SOFTWARE ENGINEERING AND PROJECT MANAGEMENT

SUBJECT CODE: 22AIM51

FACULTY : K.S.SHASHIKALA
Senior Asst. Prof / AI&ML
NHCE

SOFTWARE ENGINEERING AND PROJECT MANAGEMENT

Course Code	**22AIM51**	**CIE Marks**	**50**
L:T:P:S	**3:0:0:0**	**SEE Marks**	**50**
Hrs / Week	**3**	**Total Marks**	**100**
Credits	**03**	**Exam Hours**	**03**

Course outcomes: At the end of the course, the student will be able to:

22AIM51.1	Understand the principles of software Engineering and project management involved in developing large projects.
22AIM51.2	Apply a comprehensive project plan and estimation strategies.
22AIM51.3	Illustrate the importance of Requirement Mangement process.
22AIM51.4	Design innovative software solutions, utilizing modeling and all architecture techniques.
22AIM51.5	Evalute the project progress and manage using different project & process metrics.
22AIM51.6	Create an appropriate risk management plan based on risk register.

Mapping of Course Outcomes to Program Outcomes and Program Specific Outcomes:

	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO 10**	**PO11**	**PO12**	**PSO1**	**PSO2**
22AIM51.1	2	-		-	-	-	-		-		-	-	-	-
22AIM51.2	3	-	-	-	-	-	-	-	-	-	-	3	2	-
22AIM51.3	-	3	-	-	-	-	-	-	-	-	-	-	2	-
22AIM51.4	-	-	3-	-	3							3	2	-
22AIM51.5	-	3	-	-	-	-	-	-	-	-	-	3	2	-
22AIM51.6	-		3	-	3	-	-	-	-	-	-	-	2	-

MODULE-1	**SOFTWARE ENGINEERING – INTRODUCTION**	22AIM51.1	**8 Hours**

Software Engineering – Definition, Software life cycle activities, Challenges in System Development, Software process models: Waterfall, Prototyping, spiral, and agile model, Software development methodology.

Case Study	Investigate the Challenges of System Development, Compare any two Modern software development paradigms
Text Book	Text Book 2: 1.1-1.16, 2.1 - 2.16

MODULE-2	**SYSTEM ENGINEERING**	22AIM51.3, 22AIM51.4	**8 Hours**

Requirement Engineering - Initiating the Requirements Engineering process, Eliciting Requirements, developing use cases, Building the analysis model, Software Requirement Document, System Architectural design, Subsystems development, System integration testing and deployment, System configuration management.

Case Study	Investigate Architectural design and compare any two testing techniques
Text Book	Text Book 2: 3.1 - 3.9, 4.1 - 4.9

MODULE-3	**DOMAIN MODELLING**	22AIM51.4	**8 Hours**

Definition, Object-Orientation and Class diagram - class and object, Object and Attribute, Association, Multiplicity of Role, Aggregation, Inheritance and Polymorphism, Visualizing domain model

Case Study	Explore the class diagram for any domestic application development, Explore tools for UML class diagram
Text Book	Text Book 2: 5.1- 5.10

MODULE-4	**MANAGING SOFTWARE PROJECT (MSP)**	22AIM51.2, 22AIM51.5	**8 Hours**

Project Management Concepts, Process and Project Metrics-Estimation for Software Projects, Decomposition Techniques, Empirical Estimation models, Project Scheduling- Maintenance and Reengineering.

Case Study	Numerical Problems and case studies on: 1.Basic Effort Estimation 2. Function Points Estimation 3. CoCoMo II Estimation 4. Cost Benefit Analysis 5. Agile Estimation
Text Book	Text Book 4: Ch:24-29

MODULE-5	**RISK MANAGEMENT**	22AIM51.5 22AIM51.6	**8 Hours**

Risk identification – Assessment – Risk Planning –Risk Management – – PERT technique – Monte Carlo simulation – Resource Allocation – Creation of critical paths – Cost schedules.

Case Study	**Numerical problems and case studies on:** 1. PERT/ CPM 2. Monte Carlo Simulation
Text Book	Text Book 1: 7.1-7.14,11.1-11.9

CIE Assessment Pattern (50 Marks – Theory)

	RBT Levels	**Test**	**Assessment(s) ***	**MCQ**
		25	**15**	**10**
L1	**Remember**	5		**5**
L2	**Understand**	5	-	**5**
L3	**Apply**	10	**5**	
L4	**Analyze**	5	**10**	
L5	**Evaluate**	-	-	
L6	**Create**	-	-	

*Assessments are to be selected from the assessment list attached to **Appendix A.**

SEE Assessment Pattern (50 Marks – Theory)

	RBT Levels	**Exam Marks Distribution (50)**
L1	**Remember**	10
L2	**Understand**	10
L3	**Apply**	20
L4	**Analyze**	10
L5	**Evaluate**	-
L6	**Create**	-

Suggested Learning Resources:

Text Books:

1. Software Project Management by Bob Hughes, Mike Cotterell and Rajib Mall, Fifth Edition, Tata McGraw Hill, New Delhi, 2017. ISBN: 9780077122799, 0077122798
2. Object Oriented Software Engineering, By David Kung, McGraw Hill, 2013, ISBN: 9781259080791, 125908079X.
3. Software Engineering by Ian Sommerville,9th edition, 2016, PearsonEdu. ISBN: 9780133943030, 0133943038
4. Software Engineering – A Practitioner's Approach by Roger S Pressman,7thedition, 2014, ISBN: 007769774X, 9780077697747

Reference Books:

1. "Software Project Management: A Unified Framework" by Walker Royce,1998. ISBN: 9780321734020.
2. Managing Global Software Projects McGraw Hill Education (India), Gopalaswamy Ramesh, Fourteenth Reprint 2013. ISBN: 9781283922418
3. Effective Software Project Management by Robert K. Wysocki – Wiley Publication, 2011. ISBN: 9780470446539
4.Software Project Management in Practice by Pankaj Jalote, 5thedition 2015. ISBN: 9789352868827, 935286882X

Web links and Video Lectures (e-Resources):

- https://onlinecourses.nptel.ac.in/noc20_cs68/preview
- https://onlinecourses.nptel.ac.in/noc19_cs70/preview

Activity-Based Learning (Suggested Activities in Class)/ Practical Based learning:

- Visit to any software development organization
- Contents related activities (Activity-based discussions)
  - ➢ For active participation of students, instruct the students to prepare Flowcharts and Handouts
  - ➢ Organizing Group wise discussions on issues
  - ➢ Seminars

Shashi KS

Module 1 – Software Engineering Introduction

- Software Engineering – Definition
- Software life cycle activities
- Challenges in System Development
- Software process models: Waterfall, Prototyping, Evolutionary, spiral, unified and agile model
- Software development methodology

Self-study / Case Study / Applications:

- Investigate the Challenges of System Development, Compare any two Modern software development paradigms

Learning resources

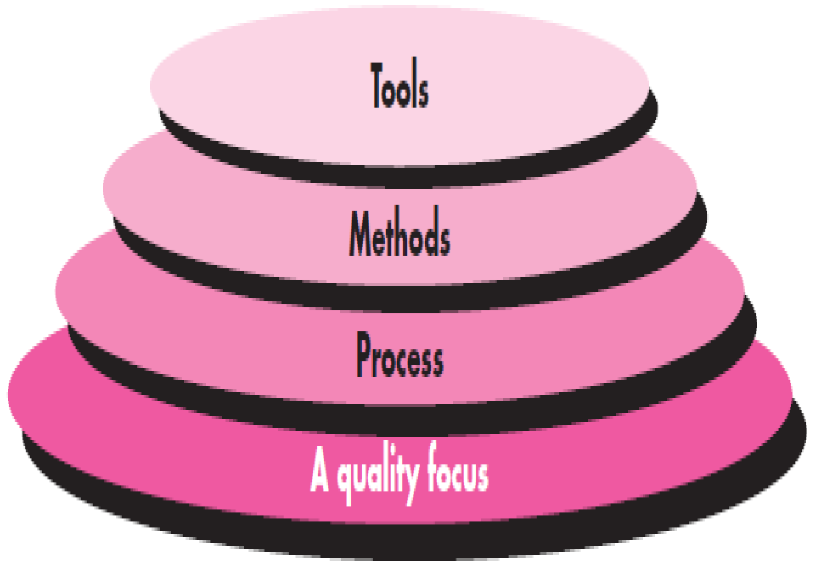
Shashi KS

- Text Books:
  1. Object Oriented Software Engineering, By David Kung edition 2018.
  2. Software Project Management by Bob Hughes, Mike Cotterell and Rajib Mall, Fifth Edition, Tata McGraw Hill, New Delhi, 2015.
  3. Software Engineering by Ian Sommerville,9th edition, 2012, Pearson Edu.
  4. Software Engineering – A Practitioner’s Approach by Roger S Pressman,7thedition, 2014,
  5. Software Project Management in Practice by Pankaj Jalote, 5thedition 2015
- Reference Books:
  1. Software Project Management: A Unified Framework” by Walker Royce.
  2. Managing Global Software Projects McGraw Hill Education (India), Gopalaswamy Ramesh, Fourteenth Reprint 2013.
  3. Effective Software Project Management by Robert K. Wysocki – Wiley Publication, 2011.

Introduction

Shashi KS

- Software Engineering is a discipline that involves the application of **systematic, disciplined, and quantifiable** approaches to the **development, operation, and maintenance of software systems.**
- Software engineering emphasizes the use of standardized processes, methodologies, tools, and techniques to **manage complexity, ensure reliability, facilitate collaboration among team members, and achieve efficiency in software development projects**
- It also addresses broader considerations such as project management, quality assurance, and ethical responsibilities in software development.



Introduction

Shashi KS

Why is Software Engineering required?

- To manage large software
- For more scalability
- Cost management
- To manage the dynamic nature of software
- For better quality management

Three parameters often drive and define a software project:

Cost:

- As the main cost of producing software is the manpower employed, the cost of developing software is generally measured in terms of person-months of effort spent in development. The productivity in the software industry for writing fresh code mostly ranges from a few hundred to about 1000 + LOC per person per month.
- **Schedule:**
- The schedule is another important factor in many projects. Business trends are dictating that the time to market a product should be reduced; that is, the cycle time from concept to delivery should be small. This means that software needs to be developed faster and within the specified time.

Quality:

- Quality is one of the main mantras, and business strategies are designed around it. Developing high-quality software is another fundamental goal of software engineering.

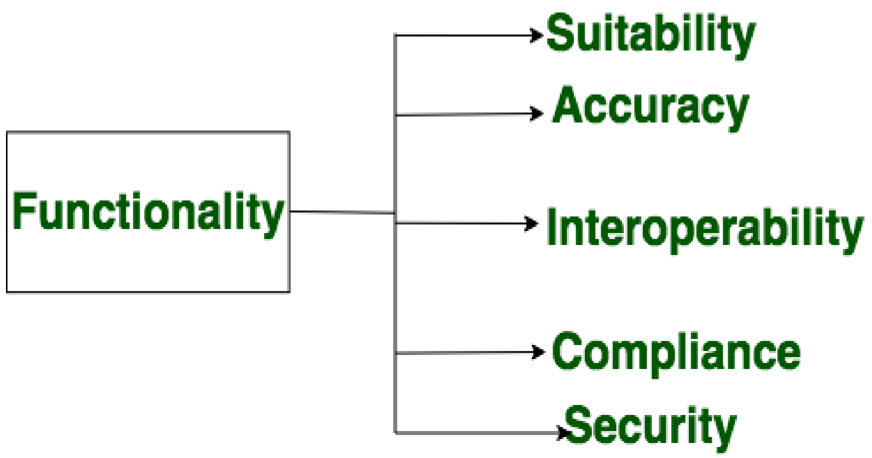
Functionality

Shashi KS

Functionality refers to the set of **features and capabilities** that a software program or system provides to its users. It is one of the most important characteristics of software, as it determines **the usefulness of the** software for the intended purpose.

