

<b>Software Engineering</b>	<b>6</b>
<b>Well Engineered Software</b>	<b>6</b>
<b>Software Engineering Key Challenges</b>	<b>7</b>
<b>Reason for software project failure</b>	<b>7</b>
<b>Software Project Milestone</b>	<b>8</b>
1. Requirements gathering	8
2. Validate your requirement expectations	8
3. Predevelopment planning	9
4. Implementation	9
5. Quality assurance testing	9
6. User acceptance testing	9
7. Deployment	10
8. Support	10
<b>Software Project Deliverables</b>	<b>10</b>
<b>Why Software Engineering</b>	<b>10</b>
<b>Factors Contributing to the Software Crisis</b>	<b>10</b>
<b>SDLC</b>	<b>10</b>
• Understanding Software Development Life Cycle (SDLC)	10
• Overview of SDLC Phases	10
■ Feasibility analysis	11
■ Requirement analysis and specification	11
■ Design	11
■ Coding	11
■ Testing	11
■ Maintenance	11
• Development Process	11
■ Requirements analysis,	11
■ Design	11
■ Coding	11
■ Testing	11
■ Delivery	11
1. Requirement Analysis	11
2. Design	11
3. Coding	11
4. Testing & Quality Assurance	12
Typical Effort Distribution	12
Distribution of effort	12
5. Delivery	12
When are defects introduced?	12
<b>SDLC Methodologies (Software Development Approaches)</b>	<b>13</b>
1. Waterfall approach - the oldest and widely used	13
Advantages	14
Disadvantages	14
	1

Usage	14
Summary	14
2. Prototyping Approach	14
Development of Prototype	15
Prototyping	15
Advantages	15
Disadvantages:	16
Applicability:	16
Variants	16
Summary	16
3. Iterative(Incremental approach) - used widely in product dev	16
Iterative Enhancement	17
Iterative Development	17
Benefits	17
Drawbacks	17
Applicability	17
Execution	17
Summary	18
4. Spiral Model	18
○ Planning	18
○ Risk analysis	18
○ Development	18
○ Assessment	18
When to use Spiral model	19
• Win-win spiral approach	19
• RAD	20
Phase 1: Requirements planning	20
Phase 2: User design	20
Phase 3: Rapid construction	21
Phase 4: Cutover	21
Benefits of RAD methodology	21
Advantages & Disadvantages	22
• Agile models	22
Agile software development values	22
Agile software development principles	22
Agility	22
Advantages and disadvantages	23
Agile methodologies	24
Ques: Which of the following approaches includes the iterative nature of the prototyping approach and the linear nature of waterfall approach?	24
<b>Software Requirement Specification</b>	<b>26</b>
<b>Characteristics of good SRS</b>	<b>26</b>
<b>Properties of a good SRS document</b>	<b>28</b>
<b>Problem Analysis</b>	<b>29</b>
<b>Feasibility: study of requirements</b>	<b>29</b>
• Operational or organizational feasibility	29
	2

• Technical feasibility	29
• Economic feasibility	29
<b>Types of Feasibility Study :</b>	<b>29</b>
1. Technical Feasibility –	29
2. Operational Feasibility –	29
3. Economic Feasibility –	30
4. Legal Feasibility –	30
5. Schedule Feasibility –	30
<b>Feasibility Study Process :</b>	<b>30</b>
<b>Need of Feasibility Study :</b>	<b>30</b>
<b>Testing</b>	<b>31</b>
• Different types and principles of software testing	31
• Levels of testing	31
1. Unit Testing	31
2. Integration Testing	32
○ Big Bang Integration testing	32
○ Top Down Integration testing	32
○ Bottom Up Integration testing	32
○ Mixed Integration testing	32
3. System Testing	33
○ Alpha Testing	33
○ Beta Testing	33
○ Acceptance Testing	33
○ Performance Testing	33
• Alpha testing	33
• Beta Testing	33
• User acceptance test(Acceptance Testing)	34
• Performance Testing	34
■ Stress Testing	34
■ Volume Testing	34
■ Configuration Testing	34
■ Compatibility Testing	34
■ Regression Testing	34
■ Recovery Testing	34
■ Maintenance Testing	34
■ Documentation Testing	34
■ Usability Testing	34
Discussion	34
• Regression testing	34
• Validation Testing	35
• Functional testing	35
• Structural testing	35
• Test documentation	35
• Metrics	35
• Test plan	36
• Test cases specifications	36

• Verification vs Validation	36
• Reliability Assessment	36
• Test case design	36
• Importance of unit test	36
• System Integration test	36
•	36
• Process Models	36
○ A generic testing process	36
○ V model	37
• Software Maintenance	37
<b>Software Testing</b>	<b>37</b>
Introduction	37
Objectives	38
Error, Bug, Fault & Failure	38
Testing Life Cycle	38
Testing Methodologies	38
Black box testing	38
White box testing	38
<b>Key topics:</b>	<b>39</b>
• Object Oriented design concept	39
• Overview of Object-Oriented Concepts	39
Ques: Which phase of SDLC includes translation of the requirements specified in the SRS into a logical structure that can be implemented in a programming language?	41
• Role of Object-Oriented Analysis and Design (OOAD) in SDLC	41
• <b>Software project management - in notes</b>	<b>41</b>
Project Scheduling	41
Process :	41
Problems arise during Project Development Stage :	42
Resources required for Development of Project :	42
Advantages of Project Scheduling :	42
<b>Project Staffing</b>	<b>43</b>
Staffing Process	43
Manpower Planning:	43
Recruitment:	43
Selection:	44
Placement and Orientation:	44
Training and Development:	44
Performance appraisal:	44
Promotion and Career planning:	44
Compensation:	45
Aspects or Components of Staffing	45
Recruitment:	45
Selection:	45
Training:	45
<b>Risk Management</b>	<b>46</b>

Uncertainty	46
loss –	46
1. Project Risks:	46
2. Technical Risks:	46
3. Business Risks:	46
<b>Principle of Risk Management</b>	<b>47</b>
Quality Management	47
<b>Activities of Software Quality Management:</b>	<b>47</b>
Quality Assurance	47
Quality Planning	47
Quality Control -	47
<b>Software Quality Assurance</b>	<b>48</b>
Software Quality Assurance has:	48
Major Software Quality Assurance Activities:	48
1. SQA Management Plan:	48
2. Set The Check Points:	48
3. Multi testing Strategy:	48
4. Measure Change Impact:	48
5. Manage Good Relations:	48
Benefits of Software Quality Assurance (SQA):	49
Disadvantage of SQA:	49
<b>Software Configuration Management</b>	<b>50</b>
Processes involved in SCM –	50
1. Identification and Establishment –	50
2. Version control –	50
3. Change control –	50
4. Configuration auditing –	50
5. Reporting –	51
SCM Tools –	51
<b>SCM</b>	<b>51</b>
Why do we need Configuration Management?	51
Importance of SCM	51
<b>Coding</b>	<b>52</b>
Coding Standards and Guidelines	52
Code Review	53
<b>Software Documentation</b>	<b>54</b>
<b>New tools and techniques of SE process management, version control, lifecycle management</b>	<b>55</b>
<b>Software development is constantly evolving</b>	<b>55</b>
Low-code & no-code software development	55
Big data security	55
DevSecOps	56
Increasing reliance on artificial intelligence	57
Augmented reality, virtual reality & mixed reality	57

Progressive web apps	58
Growth in IoT	59
Increased emphasis on UI/UX	59
Serverless computing	60
Blockchain technology	60
Multi-cloud architectures	61
Rise of modern programming languages	62
Cloud-native application development	62
Trends in software development & AppDev	62

## Software Engineering

- Computer programs and associated documentation
- Software=Program+Documentation+Operating Procedures(**Components of software**)
- Software engineering is an engineering discipline which is concerned with all aspects of software production
- Software engineers should
  - adopt a systematic and organised approach to their work
  - use appropriate tools and techniques depending on
    - the problem to be solved,
    - the development constraints and
  - use the resources available

At the first conference on software engineering in 1968, Fritz Bauer defined software engineering as “The establishment and use of sound engineering principles in order to obtain economically developed software that is reliable and works efficiently on real machines”.

Stephen Schach defined the same as “A discipline whose aim is the production of quality software, software that is delivered on time, within budget, and that satisfies its requirements”.

Both the definitions are popular and acceptable to majority. However, due to increase in cost of maintaining software, objective is now shifting to produce quality software that is maintainable, delivered on time, within budget, and also satisfies its requirements.

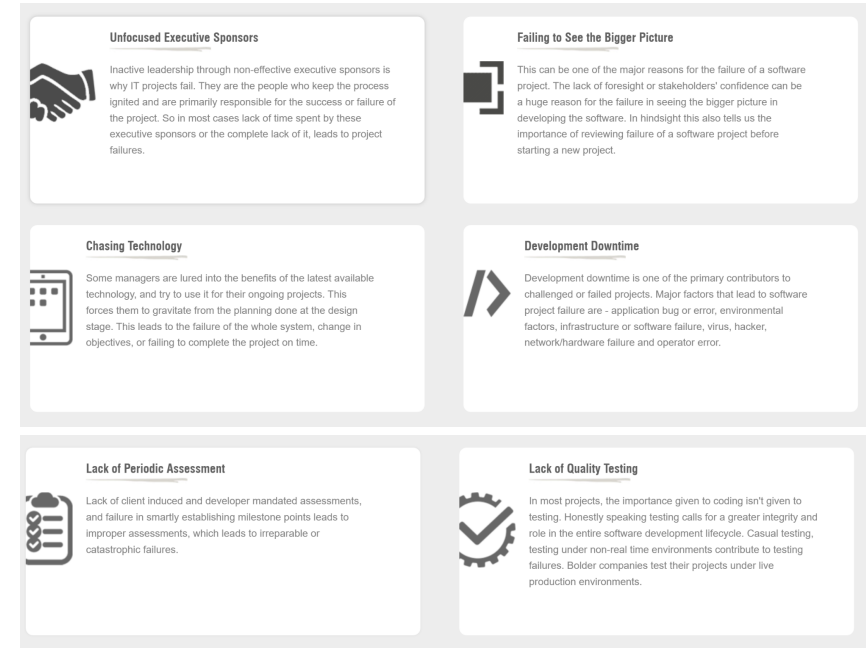
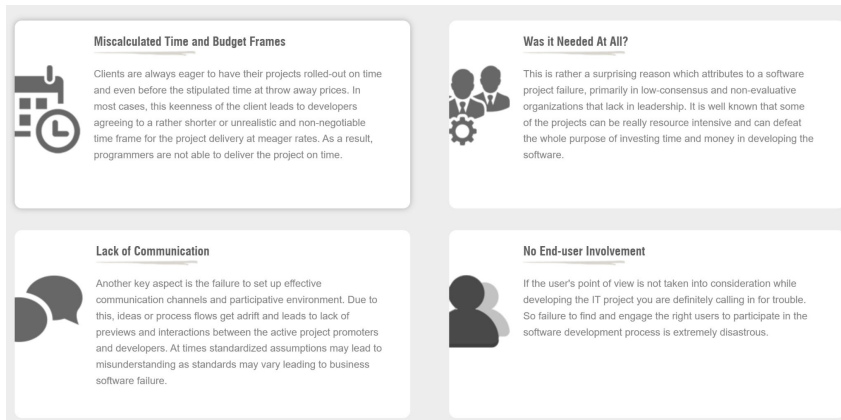
- 
- 

## Well Engineered Software

## Software Engineering Key Challenges

- Shifting Project Requirements
- Uneven Workload Distribution
- Transitioning to DevOps and DevSecOps
- Maintaining Quality
- Integrating With Other Applications and Systems
- Planning Accurately
- Data Privacy
- Rising Competition
- Talent Shortages
- How to Overcome Software Engineering Challenges

## Reason for software project failure



## Software Project Milestone

### 1. Requirements gathering

Before moving forward on any project, be sure your business understands the objectives and scope of work first at a high level and then broken down into as granular as you can. Business requirements can be presented in many forms from listed on a word document to being written up in user stories for teams that follow an agile approach. What you ultimately want to understand is a full representation of the things you need the software to do.

