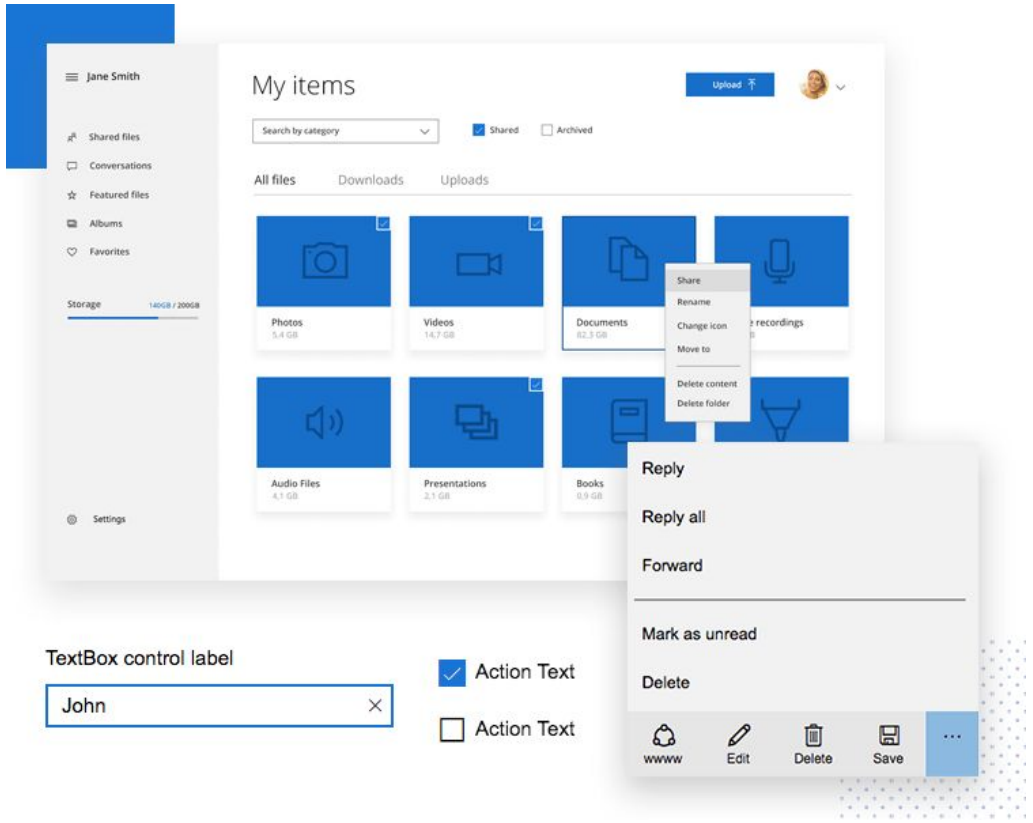
# Software Evolution:

- **Software Evolution** is a term which refers to the process of developing software initially, then timely updating it for various reasons, i.e., to add new features or to remove obsolete functionalities etc. The evolution process includes fundamental activities of change analysis, release planning, system implementation and releasing a system to customers.

# UI in late 90s.



# UI Today:


Jane Smith
My items
Upload
Search by category
Shared
Archived
All files
Downloads
Uploads
Shared files
Conversations
Featured files
Albums
Favorites
Storage
Settings
Photos
Videos
Documents
Audio Files
Presentations
Books
Share
Rename
Change icon
Move to
Delete content
Delete folder
Reply
Reply all
Forward
Mark as unread
Delete
www
Edit
Delete
Save
TextBox control label
John
Action Text
Action Text

# Necessity of Software Evolution:

- **The necessity of Software evolution:** Software evaluation is necessary just because of the following reasons:
- **a) Change in requirement with time:** With the passes of time, the organization's needs and modus Operandi of working could substantially be changed so in this frequently changing time the tools(software) that they are using need to change for maximizing the performance.
- **b) Environment change:** As the working environment changes the things(tools) that enable us to work in that environment also changes proportionally same happens in the software world as the working environment changes then, the organizations need reintroduction of old software with updated features and functionality to adapt the new environment.