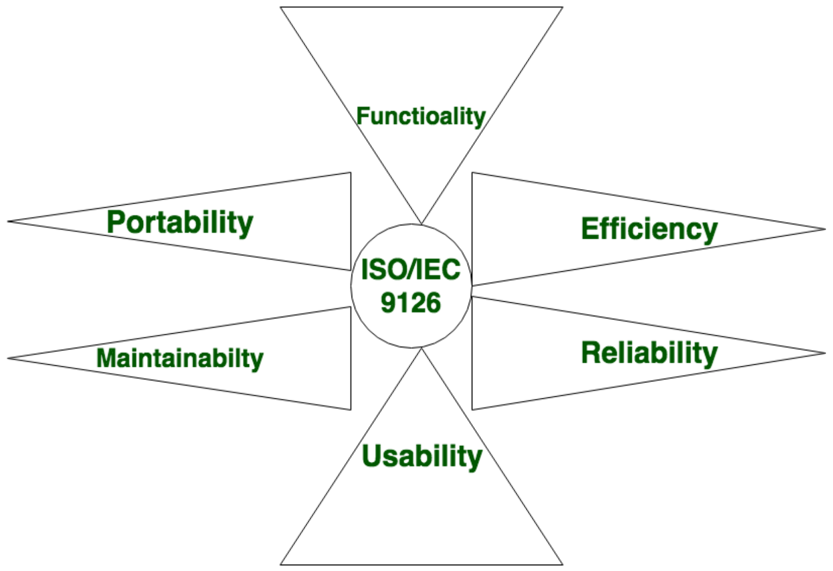
Examples of functionality in software include:

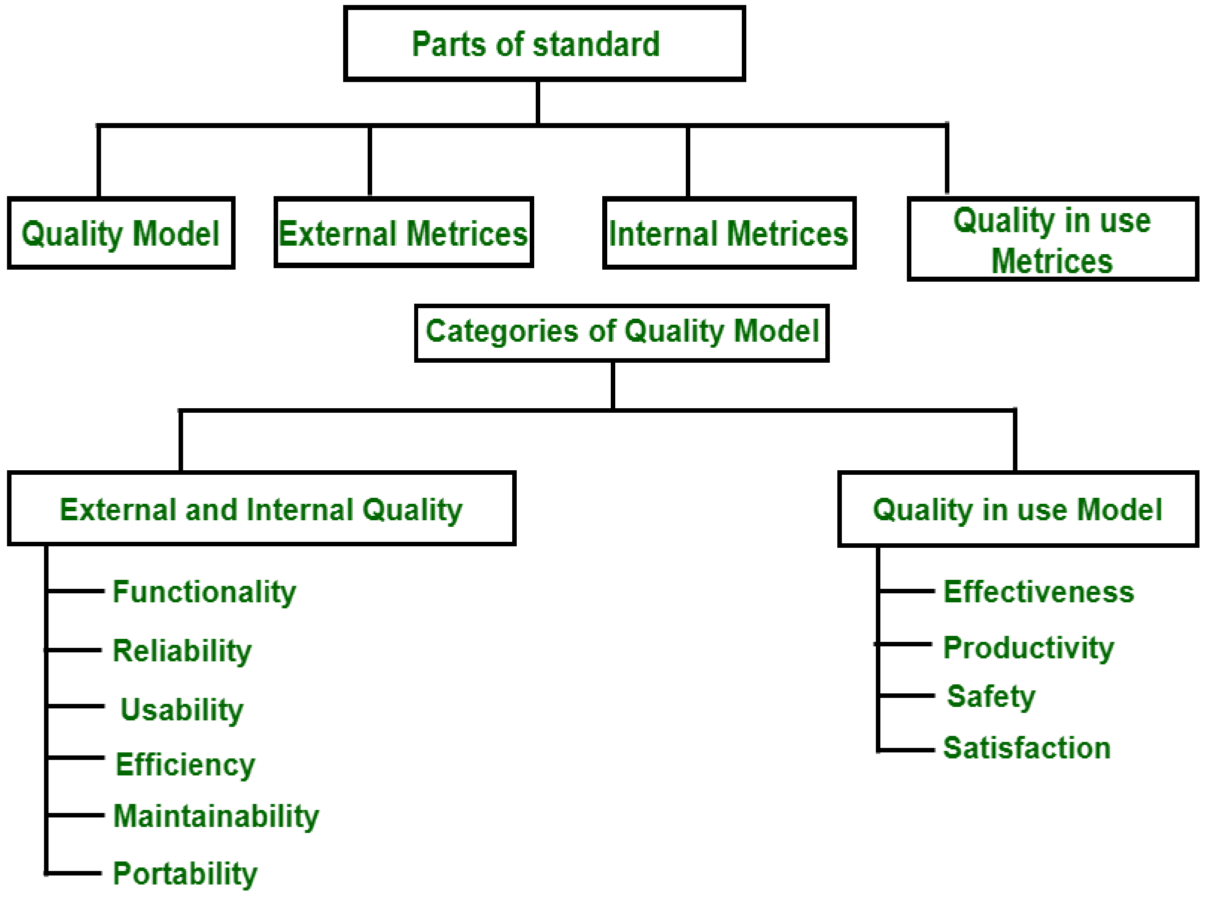
- Data storage and retrieval
- Data processing and manipulation
- User interface and navigation
- Communication and networking
- Security and access control
- Reporting and visualization
- Automation and scripting



Introduction – quality attributes

Shashi KS





ISO i.e. **International Organization for Standardization** and **IEC** i.e. **International Electrotechnical Commission** have developed ISO/IEC 9126 standards for software engineering → **Product Quality** to provide an all-inclusive specification and **evaluation model** for the quality of the software product.

Introduction

Shashi KS

- The international standard on software product quality suggests that six main attributes:

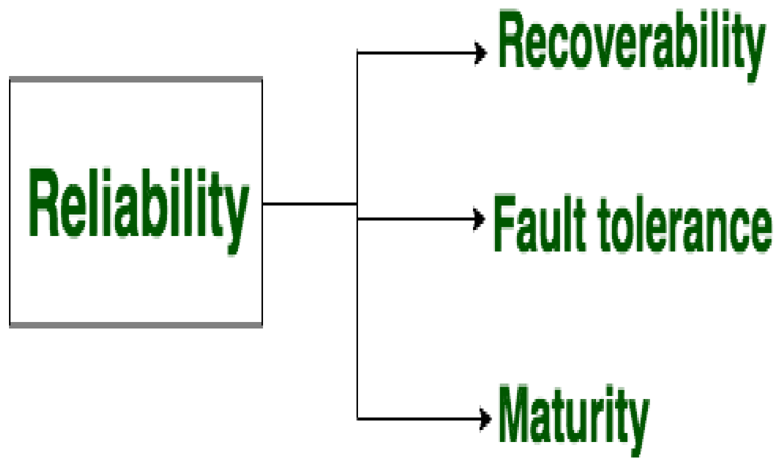
1. **Reliability:** The capability to provide failure-free service.
2. **Functionality:** The capability to provide functions that meet stated and implied needs when the software is used.
3. **Usability:** The capability to be understood, learned, and used.
4. **Efficiency:** The capability to provide appropriate performance relative to the amount of resources used.
5. **Maintainability:** the capability to be modified for purposes of making corrections, improvements, or adaptations.
6. **Portability:** The capability to be adapted for different specified environments without applying actions or means other than those provided for this purpose in the product.

Reliability

Shashi KS

- A set of attributes that bears on the capability of software to **maintain its level of performance under the given condition for a stated period of time.**
- Reliability is a characteristic of software that refers to its ability to **perform its intended functions correctly and consistently over time.** Reliability is an important aspect of software quality, as it helps ensure that the software will work correctly and not fail unexpectedly.

1. Bugs and errors in the code
2. Lack of testing and validation
3. Poorly designed algorithms and data structures
4. Inadequate error handling and recovery
5. Incompatibilities with other software or hardware



Efficiency

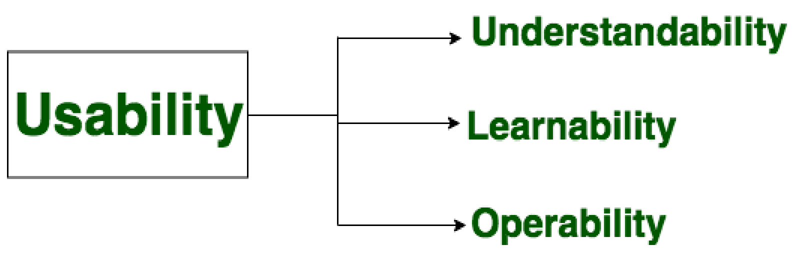
Shashi KS

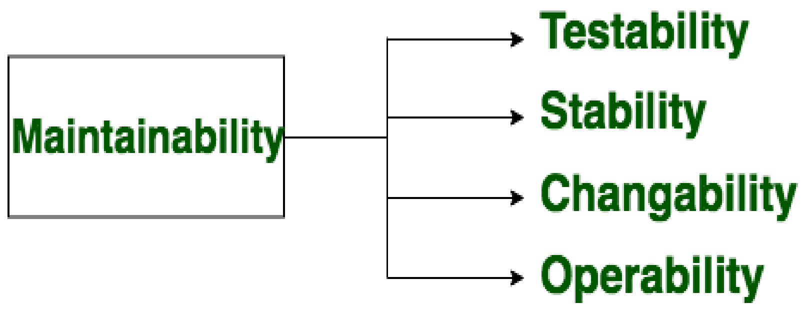
- It refers to the ability of the **software to use system resources** in the most effective and efficient manner. The software should make effective use of storage space and execute the commands as per desired timing requirements.
- Efficiency is a characteristic of software that refers to **its ability to use resources such as memory, processing power, and network bandwidth in an optimal way.**
- High efficiency means that a software program **can perform its intended functions quickly and with minimal use of resources, while low efficiency means that a software program may be slow or consume excessive resources.**
- Examples:
  1. Poorly designed algorithms and data structures
  2. Inefficient use of memory and processing power
  3. High network latency or bandwidth usage
  4. Unnecessary processing or computation
  5. Unoptimized code

Shashi KS

Usability & maintainability

- It refers to the extent to which the software can be used with ease.
- The amount of effort or time required to learn how to use the software.
- It refers to the ease with which **modifications can be made in a software system to extend its functionality, improve its performance, or correct errors.**





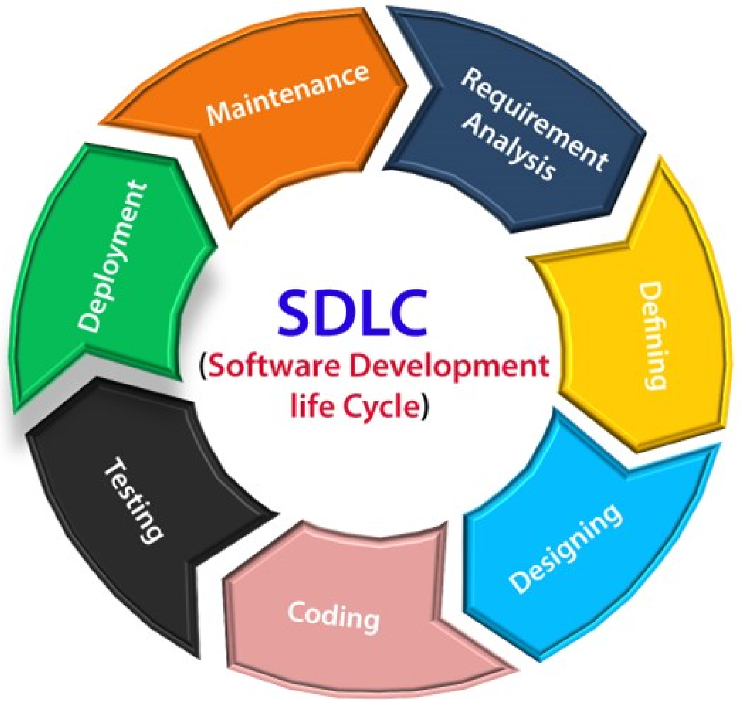
Software life cycle activities

Shashi KS

A **framework that describes the activities performed at each stage of a software development project.**

Stage 1: Planning and requirement analysis

- Inputs from all the **stakeholders and domain experts or SMEs in the industry**
- Planning for the **quality assurance requirements**
- Identifications of the **risks**
- Business analyst and Project organizer set up a meeting with the client to gather all the data like what the **customer wants to build, who will be the end user, what is the objective of the product**



Software life cycle

Shashi KS

Stage2:

Defining Requirements

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

Stage3: Designing the Software

- Bring down all the knowledge of requirements, analysis, and design of the software project.

Software life cycle

Shashi KS

Stage 4: Developing the project

- The implementation of design begins concerning writing code.
- Developers have to follow the **coding guidelines described by their management**

Stage 5: Testing

- After the code is generated, **it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage.**

- During this stage, **unit testing, integration testing, system testing, acceptance testing are done.**

Software life cycle

Shashi KS

Stage 6: Deployment

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

Stage 7: Maintenance

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

Challenges in software development

Shashi KS

Software project development can face numerous challenges, which can vary depending on factors such as **project size, complexity, team dynamics, and external constraints**. Some common challenges include:

1. **Requirements Management:** Gathering, understanding, and managing requirements from stakeholders can be challenging. Changes in requirements during the project lifecycle can affect **project timelines and budgets**.

2. **Communication Issues:**

Effective communication among team members, stakeholders, and across different departments or teams is crucial for project success. Poor communication can lead to **misunderstandings, delays, and conflicts.**

Challenges in software development

Shashi KS

3. Resource Constraints:

Limited availability of **skilled personnel, budget constraints, and tight deadlines can put pressure on project resources.**

Insufficient resources may affect the **quality of deliverables and increase the risk of project failure.**

4.Technical Complexity: Developing software solutions with **complex architectures, integrations, or technologies c**an pose technical challenges. Technical complexity may lead to design flaws, performance issues, and difficulties in implementation and maintenance.

5.Scope Creep: Changes in project scope or requirements after the project has started can lead to scope creep. Managing scope creep requires **careful planning, communication, and change control processes to prevent negative impacts on project objectives.**

Challenges in software development

Shashi KS

6. **Risk Management:** Identifying, assessing, and mitigating project risks is essential for minimizing the impact of unforeseen events or issues. Failure to manage risks effectively can lead to **project delays, budget overruns, or even project failure.**

7. **Quality Assurance:**

Ensuring the quality of software deliverables through **comprehensive testing and quality assurance processes** is challenging. Inadequate testing or QA practices can result in **defects, bugs, and user dissatisfaction.**

8. Project Management:

Effective project management is critical for **coordinating project activities, managing resources, and monitoring** progress. Poor project management practices can lead to missed deadlines, budget overruns, and project failure.