### 2. Validate your requirement expectations

Next, you want to be sure to engage your technical resources so they can translate the requirements into more specifics as to how they are going to fulfill them. For example, if your general requirement is that you want to build an e-commerce store, the end result may not be what you want. In other words, the requirement is too generic. Instead, you should ask your developer or technical resource how they plan to build the e-store and what tools or platform they will use, such as WooCommerce in WordPress, Shopify or Magento, etc. When you understand the tools and platform that will be used, it can help you plan ahead. If you leave your project at the basic requirements level, you've left too much room for interpretation, which could impact the project down the line. Before completing this step, get a qualified budget and timeline estimate, which can help you decide if you're aimed at a project that is doable before getting too far in.

### 3. Predevelopment planning

There are three teams involved in ironing out project details. The first is the Architectural and Technical Design team. Depending on your project's requirements, here are some of the questions you might ask your team to ensure proper planning:

- Where will the application be hosted?
- Are we using any existing platforms or products?
- Do we have product licenses we need to buy, and who will own the rights to it?
- Are we building in a language that is supported in our area?
- How will you model your data?

The second team is Project Planning. Think about determining which project management model will be best for your team. In an Agile model, you don't have to flesh out a full project plan up front; instead, you agree you will have a deliverable every two or three weeks. In the Agile approach, you have a planning session at the beginning of every iteration. Whereas in the Waterfall method, you work out a master project plan encompassing the entire scope of work and lay it out in a Gant Chart from start to completion. Depending on the approach you choose, your project team would then set up the respective teams, tasks, deliverables, and load any user stories you may have.

The third team is Quality Assurance. At this point in the project, QA will start writing out their test cases based on the fleshed-out requirements, begin test scripts, and get their work set up so they can start testing when development begins.

The predevelopment stage is also a great opportunity to have your technical representative validate the technical direction of your project. Because there are so many approaches one can take for the same business requirements, it's a good idea to ask all the specifics upfront, so it doesn't snag or halt the project later.

### 4. Implementation

In this milestone, your teams will begin working on all the phases you laid out in the preplanning stages. Once you get into development, implementation will be broken down into a combination of many iteration deliverables, which will end up being more than one milestone. We recommend setting up deliverables that can be reviewed every two or three weeks.

### 5. Quality assurance testing

Quality assurance can be done in two ways. One at the end of development, which is not recommended except for small projects, or it can be done throughout the project. If done throughout, QA should be done with each one of the two or three-week deliverable iterations. Like implementation, QA is not one milestone, but it will be many milestones as the project progresses. It's not uncommon to also do one QA pass at the end of the project to make sure there's end to end testing validating the whole system is working as expected.

### 6. User acceptance testing

User acceptance testing is a critical validation phase. It gives anyone who is not a part of the development team a chance to validate that all the requirements are fulfilled. In an iterative delivery model, the UA team would have seen each one of the deliverables throughout the project. So, this phase would be more of a final look for approval before it gets pushed out into production.

### 7. Deployment

Deployment is the exercise of making the application live. In this phase, there's an environment set up, where you upload the code, do a quick test to make sure everything is running in the way it was in your test environment, and a push to production. Some other exercises for turning a site live include migrating an existing URL or a site email link to point to the new server. Depending on the project requirements, the exercises could vary.

### 8. Support

Too many project plans end with go live as the last line item. After the project is complete, your development team will go away if you don't have a support plan in place. So, sometime before post-launch, you should think about how you plan to support your new application. Will you add-on a support option with your current vendor? Will you train someone on your team? Or do you need to hire someone long-term? These are all considerations to think about.

## Software Project Deliverables

## Why Software Engineering

- Change in nature & complexity of software
- Concept of one "guru" is over
- We all want improvement
  - Ready for change

## Factors Contributing to the Software Crisis

- Larger problems,
- Lack of adequate training in software engineering,
- Increasing skill shortage,
- Low productivity improvements.

## SDLC

- Understanding Software Development Life Cycle (SDLC)
  - SDLC is a disciplined and systematic approach that divides the software development process into various phases, such as requirements, design, and coding.
  - The phase-wise software development process helps you track schedule, cost, and quality of the software projects.
- Overview of SDLC Phases
  - There are six phases of SDLC:

- Feasibility analysis
- Requirement analysis and specification
- Design
- Coding
- Testing
- Maintenance

## ● Development Process

- A set of phases and each phase being a sequence of steps
- For each phase there are
  - A variety of methodologies
  - Corresponding to different sequence of steps for a phase
- Why have phases?
  - To employ divide and conquer
  - Each phase handles a different part of the problem
  - Helps in continuous validation
- Commonly has these activities:
  - Requirements analysis,
  - Design
  - Coding
  - Testing
  - Delivery
- Different models perform them in different manner!

## 1. Requirement Analysis

- State the problem precisely!
- Forms the basis of agreement between user and developer
- Specifies "what" not "how"
- Hard task - needs often not understood well
- Requirement specifications of even medium systems can be many hundreds of pages
- Output is the Software Requirements Specification (SRS) document

## 2. Design

- A major step in moving from problem domain to solution domain
- Three main tasks
  1. Architecture design – components and connectors that should be there in the system
  2. High level design – modules and data structures needed to implement the architecture
  3. Detailed design – logic of modules
- Most methodologies focus on architecture or high level design
- Outputs are arch/des/logic design documents

## 3. Coding

- Converts design into code in specific language
- Goal: Implement the design with simple and easy to understand code
- Coding phase affects both testing and maintenance
  - Well written code reduces testing and maintenance effort

- Output is code

## 4. Testing & Quality Assurance

- Defects are introduced in each phase
- Must be found and removed to achieve high quality
- Goal: Identify most of defects
- Very expensive task; must be properly planned and executed
- Outputs are
  - Test plans/results, and
  - the final tested (hopefully reliable) code

## Typical Effort Distribution

- Distribution of effort :
  1. Req. - 10-20%
  2. Design - 10-20%
  3. Coding - 20-30%
  4. Testing - 30-50%
- Coding is not the most expensive!

## Distribution of effort

- How programmers spend their time
  1. Writing programs - 13%
  2. Reading programs and manuals - 16%
  3. Job communication - 32%
  4. Others - 39%
- Programmers spend more time in reading programs than in writing them.
- Writing programs is a small part of their lives.

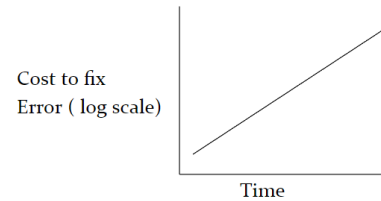
## 5. Delivery

- What the "Operations" group does.
- Varies by distribution model
  - Shrink Wrapped Software
  - In house software
  - Web-based
  - Software As A Service (SaaS)
  - ...
- From a users perspective may be as important as design!

## When are defects introduced?

- Distribution of error occurrences by phase is
  1. Req. - 20%
  2. Design - 30%
  3. Coding - 50%

- Defects can be injected at any of the major phases.
- Cost of latency: Cost of defect removal increases exponentially with latency time.



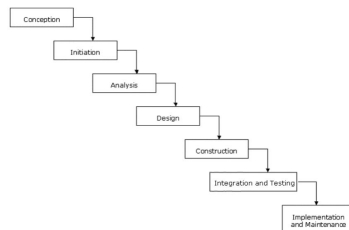
- Cheapest way to detect and remove defects close to where it is injected.
- Hence must check for defects after every phase.

## SDLC Methodologies (Software Development Approaches)

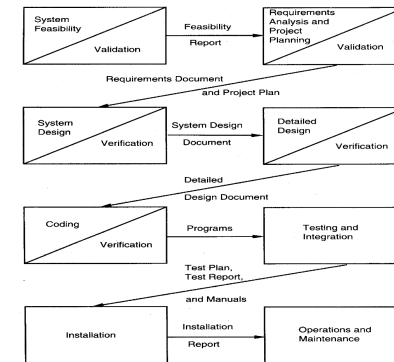
- Different types of projects have different requirements.
- It is required to tailor the SDLC phases to meet the specific needs of the project.
- The tailoring the SDLC phases, gives rise to various software development approaches:

### 1. Waterfall approach - the oldest and widely used

- Describes the software development process in a linear sequential flow.
- Is the earliest approach used for software development.
- Linear sequence of stages/phases
- Requirements -> HLD(High Level Design) -> DD(Design Development) -> Code -> Test -> Deploy
- A phase starts only when the previous has completed; no feedback!
- The phases partition the project, each addressing a separate concern
- Defines the software development process in seven phases:
  - Conception
  - Initiation
  - Analysis
  - Design
  - Construction
  - Integration and testing
  - Implementation and maintenance



•



- Linear ordering implies each phase should have some output
- The output must be validated/certified
- Outputs of earlier phases: work products
- Common outputs of a waterfall: SRS, project plan, design docs, test plan and reports, final code, supporting docs

### Advantages

- Natural approach for problem solving
- Conceptually simple, cleanly divides the problem into distinct independent phases
- Easy to administer in a contractual setup – each phase is a milestone

### Disadvantages

- Assumes that requirements can be specified and frozen early
- May fix hardware and other technologies too early
- Follows the "big bang" approach – all or nothing delivery; too risky
- Very document oriented, requiring docs at the end of each phase

### Usage

- Well suited for projects where requirements can be understood easily and technology decisions are easy
- Has been used widely
- For standard/familiar type of projects it still may be the most optimum
- Well suited to the out sourcing model

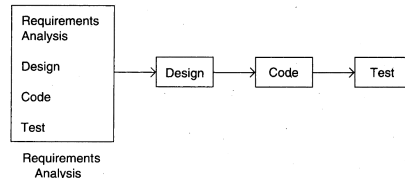
### Summary

Strength	Weakness	Types of Projects
Simple Easy to execute Intuitive and logical	All or nothing – too risky Req frozen early May chose outdated hardware/tech	Well understood problems, short duration projects, automation of existing manual systems