- **c) <u>Errors and bugs</u>:** As the age of the deployed software within an organization increases their preciseness or impeccability decrease and the efficiency to bear the increasing complexity workload also continually degrades. So, in that case, it becomes necessary to avoid use of obsolete and aged software. All such obsolete Softwares need to undergo the evolution process in order to become robust as per the workload complexity of the current environment.
- **d) <u>Security risks</u>:** Using outdated software within an organization may lead you to at the verge of various software-based cyberattacks and could expose your confidential data illegally associated with the software that is in use. So, it becomes necessary to avoid such security breaches through regular assessment of the security patches/modules are used within the software. If the software isn't robust enough to bear the current occurring Cyber attacks so it must be changed (updated).
- **e) <u>For having new functionality and features</u>:** In order to increase the performance and fast data processing and other functionalities, an organization need to continuously evolute the software throughout its life cycle so that stakeholders & clients of the product could work efficiently.

# Importance of SE:

**Importance of Software Engineering**

The importance of software engineering lies in the fact that a specific piece of Software is required in almost every industry, every business, and purpose. As time goes on, it becomes more important for the following reasons.

- **1. Reduces Complexity**
- Dealing with big Software is very complicated and challenging. Thus to reduce the complications of projects, software engineering has great solutions. It simplifies complex problems and solves those issues one by one.
- **2. Handling Big Projects**
- Big projects need lots of patience, planning, and management, which you never get from any company. The company will invest its resources; therefore, it should be completed within the deadline. It is only possible if the company uses software engineering to deal with big projects without problems.
- **3. To Minimize Software Costs**
- Software engineers are paid highly as Software needs a lot of hard work and workforce development. These are developed with the help of a large number of codes. But programmers in software engineering project all things and reduce the things which are not needed. As a result of the production of Software, costs become less and more affordable for Software that does not use this method.

- **4. To Decrease Time**
- If things are not made according to the procedures, it becomes a huge loss of time. Accordingly, complex Software must run much code to get definitive running code. So it takes lots of time if not handled properly. And if you follow the prescribed software engineering methods, it will save your precious time by decreasing it.
- **5. Effectiveness**
- Making standards decides the effectiveness of things. Therefore a company always targets the software standard to make it more effective. And Software becomes more effective only with the help of software engineering.
- **6. Reliable Software**
- The Software will be reliable if software engineering, testing, and maintenance are given. As a software developer, you must ensure that the Software is secure and will work for the period or subscription you have agreed upon.

- https://www.knowledgehut.com/blog/web-development/importance-of-software-engineering

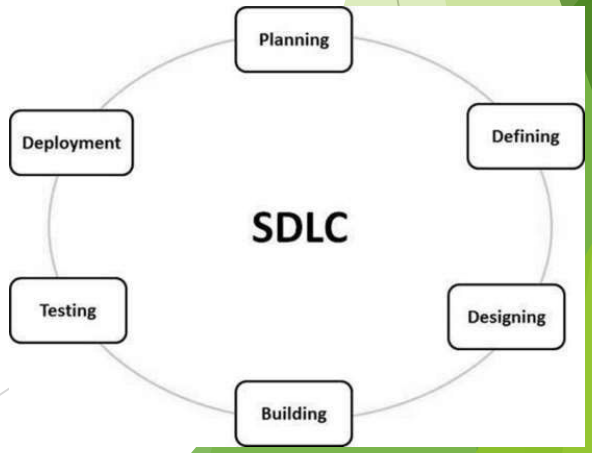
# Software Process:

# SDLC:

- Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.
- SDLC is the acronym of Software Development Life Cycle.
- It is also called as Software Development Process.
- SDLC is a framework defining tasks performed at each step in the software development process.
- ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

# What is SDLC?

- SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software.
- The life cycle defines a methodology for improving the quality of software and the overall development process.
- The following figure is a graphical representation of the various stages of a typical SDLC.



# A typical Software Development Life Cycle consists of the following stages –

- Stage 1: Planning and Requirement Analysis
- Requirement analysis is the most important and fundamental stage in SDLC.
- It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry.
- This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

- Stage 2: Defining Requirements
- Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts.
- This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

- Stage 3: Designing the Product Architecture
- SRS is the reference for product architects to come out with the best architecture for the product to be developed.
- Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.
- This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

- Stage 4: Building or Developing the Product
- In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage.
- The programming language is chosen with respect to the type of software being developed.