Challenges in software development

Shashi KS

9. **Stakeholder Management:** Engaging and managing stakeholders throughout the project lifecycle is essential for ensuring their support and satisfaction. **Failure to address stakeholder concerns or expectations can lead to conflicts and project delays.**

10. **Technology Changes: Rapid advancements in technology and evolving industry trends** can pose challenges for software projects. Keeping up with technology changes requires **continuous learning, adaptation, and sometimes reevaluation of project plans and strategies.**

Addressing these challenges **requires a combination of effective planning, communication, risk management, and collaboration among project stakeholders**. Adhering to best practices in software development and project management can help mitigate risks and improve the likelihood of project success.

SOFTWARE PROCESS

A process was defined as a collection of work activities, actions, and tasks that are performed when some work product is to be created. Each of these activities, actions, and tasks reside within a framework or model that defines their relationship with the process and with one another

A generic process framework for software engineering defines five framework activities—communication, planning, modeling, construction, and deployment. In addition, a set of umbrella activities—project tracking and control, risk management, quality assurance, configuration management, technical reviews, and others—are applied throughout the process.

A linear process flow executes each of the five framework activities in sequence, beginning with communication and culminating with deployment (Figure 2.2a). An iterative process flow repeats one or more of the activities before proceeding to the next (Figure 2.2b). An evolutionary process flow executes the activities in a “circular” manner. Each circuit through the five activities leads to a more complete version of the software (Figure 2.2c)

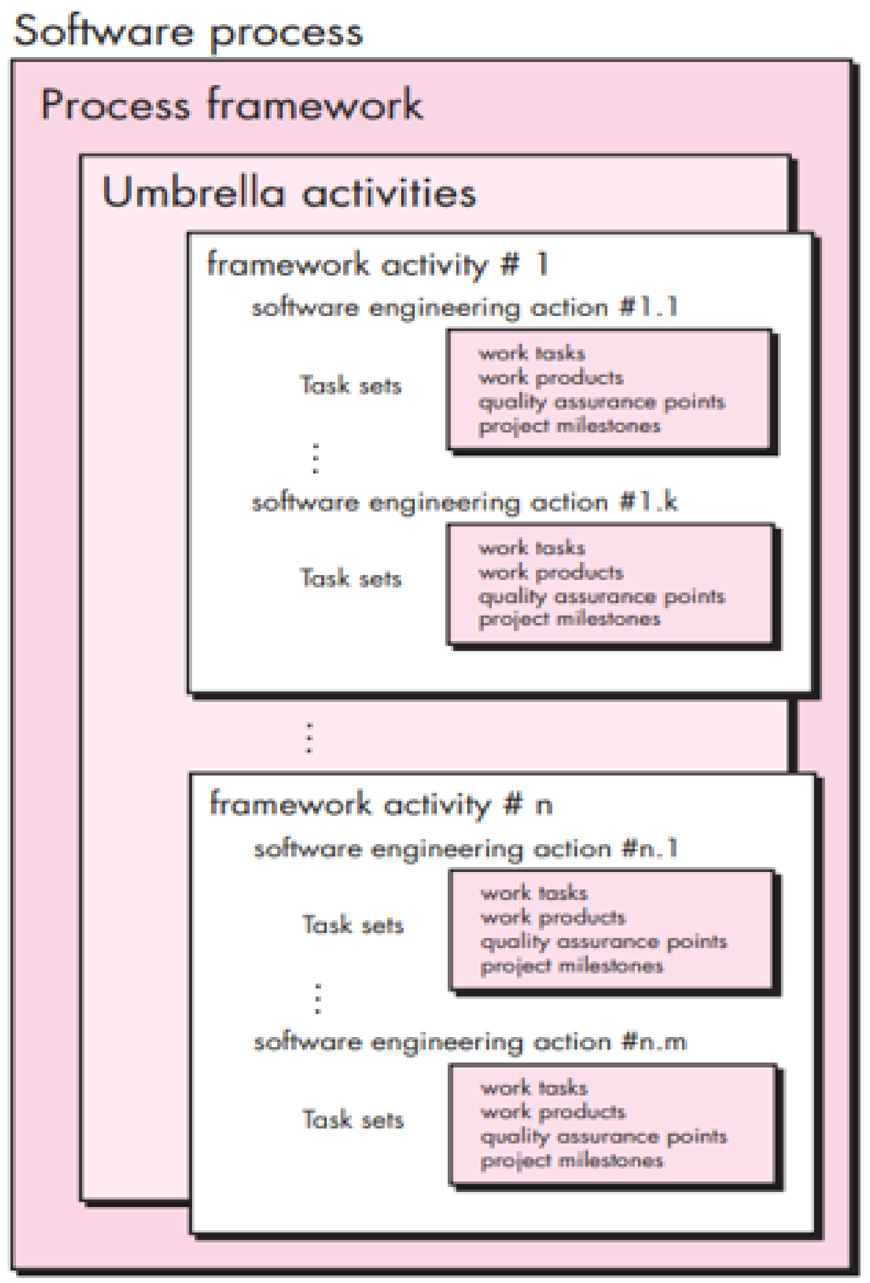
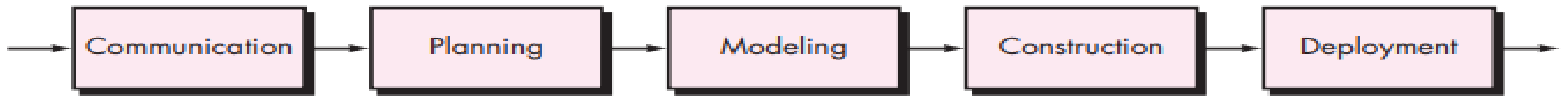
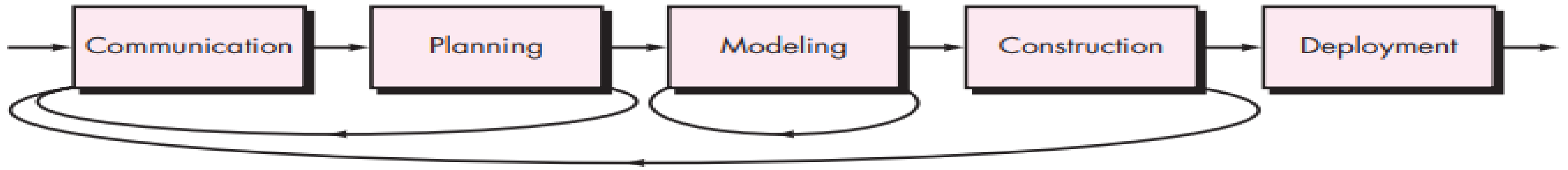


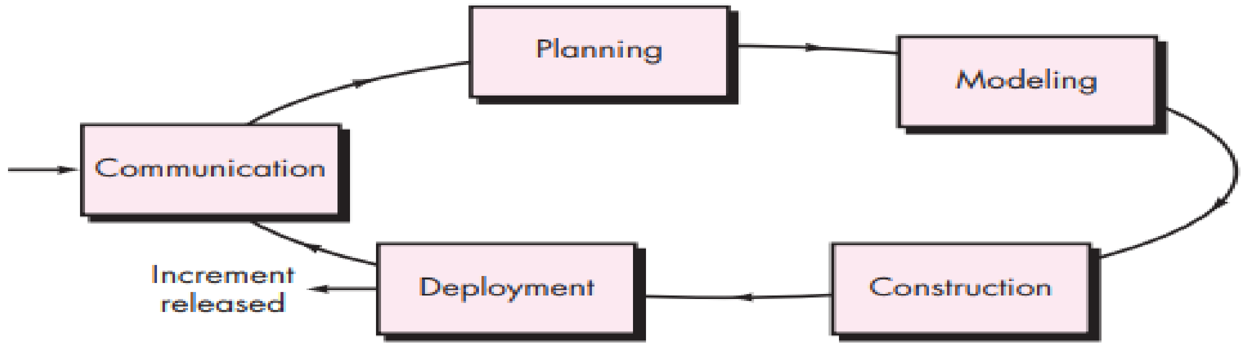
FIGURE 2.2 Process flow



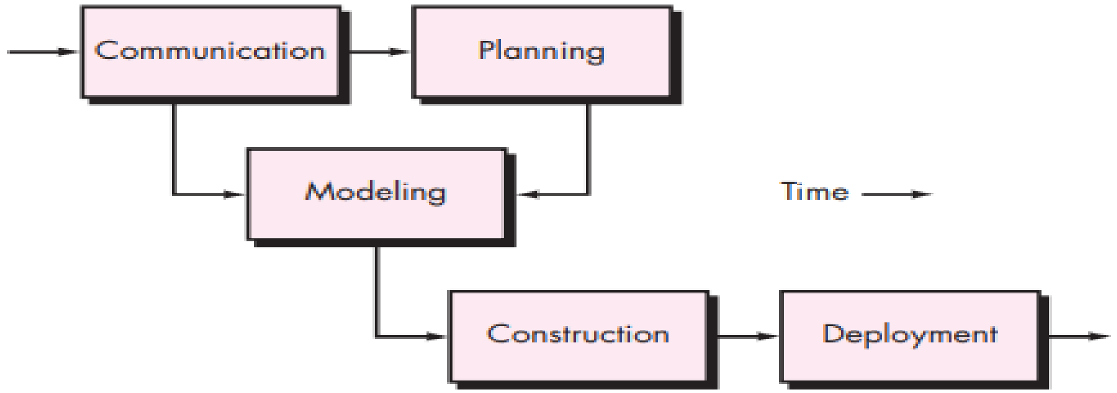
(a) Linear process flow



(b) Iterative process flow



(c) Evolutionary process flow



(d) Parallel process flow

The Waterfall Model

There are times when the requirements for a problem are well understood—when work flows from communication through deployment in a reasonably linear fashion.

This situation is sometimes encountered when well-defined adaptations or enhancements to an existing system must be made (e.g., an adaptation to accounting software that has been mandated because of changes to government regulations). It may also occur in a limited number of new development efforts, but only when requirements are well defined and reasonably stable.

The waterfall model, sometimes called the classic life cycle, suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment, culminating in ongoing support of the completed software (Figure 2.3).

Among the problems that are sometimes encountered when the waterfall model is applied are:

The principal stages of the waterfall model directly reflect the fundamental development activities:

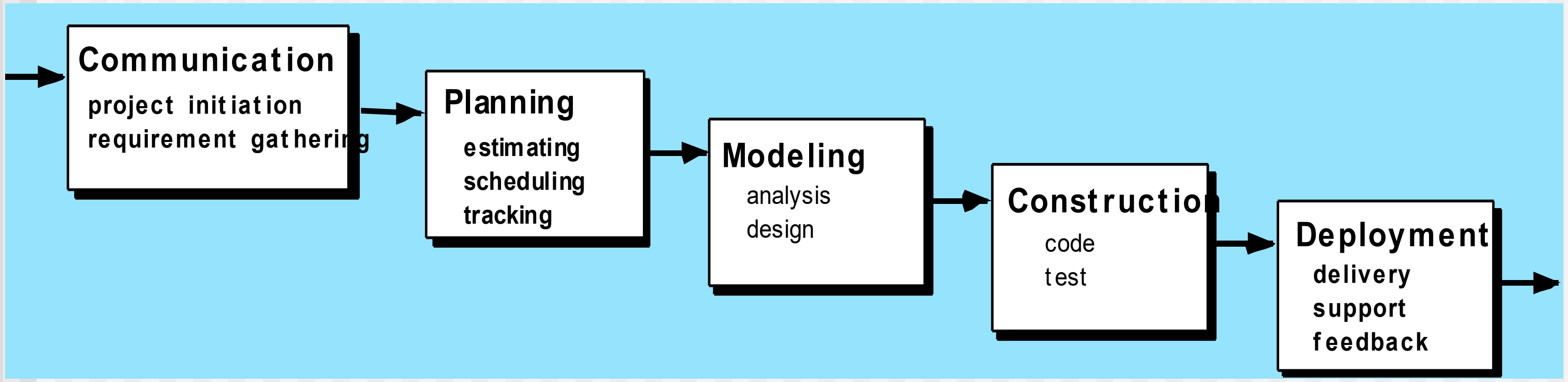
1. Requirements analysis and definition The system's services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.

2. System and software design
The systems design process allocates the requirements to either hardware or software systems by establishing an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.

3. Implementation and unit testing
During this stage, the software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.

4. Integration and system testing
The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.
5. Operation and maintenance
Normally (although not necessarily), this is the longest life cycle phase. The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life cycle, improving the implementation of system units and enhancing the system's services as new requirements are discovered.

The Waterfall Model



Software process models - Waterfall

Shashi KS

Advantages of the Waterfall model

- **Simplicity:** The linear and sequential nature of the Waterfall models makes it easy to understand and implement.
- **Clear Documentation:** Each phase has its own set of documentation, making it easier to track progress and manage the project.
- **Stable Requirements:** Well-suited for projects with stable and well-defined requirements at the beginning.
- **Predictability:** Due to its structured nature, the Waterfall model allows for better predictability in terms of timelines and deliverables.