Easy contractually	Disallows changes No feedback from users Encourages req bloating	
--------------------	--	--

## 2. Prototyping Approach

- Also known as evolutionary approach.
- Is a sample implementation of the system that shows limited and main functional capabilities of the proposed system.
- Is used in the requirements gathering and analysis phase to capture the exact requirement of the proposed system.
- Various types of prototypes:
  - Throwaway prototypes
  - Evolutionary prototypes
- Limitations of prototyping approach:
  - Gives clients the false impression that a few minor changes to the prototype will give them the required system.
  - May compromise on the overall quality of the software in the rush to develop the prototype.
- 
- 
- Addresses the requirement specification limitation of waterfall
- Instead of freezing requirements only by discussions, a prototype is built to understand the requirements
- Helps alleviate the requirements risk
- A small waterfall model replaces the requirements stage



### Development of Prototype

- Starts with initial requirements
- Only key features which need better understanding are included in prototype
- No point in including those features that are well understood
- Feedback from users taken to improve the understanding of the requirements

### Prototyping

- Cost can be kept low
- Build only features needing clarification
- "quick and dirty" – quality not important, scripting etc can be used
- Things like exception handling, recovery, standards are omitted
- Cost can be a few % of the total
- Learning in prototype building will help in building, besides improved requirements

### Advantages

- Requirement will be more stable and more likely to satisfy user needs
- Early opportunity to explore scale/performance issues
- Ability to modify or cancel the project early
- Enhanced user engagement

### Disadvantages:

- Potential hit on cost and schedule
- Potential false sense of security if prototype does not focus on key (high risk) issues

### Applicability:

- When req are hard to elicit
- When confidence in reqs is low
- Where reqs are not well understood
- When design is driven by user needs

### Variants

- Paper Prototypes
- UI Prototypes
- Technology Proving
- Rapid Prototyping environments

### Summary

Strength	Weakness	Types of Projects
Helps req elicitation Reduces risk Better and more stable final system	Front heavy Possibly higher cost and schedule Encourages req bloating Disallows later change	Systems with novice users; or areas with req uncertainty. Heavy reporting based systems can benefit from UI proto

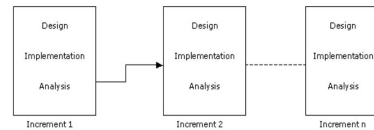
## 3. Iterative(Incremental approach) - used widely in product dev

- In an incremental approach, software requirements are broken down into various functional units.
- Each functional unit is implemented in an increment and the final product is achieved after all the functional units are implemented in the development process.
- Counters the "all or nothing" drawback of the waterfall model
- Combines benefit of prototyping and waterfall
- Develop and deliver software in increments
- Each increment is complete in itself
- Can be viewed as a sequence of waterfalls
- Feedback from one iteration is used in the future iterations
- Each increment in the incremental approach includes three phases:
  - Design
  - Implementation

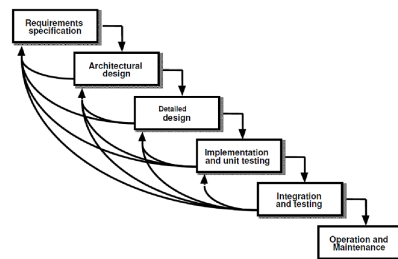


- Analysis

### Iterative Enhancement



### Iterative Development



- 
- Most Software Products follow it
- Used commonly in customized development also
- Businesses want quick response for sw
- Cannot afford the risk of all-or-nothing
- Newer approaches like XP, Agile,... all rely on iterative development

### Benefits

- Get-as-you-pay
- feedback for improvement

### Drawbacks

- Architecture/design may not be optimal
- Amount of refactoring may increase
- Total cost may increase

### Applicability

- where response time is important,
- risk of long projects cannot be taken,
- all req not known

### Execution

- Each iteration is a mini waterfall – decide the specs, then plan the iteration

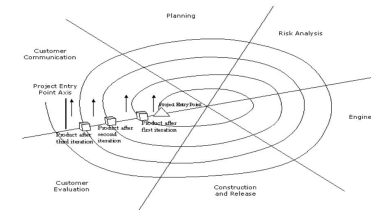
- Length of iteration driven by amount of new functionality to be added in an iteration

### Summary

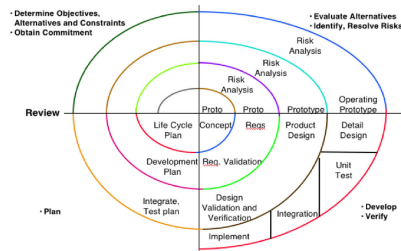
Strength	Weakness	Types of Projects
Regular deliveries, leading to biz benefit Can accommodate changes naturally Allows user feedback Avoids req bloating Naturally prioritizes req Allows reasonable exit points Reduces risks	Overhead of planning each iteration Total cost may increase System arch and design may suffer Rework may increase	For businesses where time is imp; risk of long projects cannot be taken; req not known and evolve with time

### 4. Spiral Model

- Includes the iterative nature of the prototyping approach and the linear nature of the waterfall approach.
- Is ideal for developing software that is released in various versions.
- The radial dimension of the model represents the cumulative costs. Each path around the spiral is indicative of increased costs. The angular dimension represents the progress made in completing each cycle. Each loop of the spiral from X-axis clockwise through 360 degrees represents one phase. One phase is split into four sectors of major activities:
  - Planning
    - Determination of objectives, alternatives and constraints.
  - Risk analysis
    - Analyze alternatives and attempts to identify and resolve the risks involved.
  - Development
    - Product development and testing product
  - Assessment
    - Customer evaluation
- The six phases of spiral approach are:
  - Customer communication
  - Planning
  - Risk analysis
  - Engineering
  - Construction and release
  - Customer evaluation



-



### When to use Spiral model

- When costs there is a budget constraint and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment because of potential changes to economic priorities as the
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.

### • Win-win spiral approach

- Is an extension of the spiral approach.
- The phases in this approach are the same as the phases in the spiral approach.
- In this approach, the development team and the customer hold discussions and negotiate on the requirements that need to be included in the current iteration.
- The approach is called win-win because it is a winning situation for the development team and also for the customer.
- The win-win spiral approach is generally used when you have time-bound releases.

### • RAD

Rapid application development (RAD) is an agile project management strategy popular in software development.

The key benefit of a RAD approach is fast project turnaround, making it an attractive choice for developers working in a fast-paced environment like software development. This rapid pace is made possible by RAD's focus on minimizing the planning stage and maximizing prototype development.

By reducing planning time and emphasizing prototype iterations, RAD allows project managers and stakeholders to accurately measure progress and communicate in real time on evolving issues or changes. This results in greater efficiency, faster development, and effective communication.

You can break down the process in a few ways, but in general, RAD follows four main phases.

#### Phase 1: Requirements planning

This phase is equivalent to a project scoping meeting. Although the planning phase is condensed compared to other project management methodologies, this is a critical step for the ultimate success of the project.

During this stage, developers, clients (software users), and team members communicate to determine the goals and expectations for the project as well as current and potential issues that would need to be addressed during the build.

A basic breakdown of this stage involves:

Researching the current problem

Defining the requirements for the project

Finalizing the requirements with each stakeholder's approval

It is important that everyone has the opportunity to evaluate the goals and expectations for the project and weigh in. By getting approval from each key stakeholder and developer, teams can avoid miscommunications and costly change orders down the road.

#### Phase 2: User design

Once the project is scoped out, it's time to jump right into development, building out the user design through various prototype iterations.

This is the meat and potatoes of the RAD methodology—and what sets it apart from other project management strategies. During this phase, clients work hand in hand with developers to ensure their needs are being met at every step in the design process. It's almost like customizable software development

where the users can test each prototype of the product, at each stage, to ensure it meets their expectations.

All the bugs and kinks are worked out in an iterative process. The developer designs a prototype, the client (user) tests it, and then they come together to communicate on what worked and what didn't.

This method gives developers the opportunity to tweak the model as they go until they reach a satisfactory design.

Both the software developers and the clients learn from the experience to make sure there is no potential for something to slip through the cracks.

### Phase 3: Rapid construction

Phase 3 takes the prototypes and beta systems from the design phase and converts them into the working model.

Because the majority of the problems and changes were addressed during the thorough iterative design phase, developers can construct the final working model more quickly than they could by following a traditional project management approach.

The phase breaks down into several smaller steps:

Preparation for rapid construction  
Program and application development  
Coding  
Unit, integration, and system testing

The software development team of programmers, coders, testers, and developers work together during this stage to make sure everything is working smoothly and that the end result satisfies the client's expectations and objectives.

This third phase is important because the client still gets to give input throughout the process. They can suggest alterations, changes, or even new ideas that can solve problems as they arise.

### Phase 4: Cutover

This is the implementation phase where the finished product goes to launch. It includes data conversion, testing, and changeover to the new system, as well as user training.

All final changes are made while the coders and clients continue to look for bugs in the system.

### Benefits of RAD methodology

RAD is one of the most successful software development programs available today, with numerous benefits for both software development teams as well as their clients.

Here are just a few advantages:

- RAD lets you break the project down into smaller, more manageable tasks.

- The task-oriented structure allows project managers to optimize their team's efficiency by assigning tasks according to members' specialties and experience.
- Clients get a working product delivered in a shorter time frame.
- Regular communication and constant feedback between team members and stakeholders increases the efficiency of the design and build process.

With a shorter planning phase and a focus on highly iterative design and construction, RAD teams are able to accomplish more in less time without sacrificing client satisfaction.

### Advantages & Disadvantages

#### Advantages

- ☐ Focus on automation
- ☐ Feedback from client
- ☐ Facilitates Go/no go decisions
- ☐ Risk Control

#### Disadvantages

- ☐ Team dependency
- ☐ Modularization is required in high order
- ☐ Not recommended for the small budgeted projects
- ☐ Reverse engineering may be difficult

- Agile models

### Agile software development values

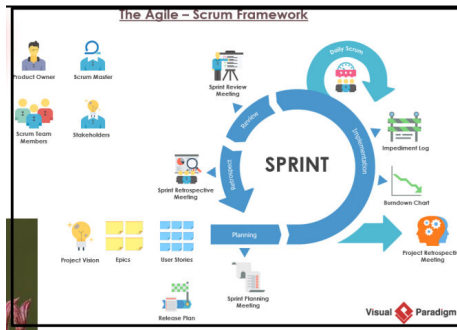
- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

### Agile software development principles

1. Customer satisfaction by early and continuous delivery of valuable software.
2. Welcome changing requirements, even in late development.
3. Deliver working software frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the primary measure of progress
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Best architectures, requirements, and designs emerge from self-organizing teams
12. Regularly, the team reflects on how to become more effective, and adjusts accordingly

### Agility

Ability to change direction and velocity at same time.