- Stage 5: Testing the Product
- This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC.
- However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

- Stage 6: Deployment in the Market and Maintenance
- Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment.
- After the product is released in the market, its maintenance is done for the existing customer base.

# Software Models:

- Software models are ways of expressing a software design.
- Usually some sort of abstract language or pictures are used to express the software design.
- This allows the designer to try different designs and decide which will be best for the final solution.

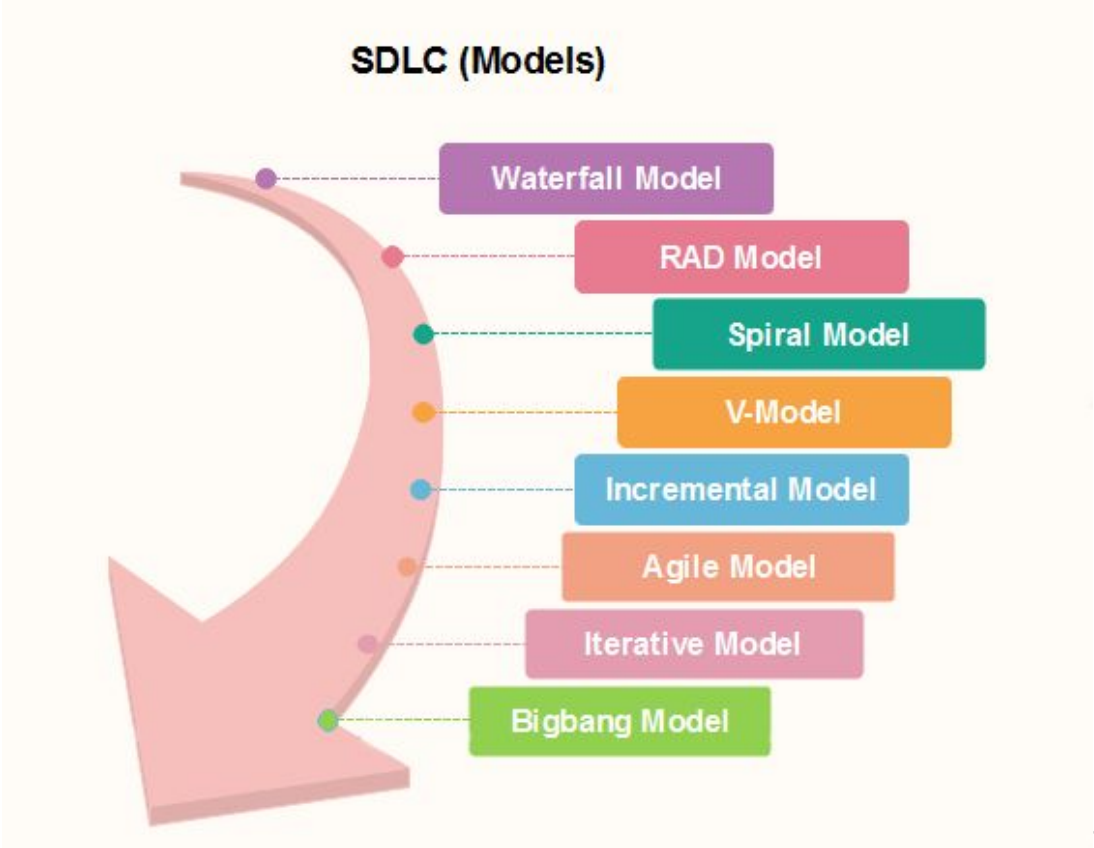
# Example:

- Think of designing your software as you would a house.
- You start by drawing a rough sketch of the floor plan and layout of the rooms and floors.
- The drawing is your modeling language and the resulting blueprint will be a model of your final design.
- You will continue to modify your drawings until you arrive at a design that meets all your requirements.
- Only then should you start cutting boards or writing code.

