Software Engineering

Dr. Mritunjay Shall Peelam
Assistant Professor-SG, UPES

Course Introduction and Objectives

Course Code	Course name	L	T	P	C
CSEG2064	Software Engineering	3	0	0	3
Total Units to be Covered: 5		Total Contact Hours: 45			
Prerequisite(s):		Syllabus version: 1.0			

Course Objectives

1. To explore software development methodologies (waterfall, agile, DevOps) and their integration of testing, quality assurance, reliability, and risk management.
2. To comprehend software requirements engineering and develop skills in creating well-structured Software Requirements Specifications (SRS).
3. To acquire understanding of planning a software project, its cost estimation models and to understand the software quality models.

Course Introduction and Objectives

Course Outcomes

CO 1. Understand the fundamental concepts and importance of Software Engineering in modern software development.

CO 2. Learn various software development methodologies, including Agile, Waterfall, and iterative approaches.

CO 3. Explore software design principles and architectural patterns for creating robust and maintainable software systems.

CO 4. Apply project management principles to effectively plan, monitor, and control software projects.

Course Introduction and Objectives

Syllabus

Unit I: Introduction to Software Engineering **7 Lecture Hours**

Definition of Software Engineering, S/W characteristics, applications, Software development life cycle ; Life Cycle Models – Waterfall (classical and iterative), Spiral, Prototyping & RAD Models, Software processes, Process Models – overview Agile Model and Various Agile methodologies - Scrum, XP, Lean, and Kanban. Scope of each model and their comparison in real-world case studies.

Course Introduction and Objectives

Unit II: Requirements Modelling and Design **9 Lecture Hours**

System and software requirements; Requirements Engineering-Crucial steps; types of requirements, Functional and non-functional requirements; Domain requirements; User requirements; Elicitation and analysis of requirements; Requirements documentation – Nature of Software, Software requirements specification, Use case diagrams with guidelines, DFD, SRS Structure, SRS Case study, Design concepts and principles - Abstraction - Refinement - Modularity Cohesion coupling, Architectural design, Detailed Design Transaction Transformation, Refactoring of designs, Object-oriented Design User-Interface Design.

Course Introduction and Objectives

Unit III: Software Reliability **9 Lecture Hours**

Introduction to Software Reliability; Hardware reliability vs. Software reliability; Reliability metrics; Failure and Faults – Prevention, Removal, Tolerance, Forecast; Dependability Concept – Failure Behavior, Characteristics, Maintenance Policy; Reliability and Availability Modeling; Reliability Evaluation Testing methods, Limits, Starvation, Coverage, Filtering; Microscopic Model of Software Risk; Classes of software reliability Models; Statistical reliability models; Reliability growth models; Defining and interpreting reliability metrics; Fault Detection and Prevention; Techniques for detecting and mitigating software faults; Static analysis tools and techniques; Dynamic analysis methods; Software Fault Tolerance; Software Maintenance and Reliability; Reliability Assessment and Evaluation; Methods for assessing and quantifying software reliability; Case Studies and Real-world Applications.

Course Introduction and Objectives

Unit IV: Software Testing, metrics and Quality Assurance **10 Lecture Hours**

Testing types and techniques such as black box, white box, and gray box testing, functional and structural testing; Test-driven development, code coverage, and quality metrics; Testing process, design of Test cases, testing techniques - boundary value analysis - equivalence class testing - decision table testing, cause-effect graphing, path testing, data flow testing, and mutation testing. Unit, integration, system, alpha, and beta testing, debugging techniques; verification and validation techniques, levels of testing, regression testing, quality management activities, product and process quality standards (ISO9000, CMM), metrics understanding (process, product, project metrics), size metrics (LOC, Function Count, Albrecht FPA), product metrics, metrics for software maintenance, cost estimation techniques (static, single variable, multivariable models), cost-benefit evaluation techniques, Testing tools and standards such as Jira and Selenium, test automation frameworks and tools (Selenium, Appium, JUnit), performance testing and load testing, and defect management and root cause analysis.