- The large project is broken down into Epics which are further broken down to User stories.
- Groups of user stories make up a work pack for a sprint. If a story is large, then it may get further broken down as sub stories.
- Each sprint may last for 14 days typically. Each sprint begins with a sprint meeting. In this meeting, the plan and responsibilities of the sprint team is decided. During the sprint duration, a scrum meeting is held, every day and progress is monitored using the artefacts of a impediment log and a burndown chart and a backlog list . A backlog list is list of things to do, a burndown chat is list of things done and remaining to do. An impediment log is the list of show-stoppers that need attention.
- The key role players are the Product owner, Scrum Master and the scrum members along with the stakeholders.
- Additionally a Project review meeting at the end of a scrum or more scrums to review the project status and check for progress is made.
- Every 3 or 4 scrums it is not uncommon to have a stabilization sprint where course corrections can be made. Also architectural sprints can be had in the beginning to validate architectural assumption and de-risk any technology related issues.
- The release plan is the plan that details the release of the sprint along with details of completion o stores in each sprint. This can undergo change based on project reviews.
- 

#### Advantages and disadvantages

##### Advantages

- ☐ Favours change
- ☐ Short communication for transparency
- ☐ Shift left philosophy

##### Disadvantages

- ☐ Focus on documentation is weak
- ☐ Losing the plot

#### Agile methodologies

- ☐ Agile Scrum Methodology
- ☐ Lean Software Development
- ☐ Kanban
- ☐ Extreme Programming (XP)
- ☐ Feature Driven Development (FDD)

##### Advantages

- ☐ Large scale projects
- ☐ Improving quality

##### Disadvantages

- ☐ Dynamic deadlines
- ☐ Skill dependent
- ☐ Scarce Documentation at detailed level

- ☐ Test Driven Development (TDD)

##### Advantages

- ☐ Simple, elegant, modular code
- ☐ Shift Left
- ☐ Concurrent documentation.
- ☐ Maintenance of code.
- ☐ Learning economies
- ☐ End – user view

##### Disadvantages

- ☐ Cart-before-the-horse effect
- ☐ Refactoring could become challenging
- ☐ Design volatility

Ques: Which of the following approaches includes the iterative nature of the prototyping approach and the linear nature of waterfall approach?

Ans. Spiral approach

## Software Development Methodologies



Properties of Model	Incremental Model	Spiral Model
Planning in early stage	Yes	Yes
Returning to an earlier step	Yes	Yes
Documentation	Yes but not much	Yes
Cost	Low	Expensive
Flexibility to change	Easy	Easy
User Involvement	Intermediate	High
Risk Involvement	Low	Medium to high risk
Testing	After every iteration	At the end of the engineering phase
Working software availability	At the end of every iteration	At the end of every iteration
Team size	Not Large Team	Large Team
Customer control over administrator	Yes	Yes

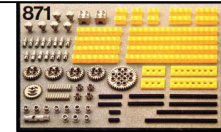
## Software Development Methodologies



Properties of Model	Waterfall	Prototyping
Planning in early stage	Yes – at beginning	Plan at every iteration
Returning to an earlier step	Difficult	Quite possible - encouraged
Documentation	Necessary, Sign off	Lesser Documentation
Cost	High	Relatively indeterminate
Flexibility to change	Low	high
User Involvement	Only in part – Req/Test	Continuous
Risk Involvement	Risk – lesser information	Lower
Testing	After completion of development	Continuous
Working software availability	Absolute End	MVP – available in parts
Team size	Large	Lesser
Customer control over administrator	Less	High

## Rapid Application Development

<https://brickset.com/sets/961-1/Parts-Pack>



Properties of Model	RAD
Planning in early stage	No
Returning to an earlier step	Yes
Documentation	Low
Cost	Low
Flexibility to change	Easy
User Involvement	Beginning
Risk Involvement	Low
Testing	After Coding
Working software availability	End of Life Cycle
Team size	Small
Customer control over administrator	Yes

## Software Requirement Specification

The production of the requirements stage of the software development process is Software Requirements Specifications (SRS) (also called a requirements document). This report lays a foundation for software engineering activities and is constructed when entire requirements are elicited and analyzed. SRS is a formal report, which acts as a representation of software that enables the customers to review whether it (SRS) is according to their requirements. Also, it comprises user requirements for a system as well as detailed specifications of the system requirements.

The SRS is a specification for a specific software product, program, or set of applications that perform particular functions in a specific environment. It serves several goals depending on who is writing it. First, the SRS could be written by the client of a system. Second, the SRS could be written by a developer of the system. The two methods create entirely various situations and establish different purposes for the document altogether. The first case, SRS, is used to define the needs and expectation of the users. The second case, SRS, is written for various purposes and serves as a contract document between customer and developer.

## Characteristics of good SRS



Following are the features of a good SRS document:

1. **Correctness:** User review is used to provide the accuracy of requirements stated in the SRS. SRS is said to be perfect if it covers all the needs that are truly expected from the system.
2. **Completeness:** The SRS is complete if, and only if, it includes the following elements:
  - (1). All essential requirements, whether relating to functionality, performance, design, constraints, attributes, or external interfaces.
  - (2). Definition of their responses of the software to all realizable classes of input data in all available categories of situations.

Note: It is essential to specify the responses to both valid and invalid values.

  - (3). Full labels and references to all figures, tables, and diagrams in the SRS and definitions of all terms and units of measure.
3. **Consistency:** The SRS is consistent if, and only if, no subset of individual requirements described in its conflict. There are three types of possible conflict in the SRS:
  - (1). The specified characteristics of real-world objects may conflicts. For example,
    - a) The format of an output report may be described in one requirement as tabular but in another as textual.
    - b) One condition may state that all lights shall be green while another states that all lights shall be blue.
  - (2). There may be a reasonable or temporal conflict between the two specified actions. For example,
    - (a) One requirement may determine that the program will add two inputs, and another may determine that the program will multiply them.
    - (b) One condition may state that "A" must always follow "B," while other requires that "A and B" co-occurs.
  - (3). Two or more requirements may define the same real-world object but use different terms for that object. For example, a program's request for user input may be called a "prompt" in one requirement's and a "cue" in another. The use of standard terminology and descriptions promotes consistency.
4. **Unambiguousness:** SRS is unambiguous when every fixed requirement has only one interpretation. This suggests that each element is uniquely interpreted. In case there is a method used with multiple definitions, the requirements report should determine the implications in the SRS so that it is clear and simple to understand.

5. **Ranking for importance and stability:** The SRS is ranked for importance and stability if each requirement in it has an identifier to indicate either the significance or stability of that particular requirement. Typically, all requirements are not equally important. Some prerequisites may be essential, especially for life-critical applications, while others may be desirable. Each element should be identified to make these differences clear and explicit. Another way to rank requirements is to distinguish classes of items as essential, conditional, and optional.

6. **Modifiability:** SRS should be made as modifiable as likely and should be capable of quickly obtain changes to the system to some extent. Modifications should be perfectly indexed and cross-referenced.

7. **Verifiability:** SRS is correct when the specified requirements can be verified with a cost-effective system to check whether the final software meets those requirements. The requirements are verified with the help of reviews.

8. **Traceability:** The SRS is traceable if the origin of each of the requirements is clear and if it facilitates the referencing of each condition in future development or enhancement documentation.

There are two types of Traceability:

1. **Backward Traceability:** This depends upon each requirement explicitly referencing its source in earlier documents.
2. **Forward Traceability:** This depends upon each element in the SRS having a unique name or reference number.

The forward traceability of the SRS is especially crucial when the software product enters the operation and maintenance phase. As code and design document is modified, it is necessary to be able to ascertain the complete set of requirements that may be concerned by those modifications.

9. **Design Independence:** There should be an option to select from multiple design alternatives for the final system. More specifically, the SRS should not contain any implementation details.

10. **Testability:** An SRS should be written in such a method that it is simple to generate test cases and test plans from the report.

11. **Understandable by the customer:** An end user may be an expert in his/her explicit domain but might not be trained in computer science. Hence, the purpose of formal notations and symbols should be avoided too as much extent as possible. The language should be kept simple and clear.

12. **The right level of abstraction:** If the SRS is written for the requirements stage, the details should be explained explicitly. Whereas, for a feasibility study, fewer analysis can be used. Hence, the level of abstraction modifies according to the objective of the SRS.

## Properties of a good SRS document

The essential properties of a good SRS document are the following:

**Concise:** The SRS report should be concise and at the same time, unambiguous, consistent, and complete. Verbose and irrelevant descriptions decrease readability and also increase error possibilities.

**Structured:** It should be well-structured. A well-structured document is simple to understand and modify. In practice, the SRS document undergoes several revisions to cope up with the user requirements. Often, user requirements evolve over a period of time. Therefore, to make the modifications to the SRS document easy, it is vital to make the report well-structured.

**Black-box view:** It should only define what the system should do and refrain from stating how to do these. This means that the SRS document should define the external behavior of the system and not discuss the implementation issues. The SRS report should view the system to be developed as a black box and should define the externally visible behavior of the system. For this reason, the SRS report is also known as the black-box specification of a system.

**Conceptual integrity:** It should show conceptual integrity so that the reader can merely understand it. Response to undesired events: It should characterize acceptable responses to unwanted events. These are called system response to exceptional conditions.

**Verifiable:** All requirements of the system, as documented in the SRS document, should be correct. This means that it should be possible to decide whether or not requirements have been met in an implementation.

## Problem Analysis

### Feasibility: study of requirements

- Operational or organizational feasibility
- Technical feasibility
- Economic feasibility

Feasibility Study in Software Engineering 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, a 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.

#### 1. 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.

#### 2. 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.

#### 3. 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.

#### 4. 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.

#### 5. 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.

### Feasibility Study Process :

The below steps are carried out during entire feasibility analysis.

1. Information assessment
2. Information collection
3. Report writing
4. General information

### Need of Feasibility Study :

Feasibility study is so important stage of Software Project Management Process as after completion of feasibility study it gives a conclusion of whether to go ahead with proposed project as it is practically feasible or to stop proposed project here as it is not right/feasible to develop or to think/analyze about proposed project again.

Along with this Feasibility study helps in identifying risk factors involved in developing and deploying system and planning for risk analysis also narrows the business alternatives and enhance success rate analyzing different parameters associated with proposed project development.

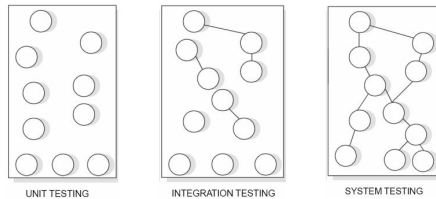
## Testing

- Different types and principles of software testing
- Levels of testing

### Levels of Testing

There are 3 levels of testing:

- Unit Testing
- Integration Testing
- System Testing



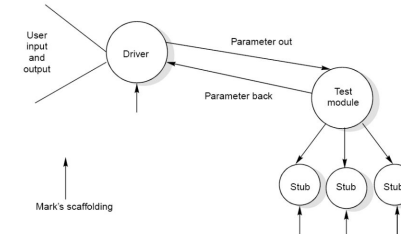
### 1. Unit Testing

#### Unit Testing

There are number of reasons in support of unit testing than testing the entire product.