Software process models - Waterfall

Shashi KS

Disadvantages of the Waterfall Model:

1. Real projects rarely follow the sequential flow that the model proposes. Although the linear model can accommodate iteration, it does so indirectly. As a result, changes can cause confusion as the project team proceeds.

2. It is often difficult for the customer to state all requirements explicitly. The waterfall model requires this and has difficulty accommodating the natural uncertainty that exists at the beginning of many projects.

3. The customer must have patience. A working version of the program(s) will not be available until late in the project time span. A major blunder, if undetected until the working program is reviewed, can be disastrous

Software process models - Waterfall

When to Use the Waterfall Models:

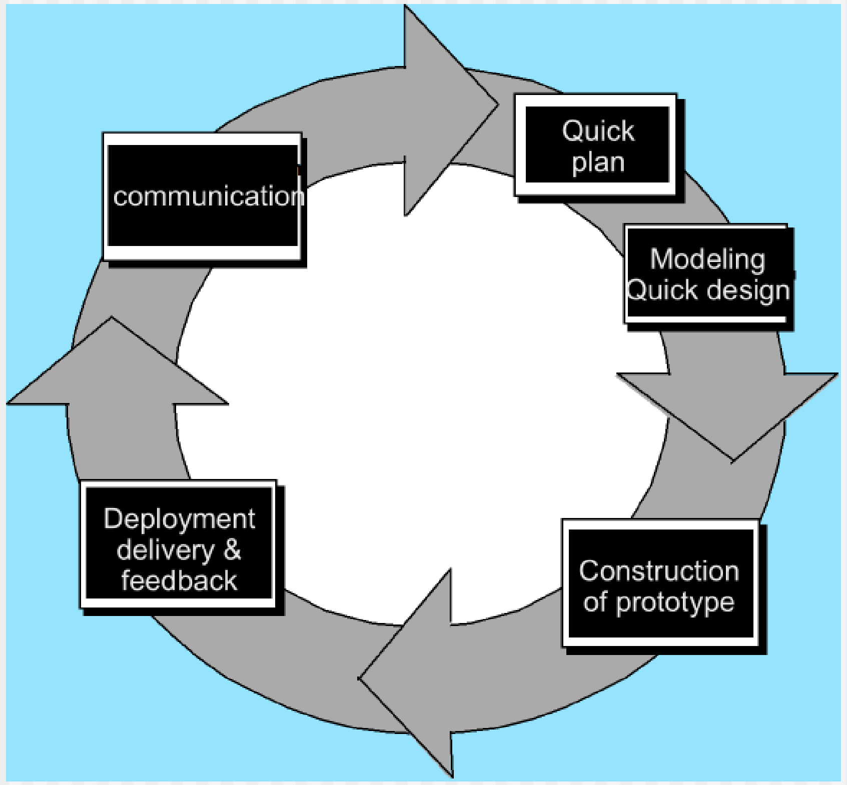
- **Well-Defined Requirements:** When project requirements are clear, stable, and unlikely to change significantly.
- **Small to Medium-Sized Projects:** For smaller projects with straightforward objectives and limited complexity.
- **Mission-Critical Systems:** In scenarios where it is crucial to have a well-documented and predictable development process, especially for mission-critical systems.

Prototyping model

Shashi KS

- Prototyping is defined as the process of developing a working replication of a product or system that has to be engineered.
- This model is used when the customers do not know the exact project requirements beforehand. In this model, a prototype of the end product is first developed, tested, and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product.

Evolutionary Models: Prototyping

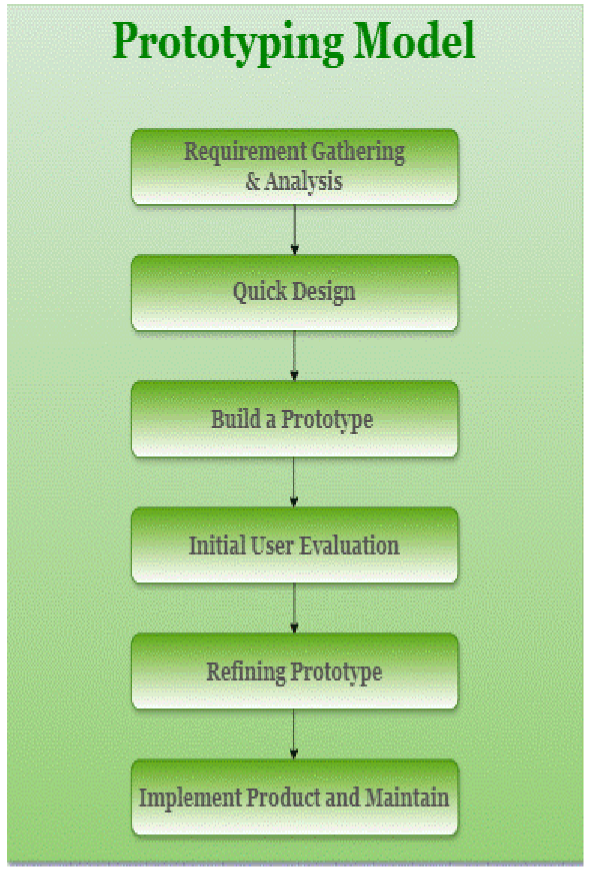


Prototyping model

Shashi KS

Steps of Prototyping Model

- **Step 1: Requirement Gathering and Analysis:** This is the initial step in designing a prototype model. In this phase, users are asked about what they expect or what they want from the system.
- **Step 2: Quick Design:** This is the second step in the Prototyping Model. This model covers the basic design of the requirement through which a quick overview can be easily described.
- **Step 3: Build a Prototype:** This step helps in building an actual prototype from the knowledge gained from prototype design.
- **Step 4: Initial User Evaluation:** This step describes the preliminary testing where the investigation of the performance model occurs, as the customer will tell the strengths and weaknesses of the design, which was sent to the developer.
- **Step 5: Refining Prototype:** If any feedback is given by the user, then improving the client's response to feedback and suggestions, the final system is approved.
- **Step 6: Implement Product and Maintain:** This is the final step in the phase of the Prototyping Model where the final system is tested and distributed to production, here the program is run regularly to prevent failures.



Prototyping model

Shashi KS

Types of Prototyping Models

There are four types of Prototyping Models, which are described below.

- Rapid Throwaway Prototyping
- Evolutionary Prototyping
- Incremental Prototyping
- Extreme Prototyping

Rapid Throwaway Prototyping

- This technique offers a useful method of exploring ideas and getting customer feedback for each of them.
- In this method, a developed prototype need not necessarily be a part of the accepted prototype.
- Customer feedback helps prevent unnecessary design faults and hence, the final prototype developed is of better quality.

Prototyping model

Shashi KS

Evolutionary Prototyping

- In this method, the prototype developed initially is incrementally refined based on customer feedback till it finally gets accepted.
- In comparison to Rapid Throwaway Prototyping, it offers a better approach that saves time as well as effort.
- This is because developing a prototype from scratch for every iteration of the process can sometimes be very frustrating for the developers.

Incremental Prototyping

- In this type of incremental prototyping, the final expected product is broken into different small pieces of prototypes and developed individually.
- In the end, when all individual pieces are properly developed, then the different prototypes are collectively merged into a single final product in their predefined order.

Incremental Prototyping

- It's a very efficient approach that reduces the complexity of the development process, where the goal is divided into sub-parts and each sub-part is developed individually.
- The time interval between the project's beginning and final delivery is substantially reduced because all parts of the system are prototyped and tested simultaneously.
- Of course, there might be the possibility that the pieces just do not fit together due to some lack of ness in the development phase – this can only be fixed by careful and complete plotting of the entire system before prototyping starts.

Prototyping model

Shashi KS

Extreme Prototyping

- This method is mainly used for web development. It consists of three sequential independent phases:
- In this phase, a basic prototype with all the existing static pages is presented in HTML format.
- In the 2nd phase, Functional screens are made with a simulated data process using a prototype services layer.
- This is the final step where all the services are implemented and associated with the final prototype.

Advantages of Prototyping Model

- The customers get to see the partial product early in the life cycle. This ensures a greater level of customer satisfaction and comfort
- New requirements can be easily accommodated as there is scope for refinement
- Missing functionalities can be easily figured out
- Errors can be detected much earlier thereby saving a lot of effort and cost, besides enhancing the quality of the software.
- The developed prototype can be reused by the developer for more complicated projects in the future.
- Flexibility in design
- Early feedback from customers and stakeholders can help guide the development process

Prototyping model - advantages

Shashi KS

- The developed prototype can be reused by the developer for more complicated projects in the future.
- Flexibility in design
- Early feedback from customers and stakeholders can help guide the development process and ensure that the final product meets their needs and expectations.
- Prototyping can be used to test and validate design decisions, allowing for adjustments to be made before significant resources are invested in development.
- Prototyping can help reduce the risk of project failure by identifying potential issues and addressing them early in the process.

Prototyping model

Disadvantages of the Prototyping Model

- Costly concerning time as well as money.
- There may be too much variation in requirements each time the prototype is evaluated by the customer.
- Poor Documentation due to continuously changing customer requirements.
- It is very difficult for developers to accommodate all the changes demanded by the customer.
- There is uncertainty in determining the number of iterations that would be required before the prototype is finally accepted by the customer.

Shashi KS

Disadvantages of the Prototyping Model

- After seeing an early prototype, the customers sometimes demand the actual product to be delivered soon.
- Developers in a hurry to build prototypes may end up with sub-optimal solutions.
- The customer might lose interest in the product if he/she is not satisfied with the initial prototype.
- The prototype may not be scalable to meet the future needs of the customer.
- The prototype may not accurately represent the final product due to limited functionality or incomplete features.
- The focus on prototype development may shift away from the final product, leading to delays in the development process.

Prototyping model

Shashi KS

Applications of Prototyping Model

- The Prototyping Model should be used when the requirements of the product are not clearly understood or are unstable.
- The prototyping model can also be used if requirements are changing quickly.
- This model can be successfully used for developing user interfaces, high-technology software-intensive systems, and systems with complex algorithms and interfaces.
- The prototyping Model is also a very good choice to demonstrate the technical feasibility of the product.

Evolutionary Model

Shashi KS

- The concept of the evolutionary model in software engineering came into picture when the users wanted to experience the partially developed system rather than waiting for the fully developed version.
- The evolutionary model is based on the idea of developing the core modules, then gradually improving the software product over time using **incremental** and **iterative** techniques with appropriate feedback.
- In this type of process model, successive versions of the product are made through several iterations and come up when the final product is built through multiple iterations.
- The evolutionary approach suggests breaking down modules into maintainable smaller chunks, prioritizing them and then delivering those chunks one at a time to the users.

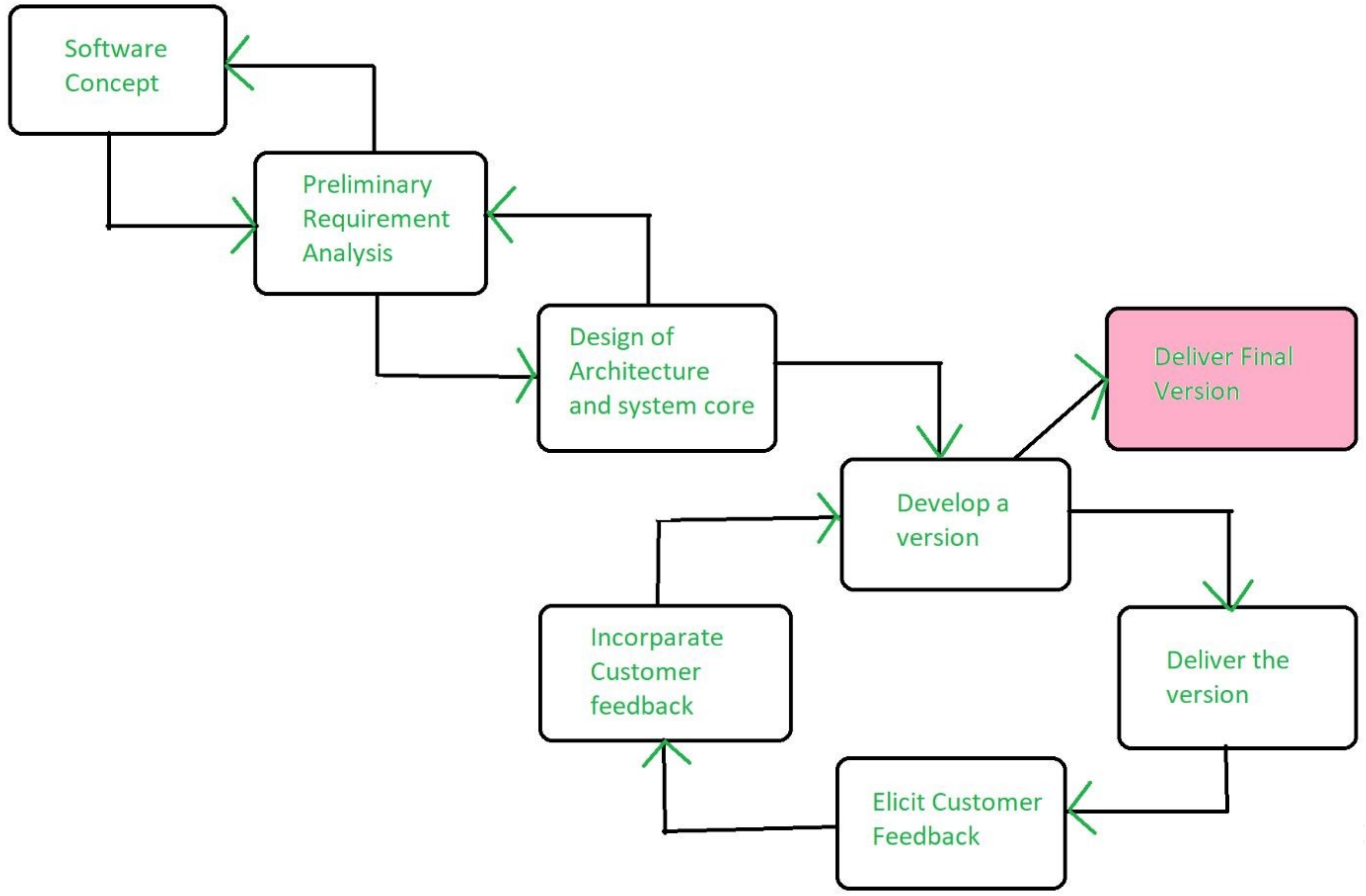
Evolutionary Model - characteristics

Shashi KS

- Each evolutionary model may be developed using an iterative waterfall model of development.
- Evolutionary models include three types- Iterative model, Incremental model and Spiral model.
- The models require several primary needs and architectural planning that must be done.
- The new release of the product may include new functionality and also a modification in the existing functionality.
- This model allows for changing requirements as well as dividing the development process into different manageable work modules.
- The development team responds to customer feedback on the product throughout the planning phase of the next cycle, frequently by altering the product, strategy, or process.