Course Introduction and Objectives

Unit V: Software Quality and Risk Management **10 Lecture Hours**

McCall quality factors, ISO and CMM Model, Tools and Techniques for Quality Control, Pareto Analysis, Statistical Sampling, Quality Control Charts and the seven Run Rule. Modern Quality Management, Risk Management – importance, types, process and phases, qualitative and quantitative risk analysis, Risk Analysis and Assessment, Risk Strategies, Risk Monitoring and Control, Risk Response and Evaluation. Software Reliability: Reliability Metrics, Reliability Growth Modeling. Use Case: Defect Tracking and Management. Test Automation Tools: Jira, Selenium:, Appium; JUnit.

Text Books and References

Textbooks	1. Roger S. Pressman, "Software Engineering: A practitioner's approach", 7th Edition, McGraw Hill, 2009. 2. Pankaj Jalote, "An integrated approach to Software Engineering", 3rd Edition, Springer/Narosa, 2005.
Reference books	1. James F. Peters, and Witold Pedrycz, "Software Engineering: an Engineering approach", John Wiley, 2007. 2. Waman S Jawadekar, "Software Engineering principles and practice", McGraw Hill, 2004.
Web Resources **Journals**	
MOOCs, online courses	

Modes of Evaluation

Modes of Evaluation: Quiz/Assignment/ presentation/ extempore/ Written Examination etc.

Examination Scheme

Components	IA	MID SEM	End Sem	Total
Weightage (%)	50	20	30	100

Unit-1

Syllabus

Unit I: Introduction to Software Engineering **7 Lecture Hours**

Definition of Software Engineering, S/W characteristics, applications, Software development life cycle ; Life Cycle Models – Waterfall (classical and iterative), Spiral, Prototyping & RAD Models, Software processes, Process Models – overview Agile Model and Various Agile methodologies - Scrum, XP, Lean, and Kanban. Scope of each model and their comparison in real-world case studies.

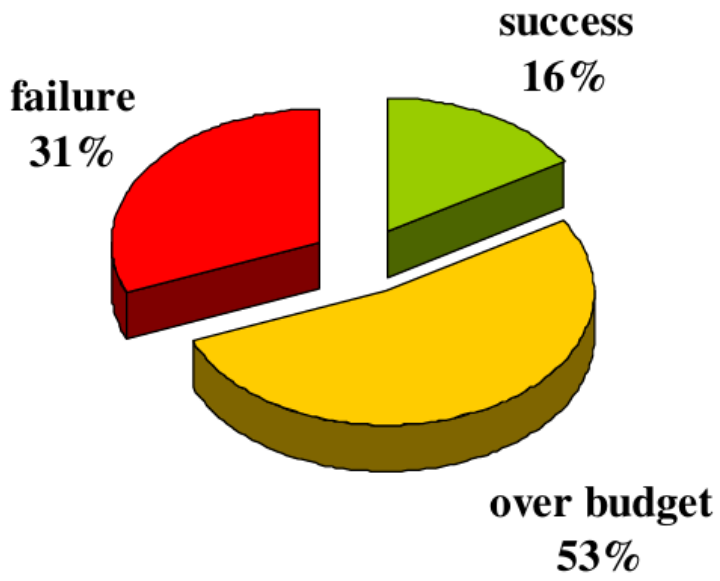
Why Software Engineering ?

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

Ready for change

The Evolving Role of Software

- Software industry is in Crisis!



Source: The Standish Group International, Inc. (CHAOS research)

The Evolving Role of Software

As per the IBM report, "31%of the project get cancelled before they are completed, 53% over-run their cost estimates by an average of 189% and for every 100 projects, there are 94 restarts".

Factors Contributing to the Software Crisis

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

Ariane 5

It took the European Space Agency **10 years and $7 billion** to produce Ariane 5, a giant rocket capable of hurling a pair of three-ton satellites into orbit with each launch and intended to give Europe overwhelming supremacy in the commercial space business.

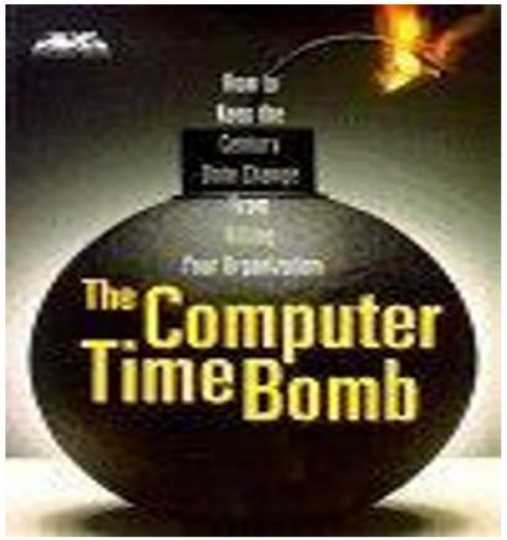
The rocket was destroyed after 39 seconds of its launch, at an altitude of two and a half miles along with its payload of four expensive and uninsured scientific satellites.