1. The size of a single module is small enough that we can locate an error fairly easily.
2. The module is small enough that we can attempt to test it in some demonstrably exhaustive fashion.
3. Confusing interactions of multiple errors in widely different parts of the software are eliminated.
  - a. Function test - work done
  - b. Performance test - under exceptional cases, work done
  - c. Stress test - stress is provided to the function

### d. Structure test



- Tests each module individually.
  - Follows a white box testing (Logic of the program).
  - Done by developers.
- ### 2. Integration Testing
- Once all the modules have been unit tested, integration testing is performed.
  - It is systematic testing.
  - Produce tests to identify errors associated with interfacing.
  - Types:
    - Big Bang Integration testing
    - Top Down Integration testing
    - Bottom Up Integration testing
    - Mixed Integration testing

### Integration Testing

The purpose of unit testing is to determine that each independent module is correctly implemented. This gives little chance to determine that the interface between modules is also correct, and for this reason integration testing must be performed. One specific target of integration testing is the interface: whether parameters match on both sides as to type, permissible ranges, meaning and utilization.

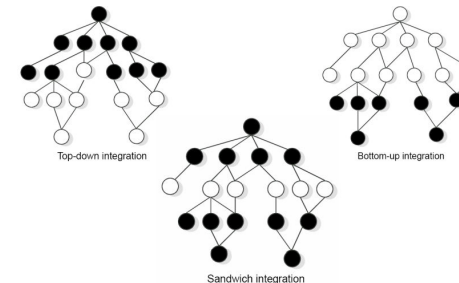


Fig. 30 : Three different integration approaches



### 3. System Testing

- The system as a whole is tested to uncover requirement errors.
- Verifies that all system elements work properly and that overall system function and performance has been achieved.
- Types:
  - Alpha Testing
  - Beta Testing
  - Acceptance Testing
  - Performance Testing

#### System Testing

Of the three levels of testing, the system level is closest to everyday experiences. We test many things; a used car before we buy it, an on-line cable network service before we subscribe, and so on. A common pattern in these familiar forms is that we evaluate a product in terms of our expectations; not with respect to a specification or a standard. Consequently, goal is not to find faults, but to demonstrate performance. Because of this we tend to approach system testing from a functional standpoint rather than from a structural one. Since it is so intuitively familiar, system testing in practice tends to be less formal than it might be, and is compounded by the reduced testing interval that usually remains before a delivery deadline.

During system testing, we should evaluate a number of attributes of the software that are vital to the user and are listed in Fig. 31. These represent the operational correctness of the product and may be part of the software specifications.

<b>Usable</b>	Is the product convenient, clear, and predictable?
<b>Secure</b>	Is access to sensitive data restricted to those with authorization?
<b>Compatible</b>	Will the product work correctly in conjunction with existing data, software, and procedures?
<b>Dependable</b>	Do adequate safeguards against failure and methods for recovery exist in the product?
<b>Documented</b>	Are manuals complete, correct, and understandable?

Fig. 31 : Attributes of software to be tested during system testing

#### Alpha testing

- It is carried out by the test team within the developing organization .
- Alpha testing is a type of acceptance testing, which is performed to identify all possible bugs/issues before releasing the product to the end-user. The test carried out by a team of users to find out the bugs that were not found previously by other tests. It needs lab environment, and usually, the testers are an internal employee of the organization. This testing is called alpha because it is done early on, near the end of the software development, but before beta testing.

#### Beta Testing

- It is performed by a selected group of friendly customers.
- Beta Testing is a type of acceptance testing; it is the final test before shipping a product to the customers. Beta testing of a product is implemented by "real users "of the software application in a "real environment." In this phase of testing, the software is released to a limited number of end-users of the product to obtain feedback on the product quality.

#### User acceptance test(Acceptance Testing)

- It is performed by the customer to determine whether to accept or reject the delivery of the system.
- User acceptance testing (UAT) is a type of testing, which is done by the customer before accepting the final product. Generally, UAT is done by the customer (domain expert) for their satisfaction, and check whether the application is working according to given business scenarios, real-time scenarios.

#### Performance Testing

- It is carried out to check whether the system meets the nonfunctional requirements identified in the SRS document.
- Types:
  - Stress Testing
  - Volume Testing
  - Configuration Testing
  - Compatibility Testing
  - Regression Testing
  - Recovery Testing
  - Maintenance Testing
  - Documentation Testing
  - Usability Testing

#### Discussion

- In order to be cost effective, the testing must be concentrated on areas where it will be most effective
- The testing should be planned such that when testing is stopped for whatever reason, the most effective testing in the time allotted has already been done.
- The absence of an organizational testing policy may result in too much effort and money will be spent on testing, attempting to achieve a level of quality that is impossible or unnecessary.

- Regression testing

Regression testing is the activity that helps to ensure that changes (due to testing or for other reasons) do not introduce unintended behavior or additional errors.

Regression testing may be conducted manually, by re-executing a subset of all test cases or using automated capture/playback tools. Capture/playback tools enable the software engineer to capture test cases and results for subsequent playback and comparison.

The regression test suite (the subset of tests to be executed) contains three different classes of test cases:

- A representative sample of tests that will exercise all software functions.

- Additional tests that focus on software functions that are likely to be affected by the change.

- Tests that focus on the software components that have been changed.

- As integration testing proceeds, the number of regression tests can grow quite large.

- Validation Testing

- Validation Testing**

- It refers to test the software as a complete product.

- This should be done after unit & integration testing.

- Alpha, beta & acceptance testing are nothing but the various ways of involving customer during testing.

- Validation testing improves the quality of software product in terms of functional capabilities and quality attributes.

- Functional testing

- Structural testing

- Test documentation

- Many organizations place a heavy emphasis on test documentation

- This can include:

- Test Strategy
    - Test Plan
    - Test Cases, Test Scripts
    - Bug Reports
    - Test Completion Reports

- Metrics

- Software testing often makes use of a wide variety of metrics, e.g.:

- Test pass / fail ratios
    - Bug arrival rates
    - Bug detection effectiveness
    - Bugs per KLOC (kilo lines of code)

- Etc.

- Test plan

- It is a systematic approach to test a system i.e. software. The plan typically contains a detailed understanding of what the eventual testing workflow will be.

- Test cases specifications

- It is a specific procedure of testing a particular requirement.

- It will include:

- Identification of specific requirement tested
    - Test case success/failure criteria
    - Specific steps to execute test
    - Test data

- Verification vs Validation

- Verification: The software should confirm to its specification (Are we building the product right?)

- Validation: The software should do what the user really requires (Are we building the right product?)

- Reliability Assessment

- Test case design

- Importance of unit test

- System Integration test

- 

- Process Models

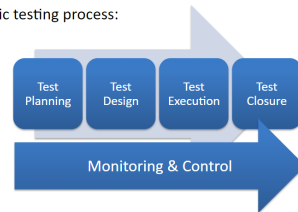
- There are a number of process models that describe how testing should proceed

- We are going to look at a generic testing process, and the "V Model"

- "Essentially, all models are wrong, but some are useful" - George E. P. Box

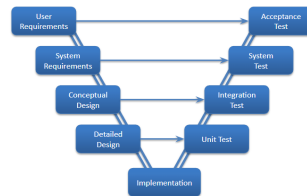
- There are a number of rational objections to both of these models

• A generic testing process:

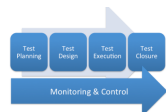


○ A generic testing process

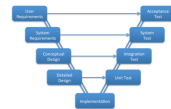
• The V Model:



○ V model



• To what degree can testing be pre-planned?  
• What about tests which are identified during execution?



• Why wait for a complete implementation before testing?  
• What about bugs identified during testing?

## • Software Maintenance

### Software Maintenance

**Software maintenance** is the modification of a software product after delivery to correct faults, to improve performance or other attributes. A common perception of maintenance is that it merely involves fixing defects. However, one study indicated that the majority, over 80%, of the maintenance effort is used for non-corrective actions. This perception is perpetuated by users submitting problem reports that in reality are functionality enhancements to the system. Software maintenance is needed to correct error enhance feature, portability to new platform.

## Software Testing

### Introduction

- It is the process used to identify the correctness, completeness and quality of developed computer software.

37

- It is the process of executing a program/application under positive and negative conditions by manual or automated means. It checks for the :-
  - Specification
  - Functionality
  - Performance

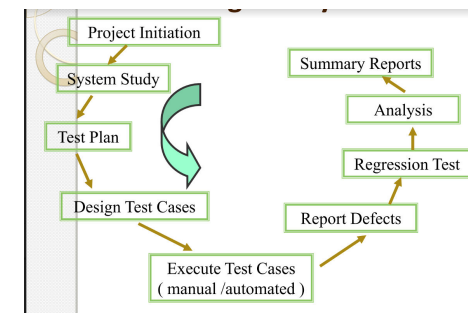
### Objectives

- Uncover as many as errors (or bugs) as possible in a given product.
- Demonstrate a given software product matching its requirement specifications.
- Validate the quality of a software testing using the minimum cost and efforts.
- Generate high quality test cases, perform effective tests, and issue correct and helpful problem reports.

### Error, Bug, Fault & Failure

- Error : It is a human action that produces the incorrect result that produces a fault.
- Bug : The presence of error at the time of execution of the software.
- Fault : State of software caused by an error.
- Failure : Deviation of the software from its expected result. It is an event.

### Testing Life Cycle

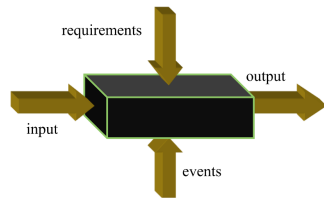


### Testing Methodologies

#### Black box testing

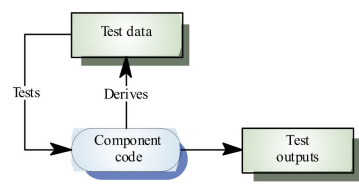
- No knowledge of internal program design or code required.
- Tests are based on requirements and functionality.

38



### White box testing

- Knowledge of the internal program design and code required.
- Tests are based on coverage of code statements, branches, paths, conditions.



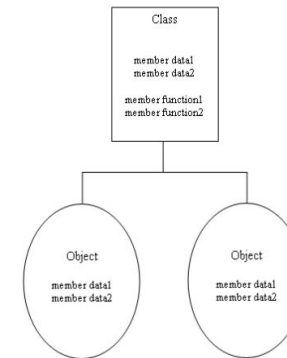
### Key topics:

#### • Object Oriented design concept

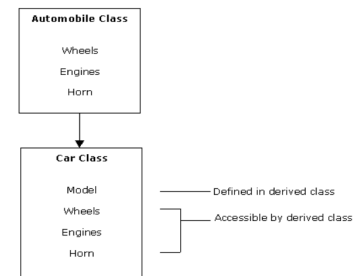
- In the design phase, there are two approaches to software development:
- Function-oriented approach:
  - Is module-centric and concentrates on functions of the software.
- Object-oriented approach:
  - Portrays things as they exist in the real world.
  - Introduces the concept of inheritance which allows reuse of existing code components.
  - Supports inheritance, reusability and encapsulation of data, abstraction, and polymorphism.