# Benefit:

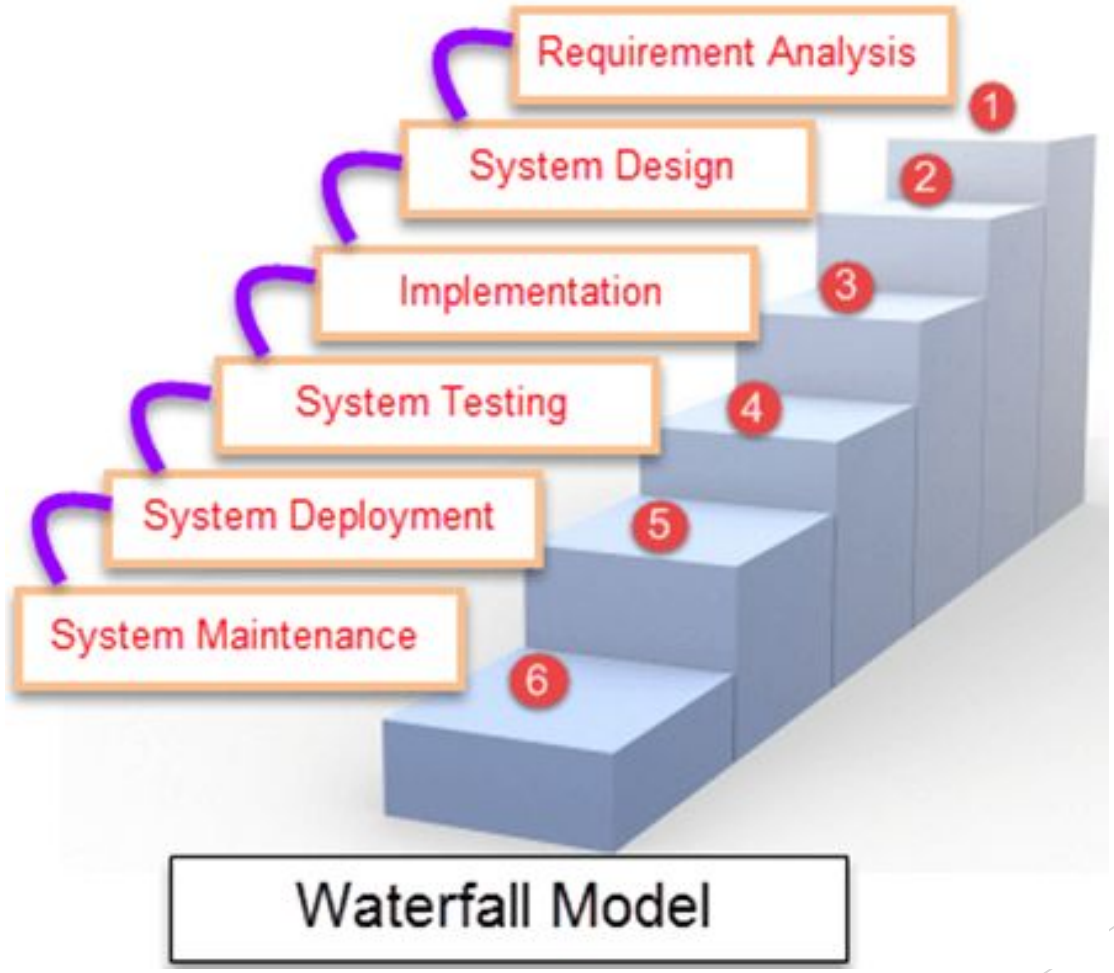
- Again the benefit of designing your software using a modeling language is that you discover problems early and and fix them without refactoring your code.

# Software Development Models:



# Waterfall model(Traditional model)

**This model is also known as the linear sequential model** or **the software life cycle.**



# Explanation:

- It was introduced in 1970 by Winston Royce.
- The classical waterfall model is the basic software development life cycle model.
- It is very simple but idealistic.
- Earlier this model was very popular but nowadays it is not used.
- But it is very important because all the other software development life cycle models are based on the classical waterfall model.

# Explanation:

- **Waterfall Model** is a sequential model that divides software development into pre-defined phases.
- Each phase must be completed before the next phase can begin with no overlap between the phases.
- Each phase is designed for performing specific activity during the SDLC phase.

# Goal/Aim:

- **Feasibility Study:** The main goal of this phase is to determine whether it would be financially and technically feasible to develop the software.
- The feasibility study involves understanding the problem and then determining the various possible strategies to solve the problem.