Some Software failures

Y2K problem:

It was simply the ignorance about the adequacy or otherwise of using only last two digits of the year.

The 4-digit date format, like 1964, was shortened to 2-digit format, like 64.



Windows XP

- Microsoft released Windows XP on October 25, 2001.
- On the same day company posted 18 MB of compatibility patches on the website for bug fixes, compatibility updates, and enhancements.
- Two patches fixed important security holes.

This is **Software Engineering.**

"No Silver Bullet"

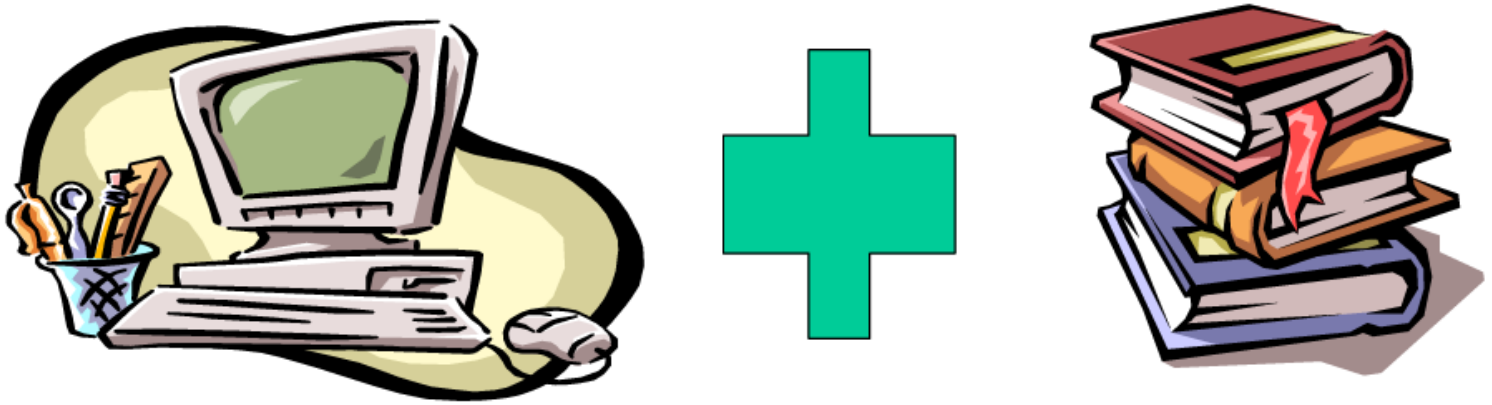
The hardware cost continues to decline drastically.

However, there are desperate cries for a silver bullet something to make software costs drop as rapidly as computer hardware costs do.

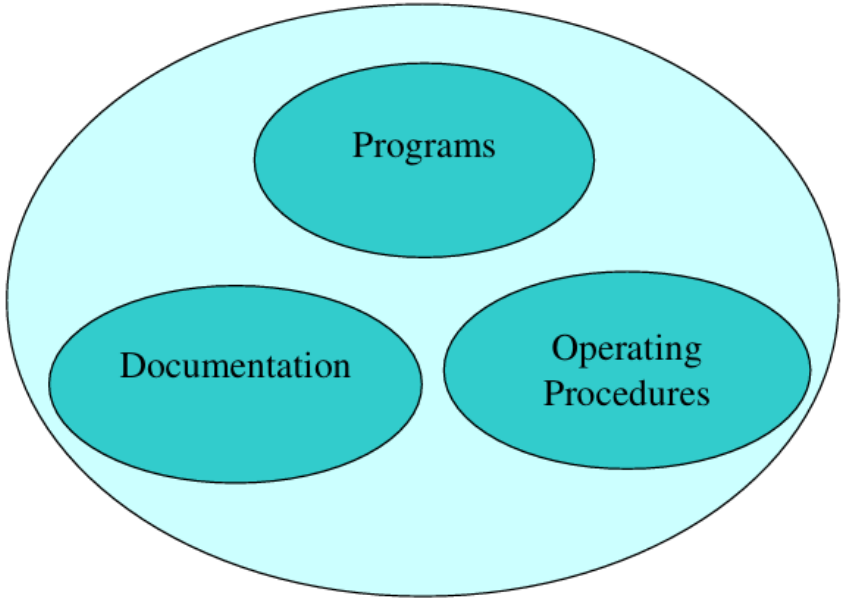
But as we look to the horizon of a decade, we see no silver bullet. There is no single development, either in technology or in management technique, that by itself promises even one order of magnitude improvement in productivity, in reliability and in simplicity.