Evolutionary Model

Shashi KS


Software Concept
Preliminary Requirement Analysis
Design of Architecture and system core
Develop a version
Deliver Final Version
Deliver the version
Elicit Customer Feedback
Incorparate Customer feedback

Evolutionary Model - characteristics

Shashi KS

Advantages Evolutionary Model

- **Adaptability to Changing Requirements:** Evolutionary models work effectively in projects when the requirements are ambiguous or change often. They support adjustments and flexibility along the course of development.
- **Early and Gradual Distribution:** Functional components or prototypes can be delivered early thanks to incremental development. Faster user satisfaction and feedback may result from this.
- **User Commentary and Involvement:** Evolutionary models place a strong emphasis on ongoing user input and participation. This guarantees that the software offered closely matches the needs and expectations of the user.
- **Improved Handling of Difficult Projects:** Big, complex tasks can be effectively managed with the help of evolutionary models. The development process is made simpler by segmenting the project into smaller, easier-to-manage portions.

Evolutionary Model - characteristics

Shashi KS

Disadvantages Evolutionary Model

- **Communication Difficulties:** Evolutionary models require constant cooperation and communication. The strategy may be less effective if there are gaps in communication or if team members are spread out geographically.
- **Dependence on an Expert Group:** A knowledgeable and experienced group that can quickly adjust to changes is needed for evolutionary models. Teams lacking experience may find it difficult to handle these model's dynamic nature.
- **Increasing Management Complexity:** Complexity can be introduced by organizing and managing several increments or iterations, particularly in large projects. In order to guarantee integration and synchronization, good project management is needed.
- **Greater Initial Expenditure:** As evolutionary models necessitate continual testing, user feedback and prototyping, they may come with a greater starting cost. This may be a problem for projects that have limited funding.

Spiral model

Shashi KS

- The spiral model is an SDLC model that combines elements of an iterative software development model with a waterfall model. It is advisable to use this model for expensive, large and complex projects.
- In the spiral model, each loop may not result in deliverable software. In the incremental model, every increment actually leads to a deployable increment at the customer site, one of the major differences between the spiral and incremental models.
- In the prototyping model, the risks which can be identified upfront can be handled, but in the spiral model, the risks which appear after the development start can be handled better.

Spiral model phases

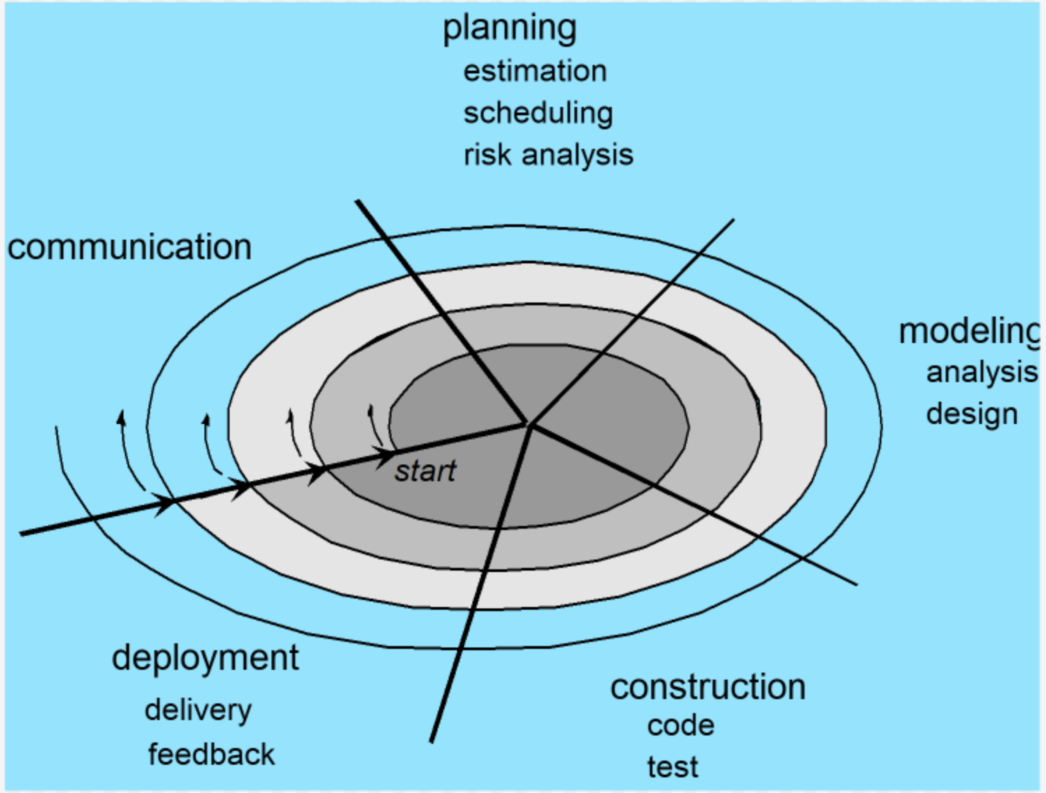
Shashi KS

The Spiral Model is a **risk-driven model**, meaning that the focus is on managing risk through multiple iterations of the software development process.

It has four stages or phases: The planning of objectives, risk analysis, engineering or development, and finally review. A project passes through all these stages repeatedly and the phases are known as a Spiral in the model.

1. **Determine objectives and find alternate solutions** – This phase includes requirement gathering and analysis. Based on the requirements, objectives are defined and different alternate solutions are proposed.
2. **Risk Analysis and resolving** – In this quadrant, all the proposed solutions are analyzed and any potential risk is identified, analyzed, and resolved.

Evolutionary Models: The Spiral



These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill, 2009). Slides copyright 2009 by Roger Pressman.

Spiral model phases

Shashi KS

3. **Develop and test:** This phase includes the actual implementation of the different features. All the implemented features are then verified with thorough testing.
4. **Review and planning of the next phase** – In this phase, the software is evaluated by the customer. It also includes risk identification and monitoring like cost overrun or schedule slippage and after that planning of the next phase is started.

- The Spiral model is called a Meta-Model because it subsumes all the other SDLC models. For example, a single loop spiral actually represents the Iterative Waterfall Model.

1. The spiral model incorporates the stepwise approach of the Classical Waterfall Model.
2. The spiral model uses the approach of the Prototyping Model by building a prototype at the start of each phase as a risk-handling technique.
3. Also, the spiral model can be considered as supporting the Evolutionary model – the iterations along the spiral can be considered as evolutionary levels through which the complete system is built.

Spiral model - advantages

Shashi KS

1. **Risk Handling:** The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.
2. **Good for large projects:** It is recommended to use the Spiral Model in large and complex projects.
3. **Flexibility in Requirements:** Change requests in the Requirements at a later phase can be incorporated accurately by using this model.
4. **Customer Satisfaction:** Customers can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product.
5. **Iterative and Incremental Approach:** The Spiral Model provides an iterative and incremental approach to software development, allowing for flexibility and adaptability in response to changing requirements or unexpected events.

Spiral model - advantages

Shashi KS

6. **Emphasis on Risk Management:** The Spiral Model places a strong emphasis on risk management, which helps to minimize the impact of uncertainty and risk on the software development process.
7. **Improved Communication:** The Spiral Model provides for regular evaluations and reviews, which can improve communication between the customer and the development team.
8. **Improved Quality:** The Spiral Model allows for multiple iterations of the software development process, which can result in improved software quality and reliability.

Spiral model - disadvantages

Shashi KS

1. **Complex:** The Spiral Model is much more complex than other SDLC models.
2. **Expensive:** Spiral Model is not suitable for small projects as it is expensive.
3. **Too much dependability on Risk Analysis:** The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced experts, it is going to be a failure to develop a project using this model.
4. **Difficulty in time management:** As the number of phases is unknown at the start of the project, time estimation is very difficult.
5. **Complexity:** The Spiral Model can be complex, as it involves multiple iterations of the software development process.
6. **Time-Consuming:** The Spiral Model can be time-consuming, as it requires multiple evaluations and reviews.
7. **Resource Intensive:** The Spiral Model can be resource-intensive, as it requires a significant investment in planning, risk analysis, and evaluations.

Spiral model - disadvantages

Shashi KS

When To Use the Spiral Model?

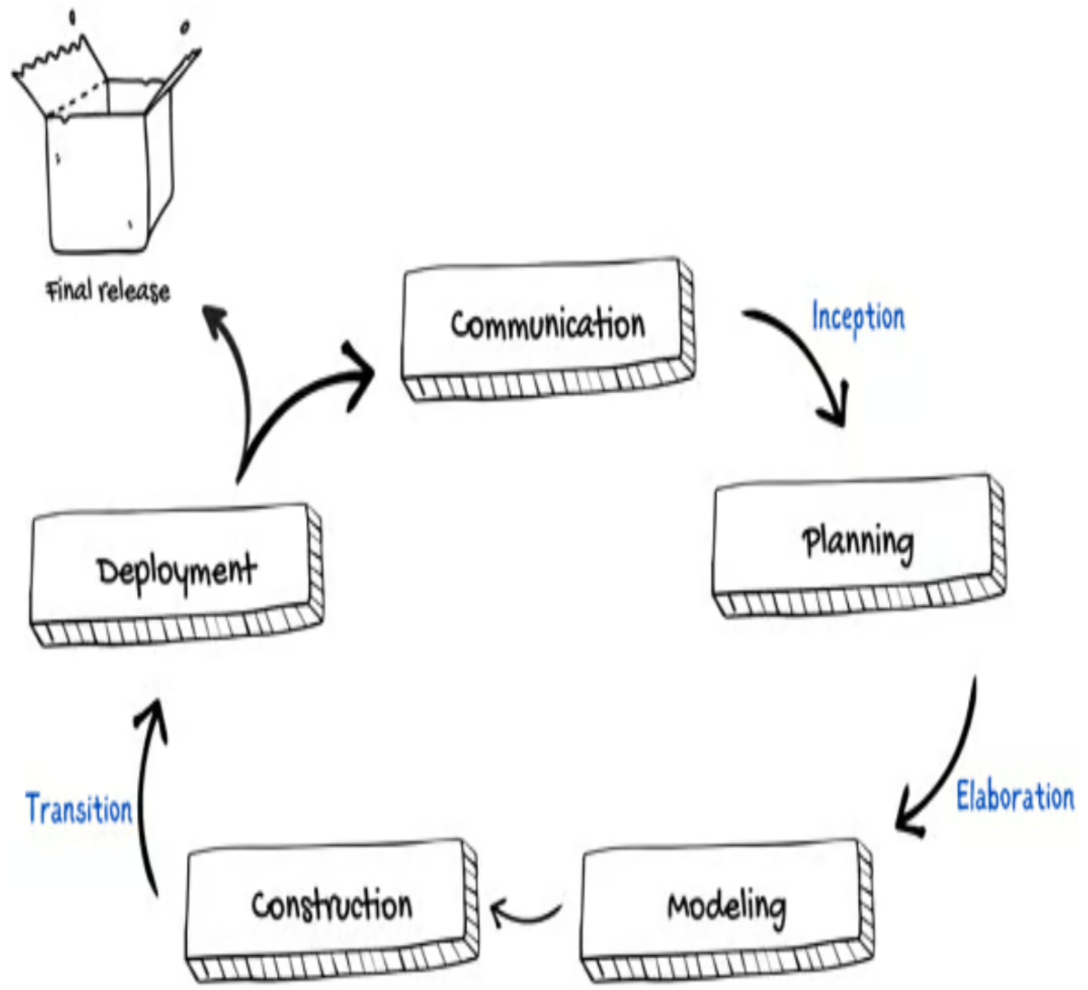
1. When a project is vast in software engineering, a spiral model is utilized.
2. A spiral approach is utilized when frequent releases are necessary.
3. When it is appropriate to create a prototype
4. When evaluating risks and costs is crucial
5. The spiral approach is beneficial for projects with moderate to high risk.
6. The SDLC’s spiral model is helpful when requirements are complicated and ambiguous.
7. If modifications are possible at any moment
8. When committing to a long-term project is impractical owing to shifting economic priorities.