# Different Phases/Activities

| Different phases | Activities performed in each stage |
| --- | --- |
| **Requirement Gathering stage** | •During this phase, detailed requirements of the software system to be developed are gathered from client |
| **Design Stage** | •Plan the programming language, for Example Java, PHP, .net<br>•or database like Oracle, MySQL, etc.<br>•Or other high-level technical details of the project |
| **Built Stage** | •After design stage, it is built stage, that is nothing but coding the software |

# Different Phases/Activities

| | |
|---|---|
| **Test Stage** | •In this phase, you test the software to verify that it is built as per the specifications given by the client. |
| **Deployment stage** | •Deploy the application in the respective environment |
| **Maintenance stage** | •Once your system is ready to use, you may later require change the code as per customer request |

# When to use SDLC Waterfall Model?

Waterfall Methodology can be used when:

- Requirements are not changing frequently
- Application is not complicated and big
- Project is short
- Requirement is clear
- Environment is stable
- Technology and tools used are not dynamic and is stable
- Resources are available and trained

# Advantages and Disadvantages of Waterfall Model:

- Here are the popular advantages of Waterfall model in Software Engineering with some disadvantages:

| Advantages | Dis-Advantages |
| --- | --- |
| •Before the next phase of development, each phase must be completed | •Error can be fixed only during the phase |
| •Suited for smaller projects where requirements are well defined | •It is not desirable for complex project where requirement changes frequently |
| •They should perform quality assurance test (Verification and Validation) before completing each stage | •Testing period comes quite late in the developmental process |

# Advantages and Disadvantages of Waterfall Model:

| | |
|---|---|
| •Elaborate documentation is done at every phase of the software's development cycle | •Documentation occupies a lot of time of developers and testers |
| •Project is completely dependent on project team with minimum client intervention | •Clients valuable feedback cannot be included with ongoing development phase |
| •Any changes in software is made during the process of the development | •Small changes or errors that arise in the completed software may cause a lot of problems |

# Waterfall model explained Simply:

**Ramon Morales**

Software Developer Since 1991 · Author has **2.3K** answers and **1.8M** answer views · 4y

A customer comes to me and says they need software to do 3 things (those are the "features"). I go back and forth with the customer until I fully understand what is needed. This is the "Requirements" phase.

I develop and test the software. This is the "Development" phase.

I get a user from the customer to use the software to see if it works as expected. If any issues are identified, they are corrected here. We keep doing this phase until the customer is happy. This is the Acceptance" phase.

At this point I am done if this is an external customer. I give the customer the software. If the project is for an internal customer, I will install the software and likely support it going forward.

Different teams do the waterfall slightly differently, but these are the main components.