What is software?

- **Computer programs** and **associated documentation**



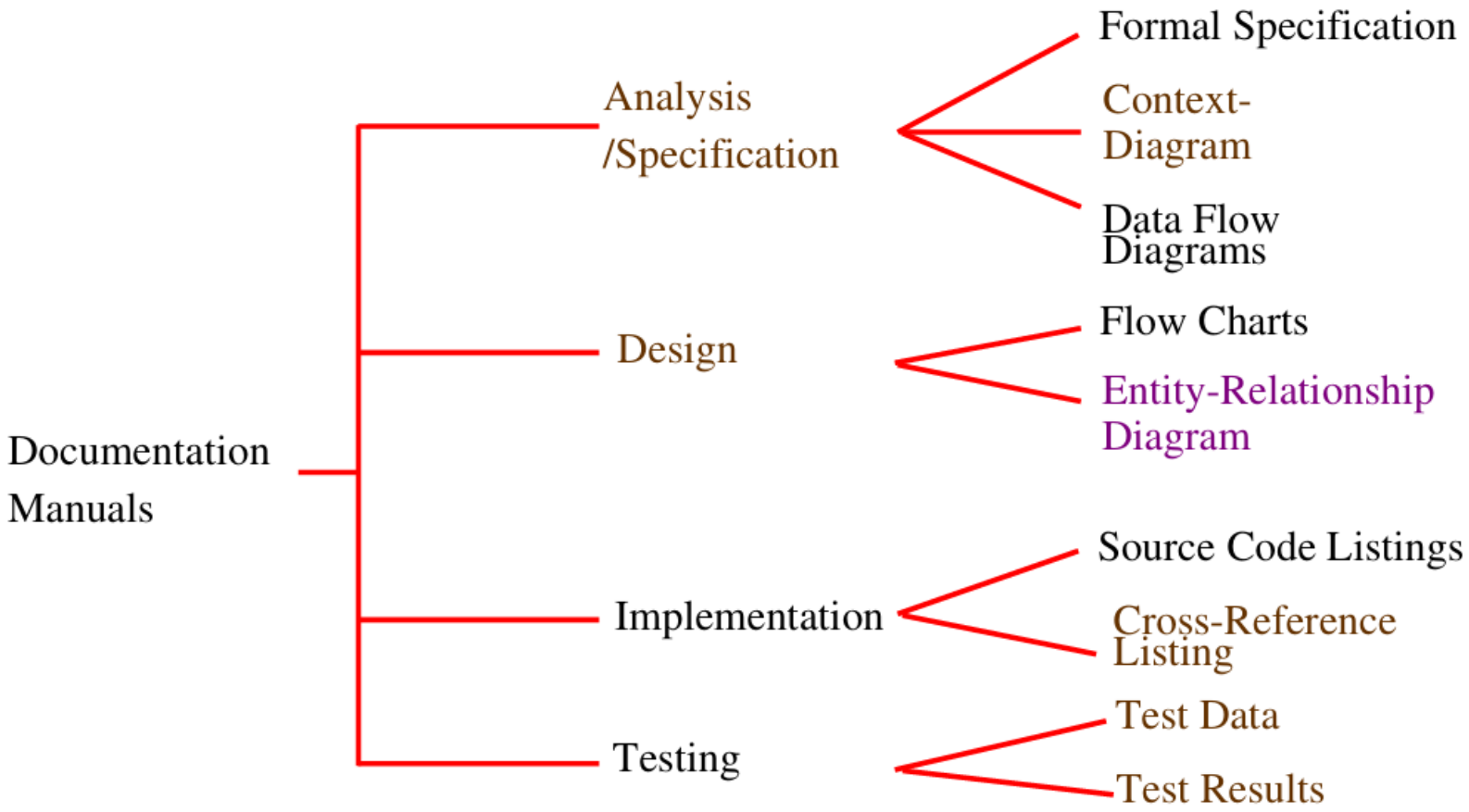
What is software?