Unified model

Shashi KS

The **Unified Process (UP)** is a software development framework used for object-oriented modeling. The framework is also known as Rational Unified Process (RUP) and the Open Unified Process (Open UP). Some of the key features of this process include:

- It defines the order of phases.
- It is component-based, meaning a software system is built as a set of software components. There must be well-defined interfaces between the components for smooth communication.
- It follows an iterative, incremental, architecture-centric, and use-case driven approach



Unified model

Shashi KS

The case-driven approach

- Use a case-driven approach that follows a set of actions performed by one or more entities.
- A use case refers to the process of the team performing the development work from the functional requirements.
- The functional requirements are made from the list of requirements that were specified by the client.
- For example, an online learning management system can be specified in terms of use cases such as "add a course," "delete a course," "pay fees," and so on.

The architecture-centric approach

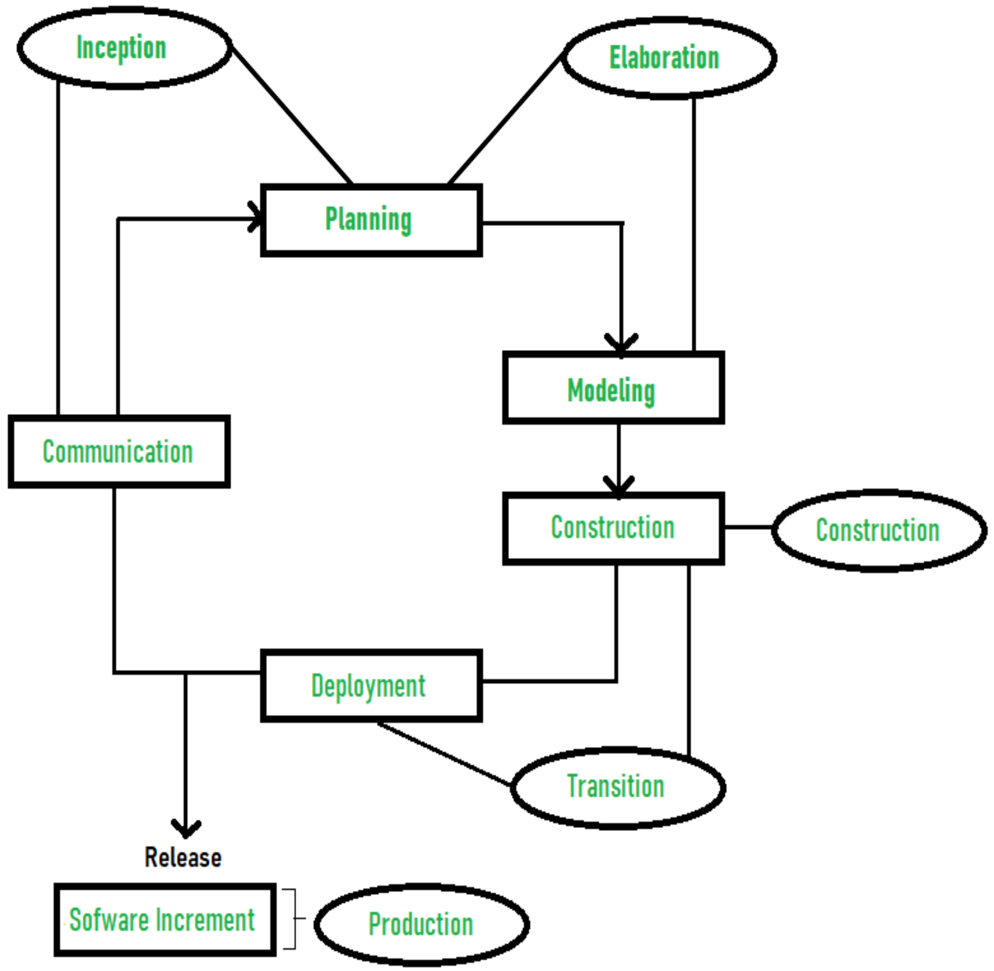
- The architecture-centric approach defines the form of the system and how it should be structured to provide a specific functionality whereas the use case defines the functionality.

Unified model - Phases

Shashi KS

We can represent a unified process model as a series of cycles. Each cycle ends with the release of a new system version for the customers. We have four phases in every cycle:

- Inception
- Elaboration
- Construction
- Transition



Unified model - Phases

Shashi KS

Inception

- Communication and planning are the main ones.
- Identifies the scope of the project using a use-case model allowing managers to estimate costs and time required.
- Customers’ requirements are identified and then it becomes easy to make a plan for the project.
- The project plan, Project goal, risks, use-case model, and Project description, are made.
- The project is checked against the milestone criteria and if it couldn’t pass these criteria then the project can be either canceled or redesigned.

Elaboration

- Planning and modeling are the main ones.
- A detailed evaluation and development plan is carried out and diminishes the risks.
- Revise or redefine the use-case model (approx. 80%), business case, and risks.
- Again, checked against milestone criteria and if it couldn’t pass these criteria then again project can be canceled or redesigned.
- Executable architecture baseline.

Unified model - Phases

Construction

- The project is developed and completed.
- System or source code is created and then testing is done.
- Coding takes place.

Transition

- The final project is released to the public.
- Transit the project from development into production.
- Update project documentation.
- Beta testing is conducted.
- Defects are removed from the project based on feedback from the public.

Shashi KS

Advantages:

1. It provides good documentation, it completes the process in itself.
2. It provides risk-management support.
3. It reuses the components, and hence total time duration is less.
4. Good online support is available in the form of tutorials and training.

Disadvantages:

1. Team of expert professional is required, as the process is complex.
2. Complex and not properly organized process.
3. More dependency on risk management.
4. Hard to integrate again and again.

Agile model

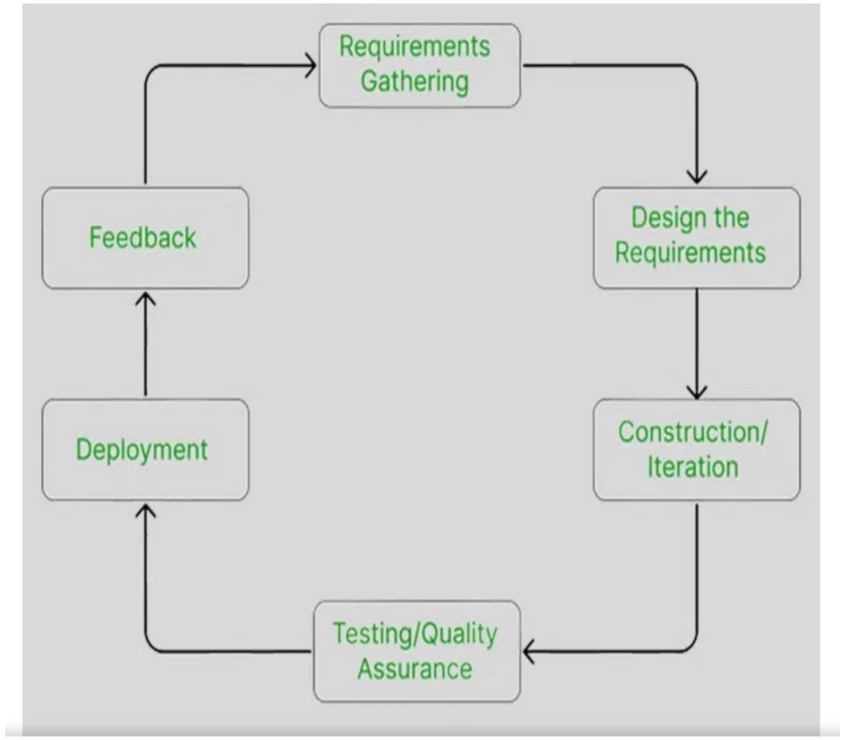
Shashi KS

- "**Agile process model**" refers to a software development approach based on iterative development.
- Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning.
- The project scope and requirements are laid down at the beginning of the development process.
- Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.
- Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks.
- The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements.
- Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

Agile model

Shashi KS

- **Requirement Gathering:-** In this step, the development team must gather the requirements, by interaction with the customer. development team should plan the time and effort needed to build the project. Based on this information you can evaluate technical and economical feasibility.
- **Design the Requirements:-** In this step, the development team will use user-flow-diagram or high-level UML diagrams to show the working of the new features and show how they will apply to the existing software. Wireframing and designing user interfaces are done in this phase.
- **Construction / Iteration:-** In this step, development team members start working on their project, which aims to deploy a working



Agile model

Shashi KS

- **Testing / Quality Assurance:-**
- Testing involves Unit Testing, Integration Testing, and System Testing**.**
  1. **Unit Testing:-** Unit testing is the process of checking small pieces of code to ensure that the individual parts of a program work properly on their own. Unit testing is used to test individual blocks (units) of code
  2. **Integration Testing:-** Integration testing is used to identify and resolve any issues that may arise when different units of the software are combined.
  3. **System Testing:-** Goal is to ensure that the software meets the requirements of the users and that it works correctly in all possible scenarios.

- **Deployment:-** In this step, the development team will deploy the working project to end users.

- **Feedback:-** This is the last step of the **Agile Model.** In this, the team receives feedback about the product and works on correcting bugs based on feedback provided by the customer

Agile model - Principles

Shashi KS

- To establish close contact with the customer during development and to gain a clear understanding of various requirements, each Agile project usually includes a customer representative on the team. At the end of each iteration stakeholders and the customer representative review, the progress made and re-evaluate the requirements.
- The agile model relies on working software deployment rather than comprehensive documentation.
- Frequent delivery of incremental versions of the software to the customer representative in intervals of a few weeks.
- Requirement change requests from the customer are encouraged and efficiently incorporated.
- It emphasizes having efficient team members and enhancing communications among them is given more importance. It is realized that improved communication among the development team members can be achieved through face-to-face communication rather than through the exchange of formal documents.
- It is recommended that the development team size should be kept small (5 to 9 people) to help the team members meaningfully engage in face-to-face communication and have a collaborative work environment.

Agile model - Characteristics

Shashi KS

- Agile processes must be adaptable to technical and environmental changes. That means if any technological changes occur, then the agile process must accommodate them.
- The development of agile processes must be incremental. That means, in each development, the increment should contain some functionality that can be tested and verified by the customer.
- The customer feedback must be used to create the next increment of the process.
- The software increment must be delivered in a short span of time.
- It must be iterative so that each increment can be evaluated regularly.

Agile model - Advantages

Shashi KS

- Working through Pair programming produces well-written compact programs which have fewer errors as compared to programmers working alone.
- It reduces the total development time of the whole project.
- Agile development emphasizes face-to-face communication among team members, leading to better collaboration and understanding of project goals.
- Customer representatives get the idea of updated software products after each iteration. So, it is easy for him to change any requirement if needed.
- Agile development puts the customer at the center of the development process, ensuring that the end product meets their needs.

Agile model - disadvantages

Shashi KS

- The lack of formal documents creates confusion and important decisions taken during different phases can be misinterpreted at any time by different team members.
- It is not suitable for handling complex dependencies.
- The agile model depends highly on customer interactions so if the customer is not clear, then the development team can be driven in the wrong direction.
- Agile development models often involve working in short sprints, which can make it difficult to plan and forecast project timelines and deliverables. This can lead to delays in the project and can make it difficult to accurately estimate the costs and resources needed for the project.
- Agile development models require a high degree of expertise from team members, as they need to be able to adapt to changing requirements and work in an iterative environment. This can be challenging for teams that are not experienced in agile development practices and can lead to delays and difficulties in the project.
- Due to the absence of proper documentation, when the project completes and the developers are assigned to another project, maintenance of the developed project can become a problem.

Software development methodology

Shashi KS

There are multiple types of Software development methodologies available and we will learn the following development methodologies.

- Lean Development
- Scrum Methodology
- Behavior Driven Development
- Feature Driven Development
- Extreme Programming (XP)

Lean development

Shashi KS

- Lean principles (Toyota Production System) got their start in manufacturing, as a way to optimize the production line to minimize waste and maximize value to the customer. These two goals are also relevant to software development, which also:
  - Follows a repeatable process
  - Requires particular quality standards
  - Relies on the collaboration of a group of specialized workers
- Applying Lean principles to knowledge work requires a shift in mindset in terms of how value, waste, and other key Lean concepts are defined.
- The seven Lean principles are: 1) Eliminate waste 2) Build quality in 3) Create knowledge, 4) Defer commitment, 5) Deliver fast, 6) Respect people, 7) Optimize the whole

Lean development

Shashi KS

Eliminate waste:

- **Unnecessary code or functionality:** Delays time to customer, slows down feedback loops
- **Starting more than can be completed:** Adds unnecessary complexity to the system, results in context-switching, handoff delays, and other impediments to flow
- **Delay in the software development process:** Delays time to customer, slows down feedback loops
- **Unclear or constantly changing requirements:** Results in rework, frustration, quality issues, lack of focus
- **Bureaucracy:** Delays speed
- **Slow or ineffective communication:** Results in delays, frustrations, and poor communication to stakeholders which can impact IT's reputation in the organization
- **Partially done work:** Does not add value to the customer or allow team to learn from work
- **Defects and quality issues:** Results in rework, abandoned work, and poor customer satisfaction
- **Task switching:** Results in poor work quality, delays, communication breakdowns, and low team morale

Lean development

Shashi KS

Build quality in:

In trying to ensure quality, many teams actually create waste – through excessive testing, for example, or an excessive logging of defects.