#### • Overview of Object-Oriented Concepts

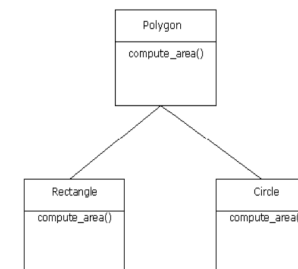
- A class is an abstract data type that contains a set of attributes and functions.
- An object is an instance of a class.



- 
- **Inheritance** refers to sharing of attributes and behaviors among classes based upon hierarchical relationships.



- 
- **Abstraction** focuses on essential, inherent aspects of an entity ignoring its implementation details.



- 
- **Encapsulation** means preventing access to non-essential details.

- **Polymorphism** is the concept of using operators or functions in different ways depending on what they are operating on.
- Using operators in different ways depending on what they are operating on is called **operator overloading**.
- Using functions in different ways is called **function overloading**.
- Consider an example of OO concept:
  - Countryside Markets has no formal system to store its employees' information. The organization now wants to store detailed information about its employees, such as name, age, date of birth, e-mail id, department, employee code, salary drawn, and date of joining.
- Countryside Markets class hierarchy

Ques: Which phase of SDLC includes translation of the requirements specified in the SRS into a logical structure that can be implemented in a programming language?

Ans. Design

### ● Role of Object-Oriented Analysis and Design (OOAD) in SDLC

- The OO approach does not replace the standard approaches, such as Data Flow Diagrams (DFD) or Entity Relationship (ER) diagrams. It is only an addition to the existing toolkit.
- OOAD uses the OO approach to solve real world problems. It uses the OO approach to analyze the system requirements and break a large and complex system into smaller and simpler components.
- OOAD is analysis of requirements and design of software systems in terms of the objects, classes, encapsulation, inheritance, polymorphism, abstraction, and dynamic binding.
- OOAD is a methodology that can be applied to linear, iterative, or incremental approaches.

### ● Software project management - in notes

## Project Scheduling

A schedule in your project's time table actually consists of sequenced activities and milestones that are needed to be delivered under a given period of time.

Project schedule simply means a mechanism that is used to communicate and know about that tasks are needed and has to be done or performed and which organizational resources will be given or allocated to these tasks and in what time duration or time frame work is needed to be performed. Effective project scheduling leads to success of project, reduced cost, and increased customer satisfaction. Scheduling in project management means to list out activities, deliverables, and milestones within a project that are delivered. It contains more notes than your average weekly planner notes. The most common and important form of project schedule is Gantt chart.

### Process :

The manager needs to estimate time and resources of project while scheduling project. All activities in project must be arranged in a coherent sequence that means activities should be arranged in a logical and well-organized manner for easy to understand. Initial estimates of project can be made optimistically which means estimates can be made when all favorable things will happen and no threats or problems take place.

The total work is separated or divided into various small activities or tasks during project schedule. Then, Project manager will decide time required for each activity or task to get completed. Even some activities

are conducted and performed in parallel for efficient performance. The project manager should be aware of fact that each stage of project is not problem-free.

### Problems arise during Project Development Stage :

- People may leave or remain absent during particular stage of development.
- Hardware may get failed while performing.
- Software resource that is required may not be available at present, etc.

The project schedule is represented as set of chart in which work-breakdown structure and dependencies within various activities are represented. To accomplish and complete project within a given schedule, required resources must be available when they are needed. Therefore, resource estimation should be done before starting development.

### Resources required for Development of Project :

- Human effort
- Sufficient disk space on server
- Specialized hardware
- Software technology
- Travel allowance required by project staff, etc.

### Advantages of Project Scheduling :

There are several advantages provided by project schedule in our project management:

- It simply ensures that everyone remains on same page as far as tasks get completed, dependencies, and deadlines.
- It helps in identifying issues early and concerns such as lack or unavailability of resources.
- It also helps to identify relationships and to monitor process.
- It provides effective budget management and risk mitigation.

### Javatpoint

Project-task scheduling is a significant project planning activity. It comprises deciding which functions would be taken up when. To schedule the project plan, a software project manager wants to do the following:

1. Identify all the functions required to complete the project.
2. Break down large functions into small activities.
3. Determine the dependency among various activities.
4. Establish the most likely size for the time duration required to complete the activities.
5. Allocate resources to activities.
6. Plan the beginning and ending dates for different activities.
7. Determine the critical path. A critical way is the group of activities that decide the duration of the project.

The first method in scheduling a software plan involves identifying all the functions required to complete the project. A good judgment of the intricacies of the project and the development process helps the supervisor to identify the critical role of the project effectively. Next, the large functions are broken down into a valid set of small activities which would be assigned to various engineers. The work breakdown structure formalism supports the manager to breakdown the function systematically after the project manager has broken down the purpose and constructs the work breakdown structure; he has to find the dependency among the activities. Dependency among the various activities determines the order in which the various events would be carried out. If an activity A necessary the results of another activity B, then activity A must be scheduled

after activity B. In general, the function dependencies describe a partial ordering among functions, i.e., each service may precede a subset of other functions, but some functions might not have any precedence ordering describe between them (called concurrent function). The dependency among the activities is defined in the pattern of an activity network.

Once the activity network representation has been processed out, resources are allocated to every activity. Resource allocation is usually done using a Gantt chart. After resource allocation is completed, a PERT chart representation is developed. The PERT chart representation is useful for program monitoring and control. For task scheduling, the project plan needs to decompose the project functions into a set of activities. The time frame when every activity is to be performed is to be determined. The end of every action is called a milestone. The project manager tracks the function of a project by audit the timely completion of the milestones. If he examines that the milestones start getting delayed, then he has to handle the activities carefully so that the complete deadline can still be met.

## Project Staffing

Staffing is that part of management which is concerned with obtaining, utilizing, and maintaining capable people to fill all positions in the organization from top-level to bottom level. It involves the scientific and systematic procurement, allocation, utilization, conservation, and development of human resources. It is the art of acquiring, developing, and maintaining a satisfactory and satisfied workforce. Staffing is that function by which a manager builds an organization through the recruitment, selection, and development of the individual, which also includes a series of activities. It ensures that the organization has the right number of people at the right places, at the right time, and performing the right thing.

## Staffing Process

As we know, the prime concern of the staffing function in the management process is in the fulfilment of the manpower requirements within an organization. These requirements may arise in the case of starting a new enterprise or expanding the existing one. It may also arise as the need for replacing those who quit, retire, transfer, or are promoted from the job. In any case, the need for 'the right person for the right job, at the right time' needs an emphasis. The process of staffing consists of several interrelated activities, such as planning for human resources requirements, recruitment, selection, training development, remuneration, and so on. These activities together make the staffing process. Therefore, these are called elements or steps of the staffing process.

## Manpower Planning:

Human resource management is a process of determining the number and type of personnel required for filling the vacant job in an organization. Manpower requirements involve two kinds of analysis, i.e., workload analysis and workforce analysis. Workload analysis involves determining the number and type of employees required to perform various jobs and achieve organizational objectives. Workforce analysis shows the number and type of human resources available with an organization.

The difference between workload and workforce is calculated to determine shortage and surplus of manpower. Excess workload indicates understaffing, i.e., the need of appointing more people and excess workforce indicates overstaffing, i.e., need to remove or transfer some employees to other places.

## Recruitment:

After estimating manpower requirements, the second step in the process of staffing is recruitment. Recruitment refers to a process of searching for prospective employees and encouraging them to apply for jobs in the organization. It involves identifying various resources of human force and attracting them to apply for the job. The main purpose of a requirement is to create a pool of applicants by a large number of qualified candidates. Recruitment can be done by both internal and external sources of recruitment. Internal sources may be used to a limited extent, and to get fresh talent and a wider choice, external sources can be used.

## Selection:

Selection is the process of choosing and appointing the right candidates for various job positions in the organization. It is treated as a negative process because it involves the rejection of some candidates. There are many steps involved in the process of employee selection. These steps include preliminary screening, filling-in application, written test, interviews, medical examination, checking references, and issuing a letter of appointment to the candidates. The most suitable candidates who meet the requirement of the vacant job are selected. The process of selection serves two important purposes, firstly, it ensures that the organization gets the best among the available candidates, and secondly, it boosts up the self-esteem and prestige of the candidates.

## Placement and Orientation:

After selection, an appropriate job is assigned to each selected person. Placement is the process of matching the candidates with the jobs in the organization. Under this process, every selected candidate is assigned a job most suitable for him. The purpose of placement is to fit the right person to the right job so that the efficiency of work is high and the employees get personal satisfaction. Correct placement helps to reduce labour turnover and absenteeism. Here, orientation means introducing new employees to the organization. It is the process of introducing and familiarizing newly appointed candidates with their job, work groups and the organization so that they may feel at home in the new environment.

## Training and Development:

People are in search of careers and not jobs. Every individual must be given a chance to rise to the top. The most favourable way for this to happen is to promote employee learning. For this, organizations either provide training themselves within the organization or through external institutions. This is beneficial for the organization as well. If the employees are motivated enough, it will increase their competence and will be able to perform even better for the organization with greater efficiency and productivity. By providing such opportunities to its employees for career advancement, the organization captivates the interest and holds on of its talented employees. The majority of the organization has a distinct department for this purpose, that is, the Human Resource Department. Though in small organizations, the line manager has to do all the managerial functions viz, planning, organizing, staffing, controlling, and directing. The process of staffing further involves three more stages.

## Performance appraisal:

After training the employees and having them on the job for some time, there should be an evaluation done on their performance. Every organization has its means of appraisal whether formal or informal. Appraisal refers to the evaluation of the employees of the organization based on their past or present performance by some pre-decided standards. The employee should be well aware of his standards and his superior is

responsible for providing feedback on his performance. The process of performance appraisal, thus includes specifying the job, performing appraisal performance, and providing feedback.

### Promotion and Career planning:

It has now become important for all organizations to deal with career-related issues and promotional routes for employees. The managers should take care of the activities that serve the long-term interests of the employees. They should be encouraged from time to time, which will help the employees to grow and find their true potential. Promotions are an essential part of any employee's career. Promotion refers to the transferring of employees from their current positions to a higher level increasing their responsibilities, authority and pay.

### Compensation:

Every organization needs to set up plans for the salary and wages of the employees. There are several ways to develop payment plans for the employees depending upon the significance of the job. The worth of the job needs to be decided. Therefore, all kinds of payments or rewards provided to the employees is referred to as compensation. The compensation may be in the form of direct financial payments, such as salary, wages, bonuses, etc., or indirect payments like insurance or vacations provided to the employee.

Direct financial payments are of two kinds, that is, performance-based and time-based. In a time-based payment plan, the salary or wages are paid daily, weekly, monthly, or annually, whereas, the performance-based payment plan is the payment of salary or wages according to the set task. There are many ways in which the compensation of the employee based on their performance can be calculated. There are also plans, which are a combination of both time-based and performance-based. There are a few factors that affect the payment plan, such as legal, company policy, union, and equity. Thus, staffing is the process that includes possession, retention, promotion, and compensation of the human capital, that is, the most important resource of the organization. There are several factors such as the supply and demand of specific skills in the labour market, legal and political considerations, the company's image, policy, unemployment rate, human resource planning cost, labour market conditions, technological developments, general economic environment, etc., that may affect the execution of recruitment, selection, and training.