Software=Program+Documentation+Operating Procedures

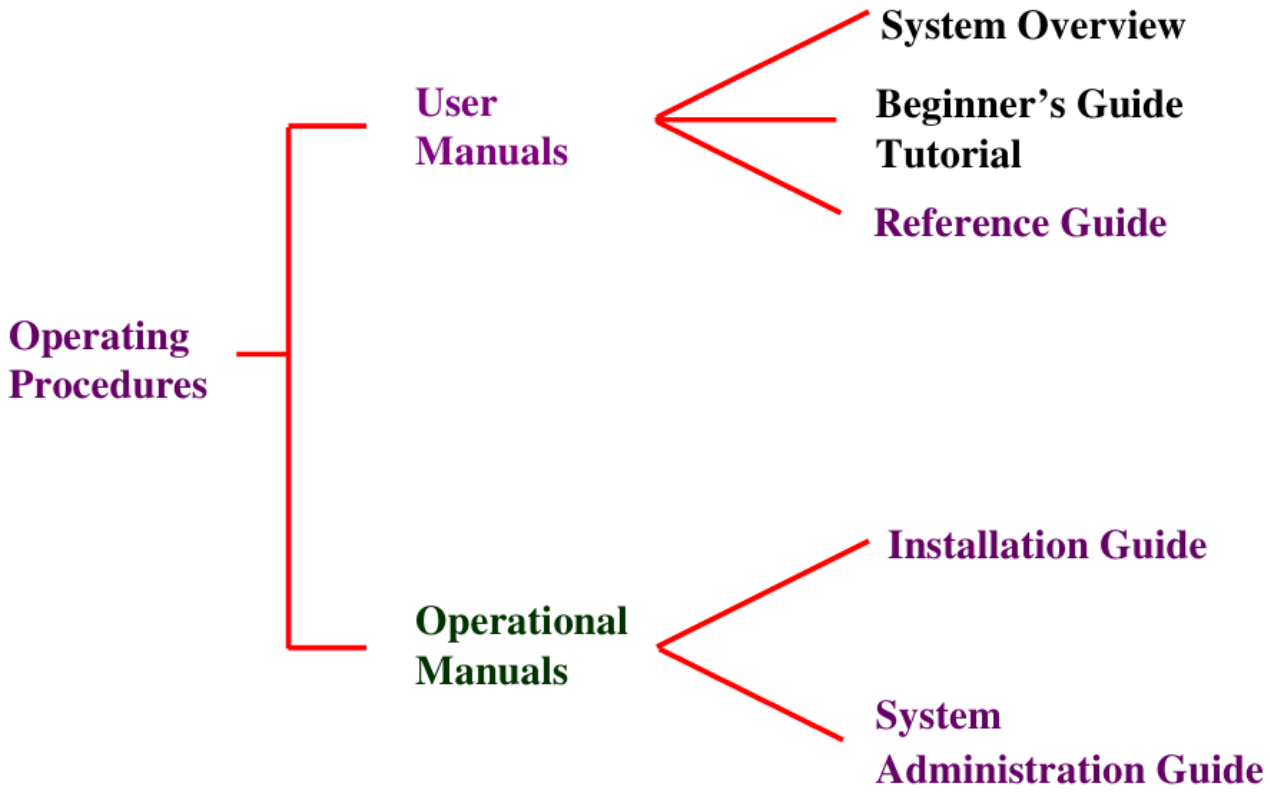
Components of software

Documentation consists of different types of manuals are



List of documentation manuals

Documentation consists of different types of manuals are



List of operating procedure manuals.

What is software engineering?

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

What is software engineering?

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.