Most popular Lean development tools:

- **Pair programming:** Avoid quality issues by combining the skills and experience of two developers instead of one
- **Test-driven development:** Writing criteria for code before writing the code to ensure it meets business requirements
- **Incremental development and constant feedback**
- **Minimize wait states:** Reduce context switching, knowledge gaps, and lack of focus
- **Automation:** Automate any tedious, manual process or any process prone to human error

Lean development

Shashi KS

Create knowledge

This principle encourages Lean teams to provide the infrastructure to properly document and retain valuable learning. This can be done by using any combination of the following tools:

- Pair programming
- Code reviews
- Documentation
- Wiki – to let the knowledge base build up incrementally
- Thoroughly commented code
- Knowledge sharing sessions
- Training
- Use tools to manage requirements or user stories

Lean development

Shashi KS

Defer commitment

This Lean principle encourages team to demonstrate responsibility by keeping their options open and continuously collecting information, rather than making decisions without the necessary data.

To defer commitment means to:

- Not plan (in excessive detail) for months in advance
- Not commit to ideas or projects without a full understanding of the business requirements
- Constantly be collecting and analyzing information regarding any important decisions

Lean development

Shashi KS

Deliver fast

- Thinking too far in advance about future requirements
- Blockers that aren’t responded to with urgency
- Over-engineering solutions and business requirements

Lean development is based on this concept: Build a simple solution, put it in front of customers, enhance incrementally based on customer feedback.

Respect people

- Communicating proactively and effectively
- Encouraging healthy conflict
- Surfacing any work-related issues as a team
- Empowering each other to do their best work

Lean development

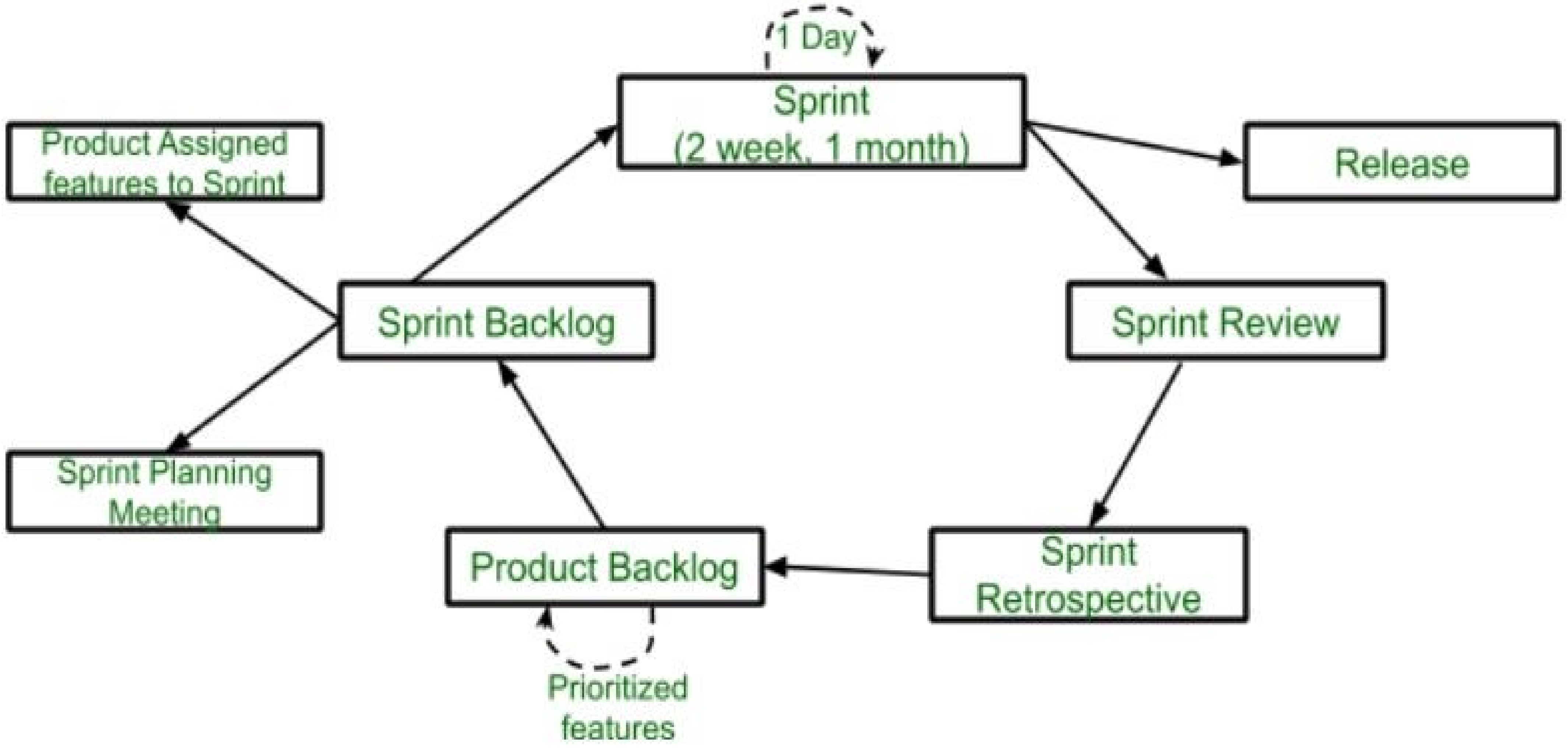
Shashi KS

Optimize the whole

- Suboptimization is a serious issue in software development
- The first is releasing sloppy code for the sake of speed. When developers feel pressured to deliver at all costs, they release code that may or may not meet quality requirements. This increases the complexity of the code base, resulting in more defects. With more defects, there is more work to do, putting more pressure on developers to deliver quickly… so the cycle continues.
- The second is an issue with testing. When testers are overloaded, it creates a long cycle time between when developers write code and when testers are able to give feedback on it.

Scrum Methodology

Shashi KS



Scrum Methodology

Shashi KS

- **Scrum** is the type of **Agile framework**. It is a framework within which people can address complex adaptive problem while productivity and creativity of delivering product is at highest possible values.
- Scrum uses **Iterative process**.

Features of Scrum

- Scrum is light-weighted framework
- Scrum emphasizes self-organization
- Scrum is simple to understand
- Scrum framework help the team to work together

Scrum Methodology

Shashi KS

Sprint: A Sprint is a time box of one month or less. A new Sprint starts immediately after the completion of the previous Sprint.

Release: When the product is completed, it goes to the Release stage.

Sprint Review: If the product still has some non-achievable features, it will be checked in this stage and then passed to the Sprint Retrospective stage.

Sprint Retrospective: In this stage quality or status of the product is checked.

Product Backlog: According to the prioritize features the product is organized.

Sprint Backlog: Sprint Backlog is divided into two parts Product assigned features to sprint and Sprint planning meeting.

Scrum Methodology

Shashi KS

Sprint Review Meeting:

A sprint review Meeting in Software Development is a scrum ceremony in which the development team highlights the work completed during the sprint, demonstrates it to stakeholders, and answers any questions they may have while examining the development.

Stakeholders offer insightful feedback that helps the team adjust the product backlog in response to shifting priorities, requirements, or new information.

Stakeholders provide valuable feedback and start working together to address any necessary improvements. A careful review of the acceptance criteria is part of the meeting to make sure they are in line with the user stories.

Purpose:

Collaborative Conversation: The gathering encourages honest and cooperative conversations between the stakeholders and the Scrum Team

Scrum Methodology

Shashi KS

Continuous Improvement: Processes are improved, collaboration strengthens and future development initiatives are guided by the insights gathered from stakeholder talks and feedback

Display of Completed Work: To give stakeholders a sense of visible progress and functionality, the development team delivers a possibly shippable product increment

Feedback from Stakeholders: The Sprint Review gives end users and the Product Owner an opportunity to discuss the features that have been demonstrated. In order to make sure that the supplied product meets the needs and expectations of stakeholders, this input is essential

Transparency and Inspection: The meeting promotes transparency and allows stakeholders to view the work produced during the sprint, giving them insight into the team's development and the current status of the product

Scrum Methodology

Shashi KS

Parameters	Sprint Review	Sprint Retrospective
Objectives	Its objective is to examine the amount of work that was finished during the sprint and modify the Product Backlog considering all of the remarks that were made during the review.	The objective is to consistently improve the team's collaboration, procedures and general efficiency.
Who Attends?	It involves the appropriate parties including clients and end users, as well as the Scrum Team, which consists of the Product Owner, Scrum Master and members of the development team.	Only the members of the development team, the scrum master, and the product owner are involved.
Focus	It's main focus is to showcase the features and functionality created during the sprint, gather feedback and modify the Product Backlog as necessary.	Its main focus is on talking about what worked, what could be better, and what needs to be done in the upcoming sprint.

Scrum Methodology

Shashi KS

Parameters	Sprint Review	Sprint Retrospective
Time Duration	At the conclusion of every sprint, there is a timed event that lasts between two and four hours, depending on how long the sprint is.	It is a scheduled event with a maximum duration of one to one and a half hours.

Behaviour driven development

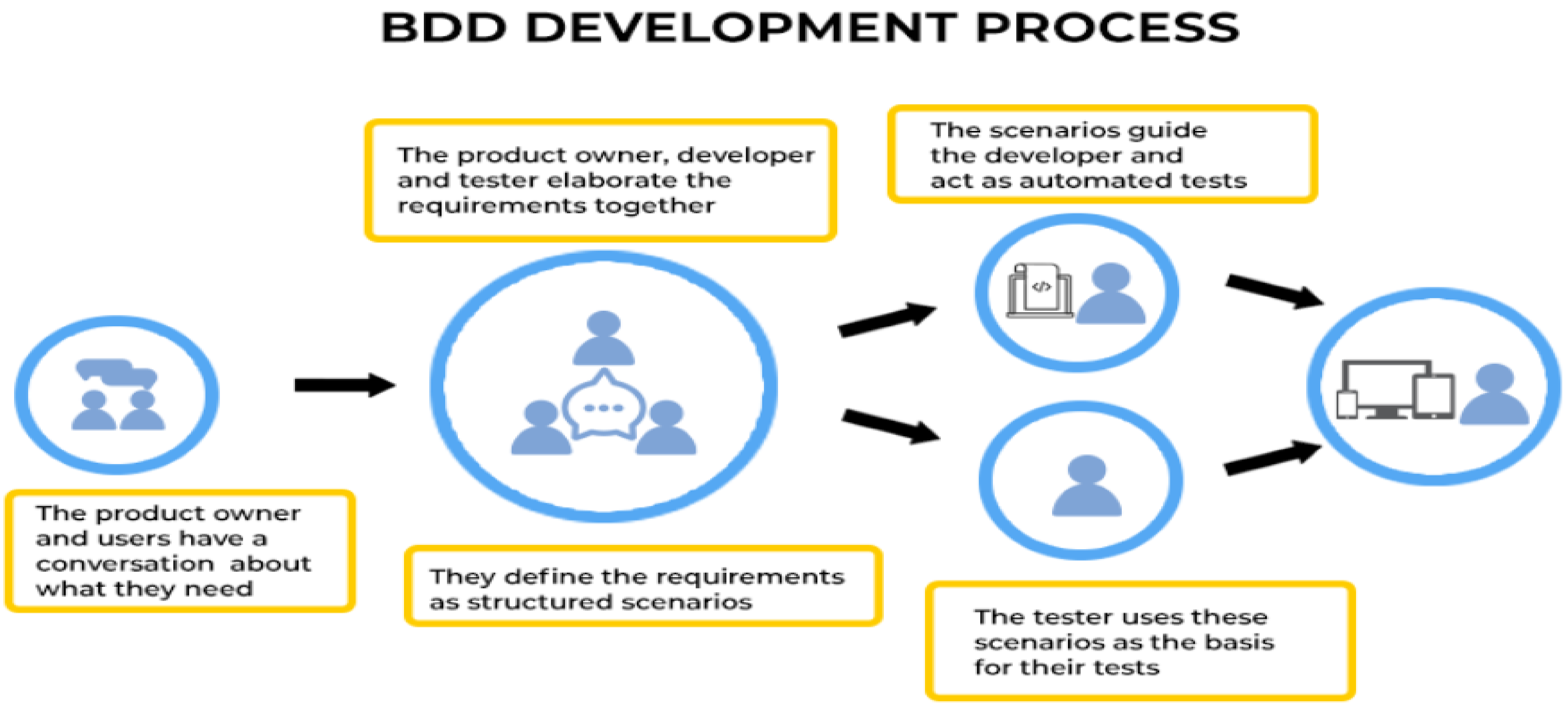
Shashi KS

BDD (behavior-driven development) is a software development process based on the Agile methodology. The concept of behavior-driven development originated from the test-driven development (TDD) approach. This software development process is focused on end-user requirements and their interactions with the product.