17.1K views · View upvotes

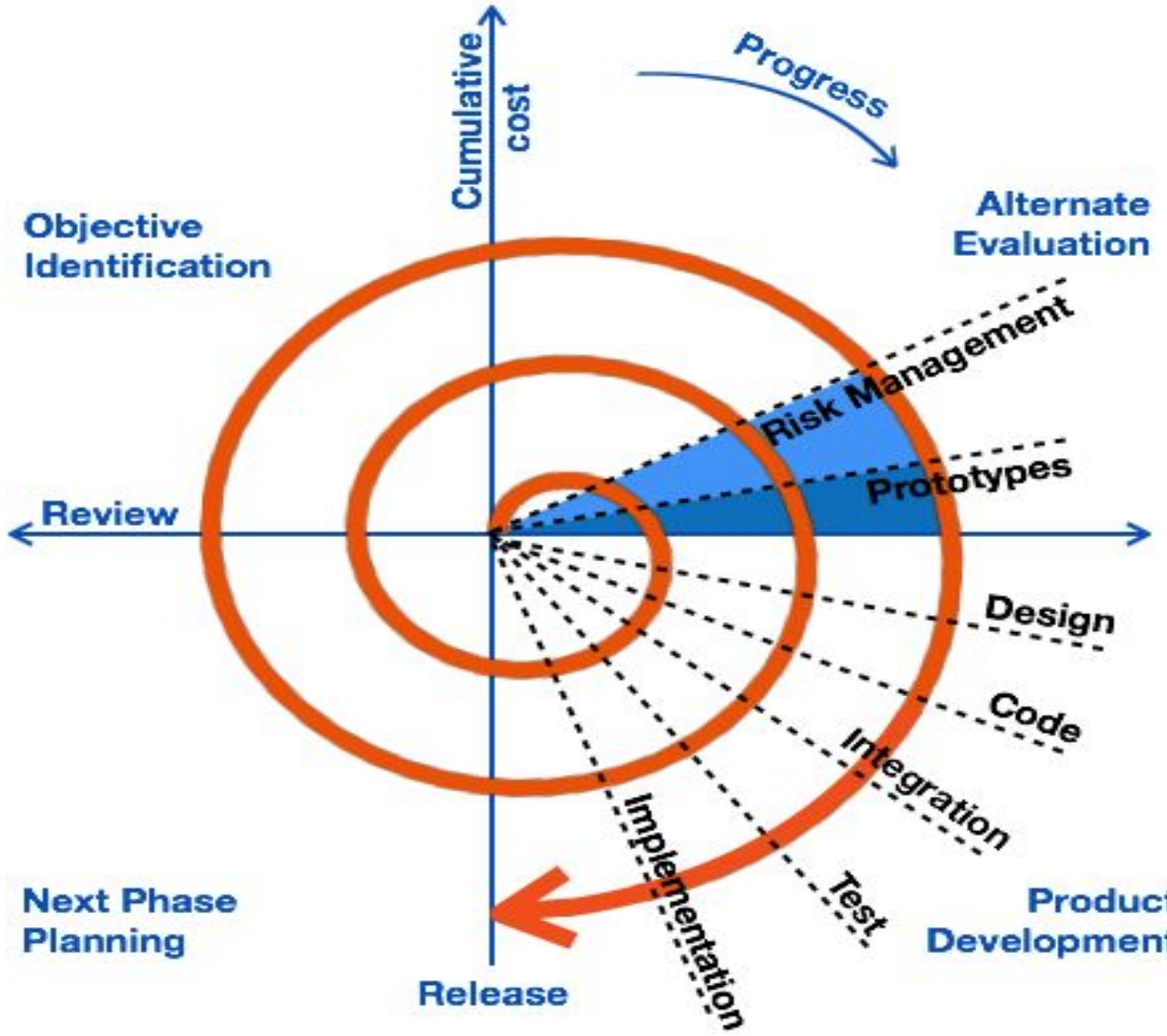
4

# Applications:

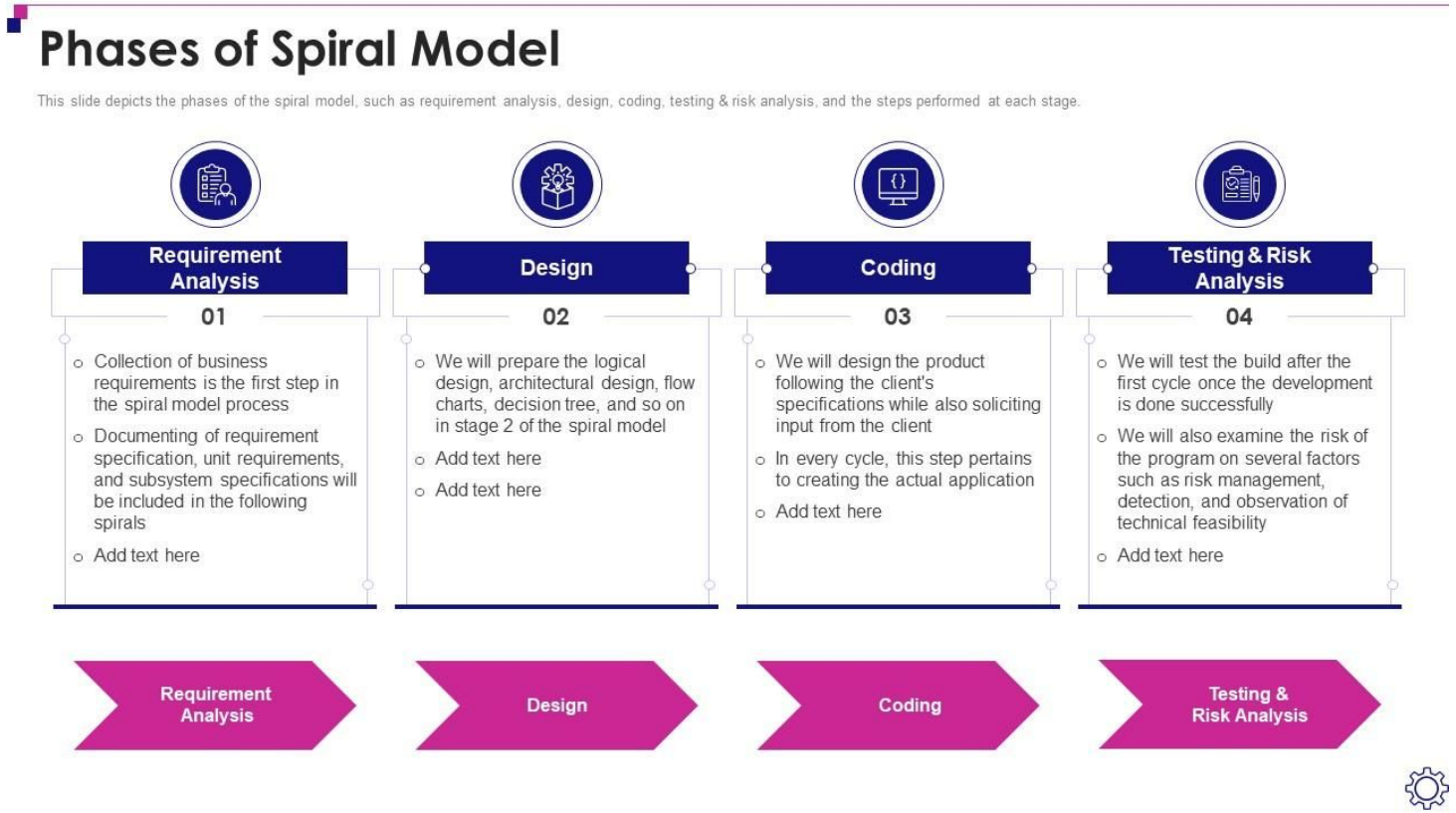
- like payment gateways, stock trading, examination portals.
- The Waterfall model was used to develop enterprise applications like:
- Customer Relationship Management (CRM) systems,
- Human Resource Management Systems (HRMS),
- Supply Chain Management Systems,
- Inventory Management Systems,
- Point of Sales (POS) systems for retail chains, etc.
- Also, for example, the software development arm of Toyota has been traditionally working with waterfall models.
- The waterfall model was also used in **banking, healthcare, control systems for nuclear facilities, space shuttles**, etc.