### Aspects or Components of Staffing

There are three aspects or components of staffing, namely, recruitment, selection, and training. They are defined below:

#### Recruitment:

It is the process of finding potential candidates for a particular job in an organization. The process of recruitment involves persuading people to apply for the available positions in the organization.

#### Selection:

It is the process of recognizing potential and hiring the best people out of several possible candidates. This is done by shortlisting and choosing the deserving and eliminating those who are not suitable for the job.

#### Training:

It is the process that involves providing the employees with an idea of the type of work they are supposed to do and how it is to be done. It is a way of keeping the employees updated on the way of work in an organization and the new and advanced technologies.

## Risk Management

A risk is a probable problem- it might happen or it might not. There are main two characteristics of risk

#### Uncertainty

the risk may or may not happen that means there are no 100% risks.

#### loss –

If the risk occurs in reality , undesirable result or losses will occur.

Risk management is a sequence of steps that help a software team to understand , analyze and manage uncertainty. Risk management consists of

1. Risk Identification
2. Risk analysis
3. Risk Planning
4. Risk Monitoring

A computer code project may be laid low with an outsized sort of risk. so as to be ready to consistently establish the necessary risks which could have an effect on a computer code project, it's necessary to reason risks into completely different categories. The project manager will then examine the risks from every category square measure relevant to the project.

There square measure 3 main classes of risks that may have an effect on a computer code project:

#### 1. Project Risks:

Project risks concern various sorts of monetary funds, schedules, personnel, resource, and customer-related issues. a vital project risk is schedule slippage. Since computer code is intangible, it's terribly tough to observe and manage a computer code project. it's terribly tough to manage one thing that can not be seen. For any producing project, like producing cars, the project manager will see the merchandise taking form.

For example, see that the engine is fitted, at the moment the area of the door unit fitted, the automotive is obtaining painted, etc. so he will simply assess the progress of the work and manage it. The physical property of the merchandise being developed is a vital reason why several computer codes come to suffer from the danger of schedule slippage.

#### 2. Technical Risks:

Technical risks concern potential style, implementation, interfacing, testing, and maintenance issues. Technical risks conjointly embody ambiguous specifications, incomplete specification, dynamic specification, technical uncertainty, and technical degeneration. Most technical risks occur thanks to the event team's lean information concerning the project.

### 3. Business Risks:

This type of risk embodies the risks of building a superb product that nobody needs, losing monetary funds or personal commitments, etc.

## Principle of Risk Management

1. **Global Perspective:** In this, we review the bigger system description, design, and implementation. We look at the chance and the impact the risk is going to have.
2. **Take a forward-looking view:** Consider the threat which may appear in the future and create future plans for directing the next events.
3. **Open Communication:** This is to allow the free flow of communications between the client and the team members so that they have certainty about the risks.
4. **Integrated management:** In this method risk management is made an integral part of project management.
5. **Continuous process:** In this phase, the risks are tracked continuously throughout the risk management paradigm.

## Quality Management

Software Quality Management ensures that the required level of quality is achieved by submitting improvements to the product development process. SQA aims to develop a culture within the team and it is seen as everyone's responsibility.

Software Quality management should be independent of project management to ensure independence of cost and schedule adherences. It directly affects the process quality and indirectly affects the product quality.

## Activities of Software Quality Management:

### Quality Assurance

QA aims at developing Organizational procedures and standards for quality at Organizational level.

### Quality Planning

Select applicable procedures and standards for a particular project and modify as required to develop a quality plan.

### Quality Control -

Ensure that best practices and standards are followed by the software development team to produce quality products.

## Software Quality Assurance

Software Quality Assurance (SQA) is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards are suitable for the project and implemented correctly.

Software Quality Assurance is a process which works parallel to development of software. It focuses on improving the process of development of software so that problems can be prevented before they become a major issue. Software Quality Assurance is a kind of Umbrella activity that is applied throughout the software process.

### Software Quality Assurance has:

- A quality management approach
- Formal technical reviews
- Multi testing strategy
- Effective software engineering technology
- Measurement and reporting mechanism

### Major Software Quality Assurance Activities:

#### 1. SQA Management Plan:

Make a plan for how you will carry out the sqa through out the project. Think about which set of software engineering activities are the best for project. check level of sqa team skills.

#### 2. Set The Check Points:

SQA team should set checkpoints. Evaluate the performance of the project on the basis of collected data on different check points.

#### 3. Multi testing Strategy:

Do not depend on a single testing approach. When you have a lot of testing approaches available use them.

#### 4. Measure Change Impact:

The changes for making the correction of an error sometimes re introduces more errors keep the measure of impact of change on project. Reset the new change to change check the compatibility of this fix with whole project.



## 5. Manage Good Relations:

In the working environment managing good relations with other teams involved in the project development is mandatory. Bad relation of sqa team with programmers team will impact directly and badly on project. Don't play politics.

### Benefits of Software Quality Assurance (SQA):

- SQA produces high quality software.
- High quality application saves time and cost.
- SQA is beneficial for better reliability.
- SQA is beneficial in the condition of no maintenance for a long time.
- High quality commercial software increase market share of company.
- Improving the process of creating software.
- Improves the quality of the software.

### Disadvantage of SQA:

- There are a number of disadvantages of quality assurance. Some of them include adding more resources, employing more workers to help maintain quality and so much more.

## Software Configuration Management

Whenever a software is build, there is always scope for improvement and those improvements brings changes in picture. Changes may be required to modify or update any existing solution or to create a new solution for a problem. Requirements keeps on changing on daily basis and so we need to keep on upgrading our systems based on the current requirements and needs to meet desired outputs. Changes should be analyzed before they are made to the existing system, recorded before they are implemented, reported to have details of before and after, and controlled in a manner that will improve quality and reduce error. This is where the need of System Configuration Management comes.

System Configuration Management (SCM) is an arrangement of exercises which controls change by recognizing the items for change, setting up connections between those things, making/characterizing instruments for overseeing diverse variants, controlling the changes being executed in the current framework, inspecting and revealing/reporting on the changes made. It is essential to control the changes in light of the fact that if the changes are not checked legitimately then they may wind up undermining a well-run programming. In this way, SCM is a fundamental piece of all project management activities.

### Processes involved in SCM –

Configuration management provides a disciplined environment for smooth control of work products. It involves the following activities:

#### 1. Identification and Establishment –

Identifying the configuration items from products that compose baselines at given points in time (a baseline is a set of mutually consistent Configuration Items, which has been formally reviewed and agreed upon, and serves as the basis of further development). Establishing relationship among items, creating a mechanism to manage multiple level of control and procedure for change management system.

#### 2. Version control –

Creating versions/specifications of the existing product to build new products from the help of SCM system. A description of version is given below:

Suppose after some changes, the version of configuration object changes from 1.0 to 1.1. Minor corrections and changes result in versions 1.1.1 and 1.1.2, which is followed by a major update that is object 1.2. The development of object 1.0 continues through 1.3 and 1.4, but finally, a noteworthy change to the object results in a new evolutionary path, version 2.0. Both versions are currently supported.

#### 3. Change control –

Controlling changes to Configuration items (CI). The change control process is explained in Figure below: A change request (CR) is submitted and evaluated to assess technical merit, potential side effects, overall impact on other configuration objects and system functions, and the projected cost of the change. The results of the evaluation are presented as a change report, which is used by a change control board (CCB) —a person or group who makes a final decision on the status and priority of the change. An engineering change Request (ECR) is generated for each approved change.

Also CCB notifies the developer in case the change is rejected with proper reason. The ECR describes the change to be made, the constraints that must be respected, and the criteria for review and audit. The object to be changed is "checked out" of the project database, the change is made, and then the object is tested again. The object is then "checked in" to the database and appropriate version control mechanisms are used to create the next version of the software.

#### 4. Configuration auditing –

A software configuration audit complements the formal technical review of the process and product. It focuses on the technical correctness of the configuration object that has been modified. The audit confirms the completeness, correctness and consistency of items in the SCM system and track action items from the audit to closure.

#### 5. Reporting –

Providing accurate status and current configuration data to developers, tester, end users, customers and stakeholders through admin guides, user guides, FAQs, Release notes, Memos, Installation Guide, Configuration guide etc .

#### SCM Tools –

Different tools are available in market for SCM like: CFEngine, Bcfg2 server, Vagrant, SmartFrog, CLEAR CASETOOL (CC), SaltStack, CLEAR QUEST TOOL, Puppet, SVN- Subversion, Perforce, TortoiseSVN, IBM Rational team concert, IBM Configuration management version management, Razor, Ansible, etc. There are many more in the list.

It is recommended that before selecting any configuration management tool, have a proper understanding of the features and select the tool which best suits your project needs and be clear with the benefits and drawbacks of each before you choose one to use.

### SCM

A configuration of the product refers not only to the product's constituent but also to a particular version of the component.

Therefore, SCM is the discipline which

- Identify change
- Monitor and control change
- Ensure the proper implementation of change made to the item.
- Auditing and reporting on the change made.

Configuration Management (CM) is a technic of identifying, organizing, and controlling modification to software being built by a programming team.

The objective is to maximize productivity by minimizing mistakes (errors).

CM is used to essential due to the inventory management, library management, and updation management of the items essential for the project.

#### Why do we need Configuration Management?

Multiple people are working on software which is consistently updating. It may be a method where multiple version, branches, authors are involved in a software project, and the team is geographically distributed and works concurrently. It changes in user requirements, and policy, budget, schedules need to be accommodated.

### Importance of SCM

- It is practical in controlling and managing the access to various SCIs e.g., by preventing the two members of a team for checking out the same component for modification at the same time.
- It provides the tool to ensure that changes are being properly implemented.
- It has the capability of describing and storing the various constituent of software.
- SCM is used in keeping a system in a consistent state by automatically producing derived version upon modification of the same component

### Coding

The objective of the coding phase is to transform the design of a system into code in a high-level language and then to unit test this code.

Good software development organizations normally require their programmers to adhere to some well-defined and standard style of coding called coding standards.

#### Coding Standards-

- A coding standard gives a uniform appearance to the codes written by different engineers.
- It enhances code understanding.
- It encourages good programming practices.

#### Coding Standards and Guidelines

#### *Representative Coding Standards*

- Limiting the use of global data type
- Contents of the headers preceding codes for different modules
- Naming conventions for global variables, local variables, and constant identifiers
- Error return conventions and exception handling mechanisms

## Representative Coding Guideline

- Do not use a coding style that is too clever or too difficult to understand
- Avoid obscure side effects
- Do not use an identifier for multiple purposes
- The code should be well-documented
- The length of any function should not exceed 10 source lines
- Do not use goto statements

### Code Review

## Code Review

Code review for a model is carried out after the module is successfully compiled and the all the syntax errors have been eliminated

*Normally, two types of reviews are carried out on the code of a module*

- **Code Walk Through:**

To discover the algorithm and logical errors in the code.