The concept behind BDD is to enable developers, testers, and business stakeholders to clearly understand an application’s behavior and ensure collaboration between technical and non-technical teams.

Behaviour driven development

Shashi KS


BDD DEVELOPMENT PROCESS
The product owner and users have a conversation about what they need
The product owner, developer and tester elaborate the requirements together
They define the requirements as structured scenarios
The scenarios guide the developer and act as automated tests
The tester uses these scenarios as the basis for their tests

Behaviour driven development

Shashi KS

Advantages:

1. Customer driven product development – Early feedback, customer involvement
2. Proper prioritization of features - core product features, prioritize them, and segregate them as per requirements
3. Greater transparency - greater visibility of the system features so that all the developers, testers, business stakeholders, and other team members can clearly understand what is being developed
4. Reduced maintenance and project risk
5. Better alignment of overall business goal - As this development approach emphasizes the behavior of each feature, details about what purpose they serve, how well it fits in the current market, how it can drive more profits, etc., can be conveyed better.

Behaviour driven development process

The behavior-driven development process is based on three critical approaches:

1. Test first approach
2. Agile testing approach
3. Built-in quality approach

Test first approach

The test-first approach is focused on testing the product at the right time.

It requires the agility of the development, testing, and project teams as well as clarity regarding the technical process, principles, and methods adopted during various project development phases.

Agile testing approach

Agile testing is based on the famous iterative development and testing methodology where particular emphasis is put on the collaboration between customers and teams. Customers' perspectives are considered during the development and testing process.

Behaviour driven development process

Built-in quality approach

Behavior-driven development adheres to a built-in quality approach, which ensures that the end product has all the required features and solutions and fits the current market. It is not enough to have a set of features and solutions; but it should be market-driven and customer-centric.

BDD mainly involves three phases: discovery, formulation, and automation.

Discovery phase

The discovery phase is the first phase of behavior-driven development, where the acceptance criteria are researched and decided. Generally, the product manager actively participates in behavior-driven development’s discovery phase.

In this stage, the acceptance criteria are crafted based on product type, key features, target audience, present market, and other factors. Other team members, such as the project manager, developer, tester, or operation team, contribute with their input. Overall, the discovery phase of behavior-driven development requires the collaborative efforts of team members to come up with the acceptance criteria.

Behaviour driven development process

Shashi KS

Formulation phase

The formulation phase is mainly relevant when a backlog item is just about to get implemented.

The primary purpose of this phase is to ensure that the acceptance criteria are confirmed and ready to be applied in real-time. Acceptance tests in the formulation phase execute this part.

The purpose of acceptance testing is to run quality assurance methods to determine whether an application or product is as per the requirements.

Automation phase

Automates the process of acceptance tests. It optimizes the time taken and resources used while ensuring that the required outputs are obtained, and criteria are matched.

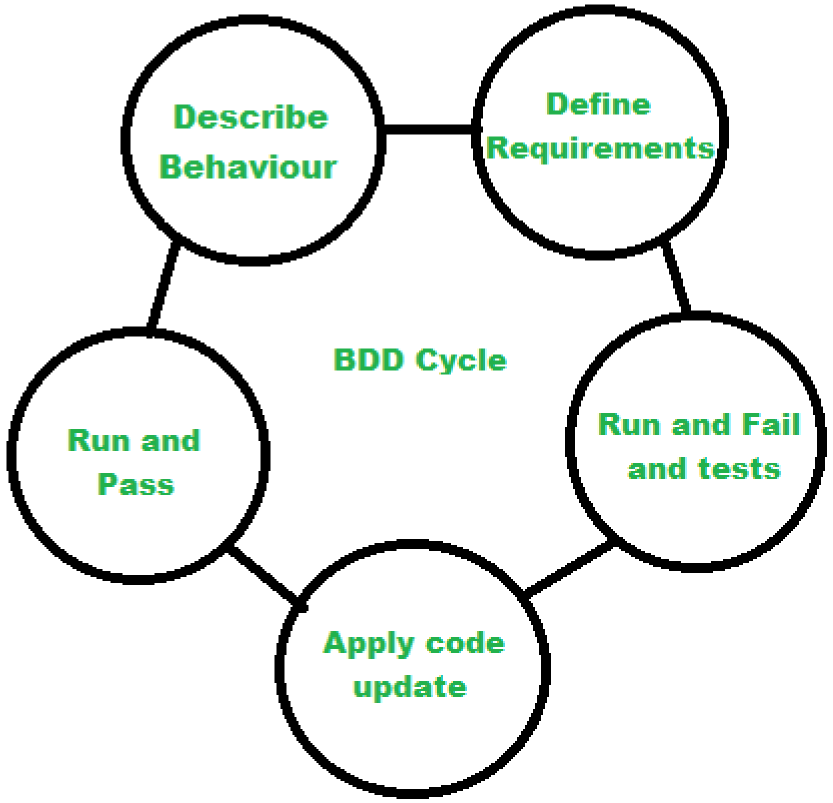
Behaviour driven development process

Shashi KS

BDD Life Cycle

1. **Describe behavior –**This includes the flow and features of the product means the main vision.
2. **Define requirements –**Modeled requirements with business rules for a shared understanding.
3. **Run and fail the tests –**Develop and run the test cases.
4. **Apply code update –**Refactor it according to the requirement.
5. **Run and pass the tests –**Run the updated code and pass the test cases.

But the important point is that BDD is not about testing like TDD. BDD is all about achieving business goals and requirements.



Behaviour driven development process

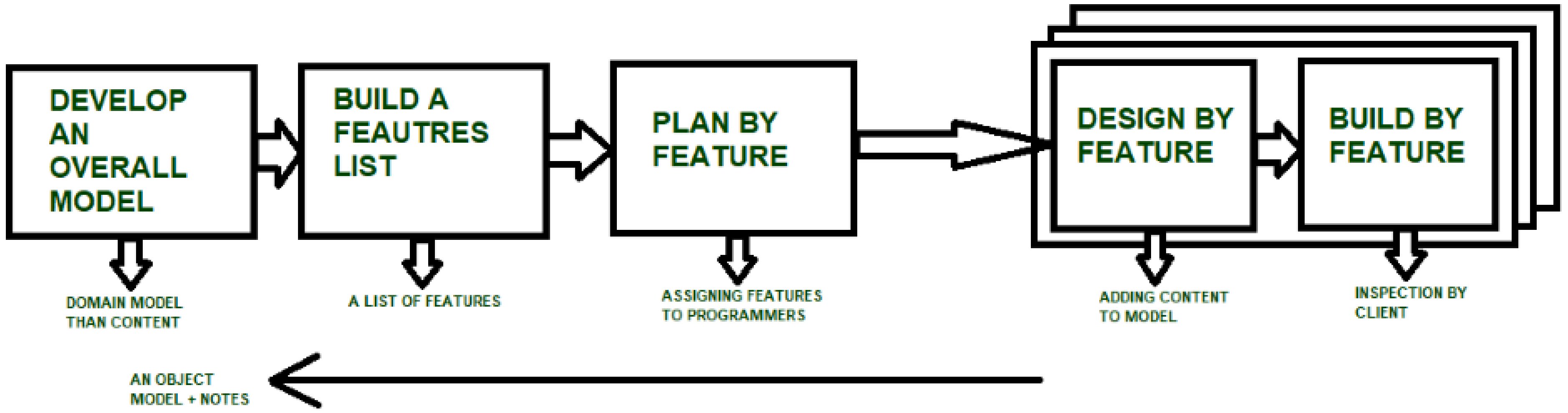
Shashi KS

Limitations of Behavioral-Driven Development

- BDD can be time-consuming to create and maintain extensive documentation, including user stories, acceptance criteria, and step definitions
- Involving all stakeholders, especially non-technical stakeholders, can be a challenge.
- BDD tests heavily rely on the quality of initial requirements and user stories; incomplete or inaccurate requirements can lead to insufficient test coverage and bugs
- BDD may not be suitable for short development cycles or projects with frequently changing requirements.

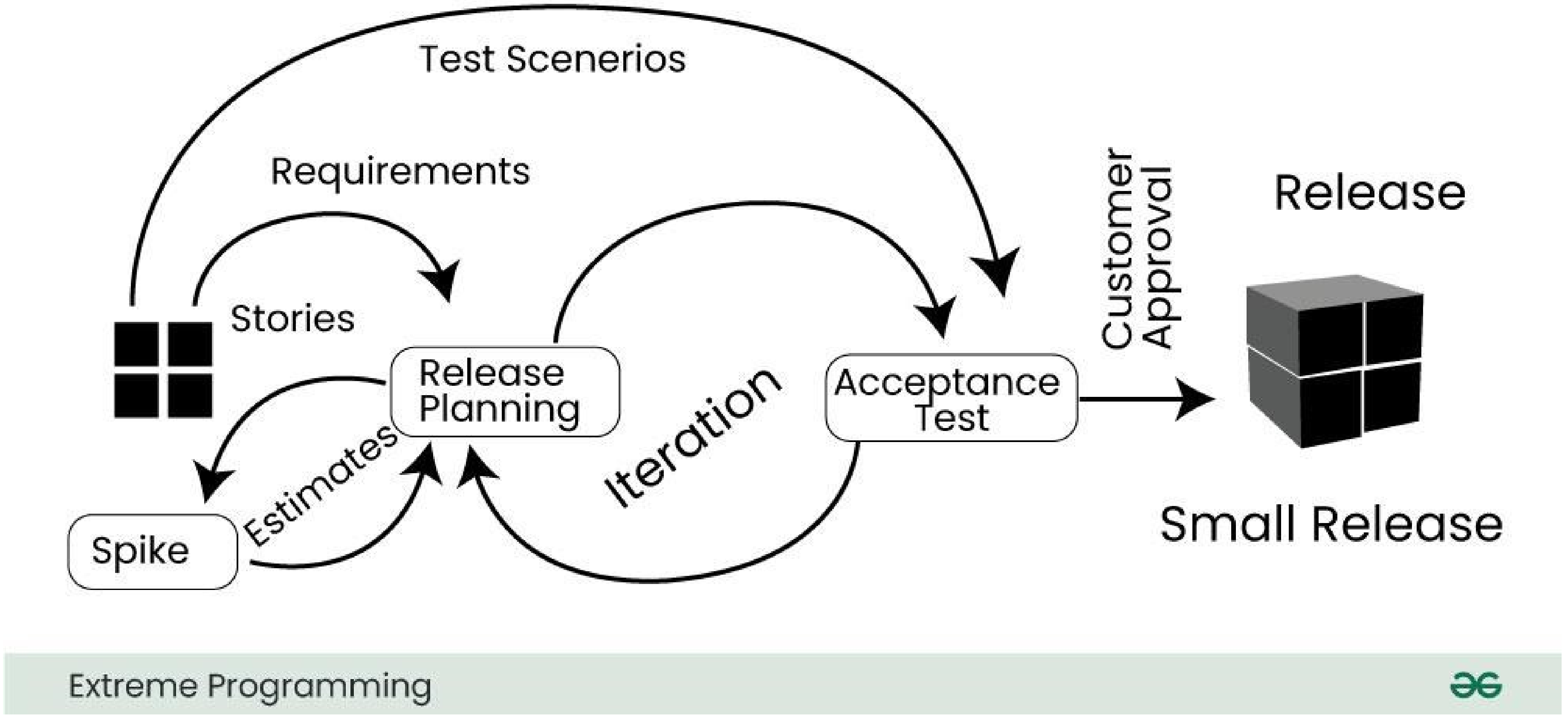
Feature driven development

Shashi KS



Extreme programming

Shashi KS



Shashi KS

Difference between feature driven and test driven development

- **Feature-driven development (FDD)**:
  - **Focus**: FDD focuses on delivering features incrementally.
  - **Process**: It begins with an overall model of the system, which is refined into a detailed feature list. Each feature is then designed, implemented, and tested individually before being integrated into the system.
  - **Iterations**: Development occurs in short, time-boxed iterations, typically lasting 1-2 weeks.
  - **Emphasis on Design**: FDD emphasizes design up front, with an initial focus on creating a comprehensive feature list and a domain model.
  - **Roles**: FDD often involves specialized roles like Chief Architect, Chief Programmer, and Domain Experts.
  - **Testing**: While testing is a part of FDD, it's not as prominent as in TDD. Testing typically occurs after feature implementation.

Shashi KS

Difference between feature driven and test driven development

- **Test-driven development (TDD)**:
  - **Focus**: TDD focuses on writing tests before writing the code.
  - **Process**: Developers start by writing a failing automated test that defines a desired improvement or new function. Then, they write the minimum amount of code required to pass that test. Finally, they refactor the code to improve its design while ensuring that all tests continue to pass.
  - **Iterations**: TDD also operates in short iterations, typically with cycles lasting a few minutes to an hour.
  - **Emphasis on Testing**: Testing is central to TDD. By writing tests first, developers ensure that their code behaves as expected and that any changes made in the future don't introduce regressions.
  - **Roles**: TDD doesn't prescribe specific roles; instead, it's a practice that can be adopted by any member of a development team.
  - **Continuous Integration**: TDD often works hand-in-hand with continuous integration, where code changes are frequently integrated into a shared repository and automatically tested.