- These days most projects need to be performed by means of more flexible models (i.e., Agile-based Scrum and Kanban and Lean methodology).
- The Waterfall model is **mostly used on smaller projects where the requirements are clear and there's no need to change them quickly**.
- It's a well-structured approach.
- The stages are well-defined and easy to understand for all.

# Spiral model:



# Phases:


Phases of Spiral Model
This slide depicts the phases of the spiral model, such as requirement analysis, design, coding, testing & risk analysis, and the steps performed at each stage.
Requirement Analysis
01
Collection of business requirements is the first step in the spiral model process
Documenting of requirement specification, unit requirements, and subsystem specifications will be included in the following spirals
Add text here
Design
02
We will prepare the logical design, architectural design, flow charts, decision tree, and so on in stage 2 of the spiral model
Add text here
Add text here
Coding
03
We will design the product following the client's specifications while also soliciting input from the client
In every cycle, this step pertains to creating the actual application
Add text here
Testing & Risk Analysis
04
We will test the build after the first cycle once the development is done successfully
We will also examine the risk of the program on several factors such as risk management, detection, and observation of technical feasibility
Add text here
Requirement Analysis
Design
Coding
Testing & Risk Analysis

# Spiral Model - Design

- The spiral model has four phases.
- A software project repeatedly passes through these phases in iterations called Spirals.

# Phase 1:

- **Identification:**
- This phase starts with gathering the business requirements in the baseline spiral.
- In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase.

- This phase also includes understanding the system requirements by continuous communication between the customer and the system analyst.
- At the end of the spiral, the product is deployed in the identified market.

# Phase 2:

- **Design:**
- The Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and the final design in the subsequent spirals.

# Phase 3:

- **Construct or Build:**
- The Construct phase refers to production of the actual software product at every spiral.
- In the baseline spiral, when the product is just thought of and the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback.
- Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number.
- These builds are sent to the customer for feedback.

# Phase 4:

- **Evaluation and Risk Analysis:**
- Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks,
- such as schedule slippage and cost overrun.
- After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

# Activity:

- Compare between Waterfall model and spiral model.
- Real time Applications of Spiral model.