- **Code Inspection:**

The aim of code inspection is to discover some common types of errors caused due to oversight and improper programming.

## Software Documentation

Good documents are very useful and server the following purposes:

- Good documents enhance understandability and maintainability of a software product.
- Helps the users in effectively using the system.
- Helps in effectively handling the manpower turnover problem
- Helps the manager in effectively tracking the progress of the project

Software Documentation classified into the following:

- *Internal documentation:* These are provided in the source code itself
- *External documentation:* These are the supporting documents that usually accompany a software product

## New tools and techniques of SE process management, version control, lifecycle management

### Software development is constantly evolving

Software development is constantly evolving, thanks to a variety of changes and improvements in the [software development lifecycle](#) (SDLC), such as:

- The constantly changing technological landscape
- New consumer and business demands
- Market trends

A technology that is prominent today may quickly become obsolete. If we look at the current software development process, many significant trends are changing how we develop, deploy, and manage software and the platforms to develop for.

In this post, let's discuss some trends in software development that will have a high impact on the broad IT industry.

### Low-code & no-code software development


This concept might seem contradictory: after all, coding is the basis of software development.

However, software development is increasingly complex, and the need to deliver software faster never slows. This has put software development teams under constant pressure and made new talent reluctant to enter such high-pressure environments.

Exactly for these reasons have [low-code and no-code](#) products quickly gained popularity across the software development field. Importantly, low-code and no-code tools and platforms will never replace actual coding in software development—someone needs to develop them. But more people are getting on board with the benefits they bring:

Low-code and no-code options can be utilized to complement software development pipelines by allowing users to develop, deploy, and manage some parts of software solutions and delivery pipelines. Another advantage? Lowering

the barrier to enter the software development field and helping to attract new talent.



Low Code	No Code
Use for more complex applications	Use for reporting, analytics and tracking apps
Usually for apps that are foundational or run important processes for a business	Apps that evolve with frequent updates and changes in use-case
Apps with: <ul style="list-style-type: none"><li>• More than 5 years lifecycle</li><li>• Fewer updates</li></ul>	Can be integrative or stand alone
Can be mission critical	Good for self-deploying apps
Offers more developer control	Mobile responsive

### Big data security

[Big data](#) and [data science](#) have become the norm in the IT industry, with data the cornerstone of any business.

Software development has evolved to cater to big data needs from collecting, storing, and analyzing data. With the increased scrutiny on big data by users, regulatory bodies, and governments, [securing all this collected data](#) has become the number one priority for any organization.

This need, in turn, has led to integrating security as a fundamental component. It requires integrating security particles from the beginning of the development in any software that interacts with data. With the rise of the "data as a service" platform and continuous internet threats, big data security will become the next major trend.

### DevSecOps

DevOps has changed the way we develop software leading to more agile and faster software developments while also improving the overall quality of software. However, as the cloud has become central to most software developments, along

with increased reliance on the internet to deliver software, the threats for software may also increase.

All these facts have led to the inability of security teams to keep up with the rapid pace of the software development and delivery process.

In contrast, [DevSecOps](#) has integrated security into every nook and cranny of the software development process, with security teams constantly monitoring all aspects of the DevOps process. This integration of security as a first-class citizen in the SDLC leads to more secure software. Hence, DevSecOps will supplant DevOps as the standard operating process for [most development teams](#) in the coming years.

(Compare [SecOps to DevSecOps](#).)

### Increasing reliance on artificial intelligence

AI has already become a core component in most software, from simple computer vision applications to enterprise-scale predictive analytics. Making huge gains in recent years, [artificial intelligence](#) seems to eye no end to this expansive growth. As most narrow AIs have become intelligent enough to completely replace humans in most aspects, AI that can match human intelligence may become a reality in the next decade.

AI coupled with [neural networks](#) and machine learning software is changing from static pieces of logic to self-learning and evolving entities. This will change how software is developed. Most developments will shift from creating static logic to creating algorithms that can learn and evolve to meet changing end-user requirements.

(Compare [AI to machine learning](#).)

### Augmented reality, virtual reality & mixed reality

While [AR, VR, and MR](#) seem to have stagnated in recent years, the reality shows constant progress in all three technologies. With most consumers craving new experiences, these technologies will fundamentally change how we see and interact with the world.

Augmented reality (AR)

This can be the most widely used technology as it has applications across most industries, from eCommerce to changing the way users shop to navigation with AR-powered GPS apps.

AR can become a core part of the user experience with more and more reliance on digital technologies. All this equates to software developments targeted at AR. Furthermore, AR development is quickly gaining momentum with tools like ARCore and ARKit from Google and Apple.

(Experience [the AR cloud](#).)

Virtual reality (VR)

When thinking of VR, gaming immediately comes to our mind—because that's its big use case.

VR can offer an unparalleled user experience, immersing you in entirely virtual worlds from a first-person perspective. You can be exploring a distant planet now and, in the next second, exploring the deepest trenches in the ocean, all in the comfort of your home.

However, most of us forget that VR has applications far beyond gaming—applications that can change education and other entertainment sectors, in particular. Students can get hands-on experience in a virtual world and even watch VR-based movies. Virtual reality-based software developments will be widespread thanks to two new developments:

- Dedicated tools like Amazon Sumerian and Google VR
- Lower VR hardware costs

Mixed reality (MR)

MR is the bridge between reality and digital worlds—it's the platform that bends the digital with the physical. Mixed reality has the potential to fundamentally change how we interact with the physical world.

Companies like Microsoft backing MR with products like HoloLens have laid the groundwork for MR becoming the next big thing.

## Progressive web apps

With the continuous growth of mobile and web-based applications, developing and maintaining separate applications for both platforms may become an unnecessary burden.

PIA or Progressive Web Apps can enable developers to create mobile-focused versions of their web-based applications using web-based languages like JavaScript, CSS, and HTML that serve through mobile browsers yet provide a native app-like experience.

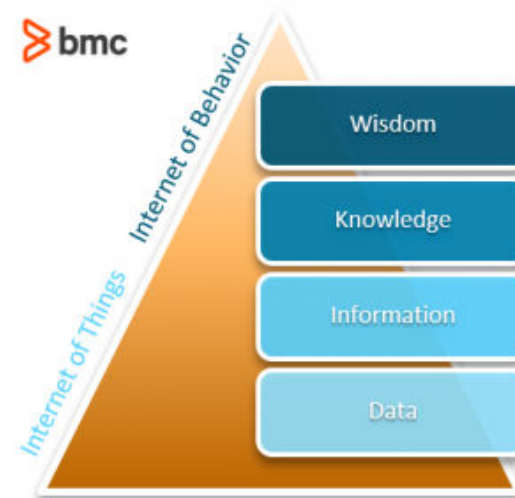
Who should use this option?

- This can be an excellent option for smaller development teams who want to attract a mobile user base without dedicating resources to mobile development.
- Enterprises can also leverage this option to provide a near-native user experience across multiple platforms via a streamlined web-based PIA.

## Growth in IoT

The [Internet of Things](#) is ever-expanding, with billions of smart devices powering many industries across the world, ranging from simple home appliances to medical devices. Thus, more and more software will be developed targeting these IoT devices. This will lead to an even tighter relationship between cloud and IoT technologies—after all, most IoT devices rely on distributed and cloud-based technologies across their lifecycle.

(See how IoT spurs the [internet of behaviors](#).)



## Increased emphasis on UI/UX

The increased reliance on digital services and experience has put a spotlight on user interfaces and user experience.

Traditionally, most software developments gave priority to the application logic and backend services without much consideration on UI or UX. This has changed rapidly, driven by increased demand for user-friendly and novel user experiences. In this way, UI/UX will come to the forefront of the software development process and might even lead to situations where UI/UX determines how application logic is implemented.

## Serverless computing

Cloud has fundamentally changed how software is deployed and delivered. Now, the new [serverless computing paradigm](#) is further evolving by eliminating any infrastructure management requirements and allowing developers to create solutions that can be directly deployed in these serverless environments. Some even support the direct deployment of containers all serverless. This will undoubtedly change how software is developed and should lead to even faster development lifecycles.

## Blockchain technology

[Blockchain](#) changed how we look at transaction recording. First popularized by digital currencies, blockchain quickly became popular in the finance sector as a decentralized digital ledger to record transactions securely. This decentralized nature and the ability to store any type of data make it one of the most secure technologies available. This has led to an explosion in blockchain technology being utilized in other sectors such as:

- Logistics
- Publishing
- Healthcare
- Etc.

More sectors, of course, means more software products using blockchain in some capacity. Services like Amazon-managed Blockchain allow leveraging the power of the cloud to create blockchain-based software solutions in software development.

## Cloud-based CI/CD

[Continuous Integration and Continuous Delivery](#) are the basis for automation in any SDLC. With the increased reliance on the cloud, cloud-based CI/CD solutions are quickly gaining popularity, even resulting in creating complete cloud-based software development pipelines. These managed CI/CD solutions will only grow in popularity as they will further reduce the management overhead and even lead to cost savings.

In the next few years, more and more software developments will be powered by cloud-based CI/CD tools utilizing other cloud-based solutions, such as:

- Code repositories
- Planning and management tools
- Test suites
- Etc.

## Multi-cloud architectures

Many organizations rely on a single cloud provider to power their applications. However, [multi-cloud architectures](#) are becoming increasingly popular—allowing

developers more freedom to mix and match different services from multiple providers to suit their exact needs without [being locked into](#) a single platform. Multi-cloud service providers like HashiCorp will become prominent players in this sector by offering tools to manage infrastructure, applications, and networking across cloud providers. All these things will change software developments from targeting a single platform to a more platform-agnostic development style.

## Rise of modern programming languages

Programming languages like [Python and Java](#) still dominate in the field of software development. However, some modern languages have emerged to challenge these existing goliaths by offering new features and native support for new technologies. They even provide solutions for existing issues, offering a better development experience for developers.

Languages like Rust, [Go](#), Kotlin, and TypeScript will become mainstream and may replace established languages like C/C++, Java in most use cases. Kotlin has already replaced Java in the mobile development landscape, while TypeScript is becoming favored in projects like Vue.js.

(Explore [popular programming languages](#).)

## Cloud-native application development

With the unstoppable growth in cloud services, almost all software will be using some kind of cloud service in the next decade, with most software explicitly designed for cloud-based environments. This way, cloud-native application development will become the standard practice in the future.

The software will be naturally evaluated to be deployed and managed in the cloud as fully online solutions due to most software development tools and services also moving to the cloud. In some sense, this has already happened, yet this trend will only intensify with more and more software transitions to cloud-native architectures.

## Trends in software development & AppDev

These trends will fundamentally change how software is developed and managed, and traditional software development methods will be relegated to history books.

But it's not just software development that's changing—the importance we place on software will only grow as it powers so many aspects of life in this increasingly connected world.