Software Process

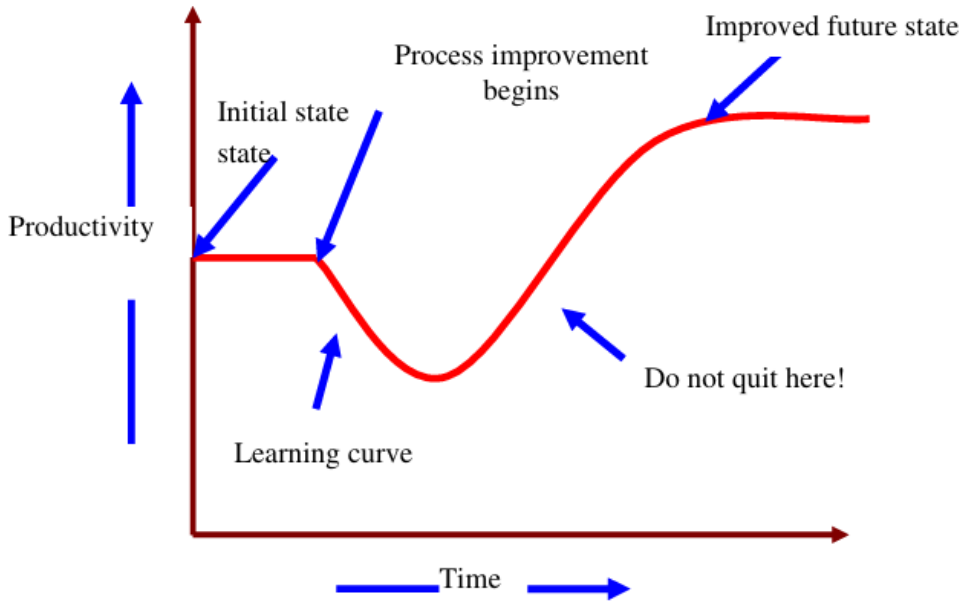
The software process is the way in which we produce software.

Why is it difficult to improve software process ?

- Not enough time
- Lack of knowledge

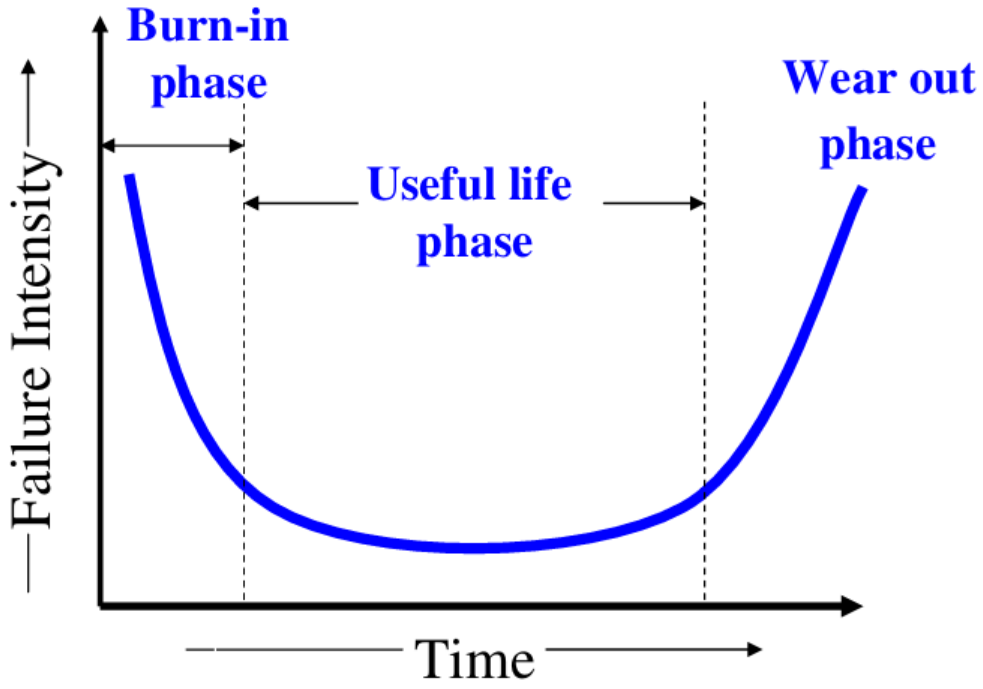
Software Process

- Wrong motivations
- Insufficient commitment



Software Characteristics:

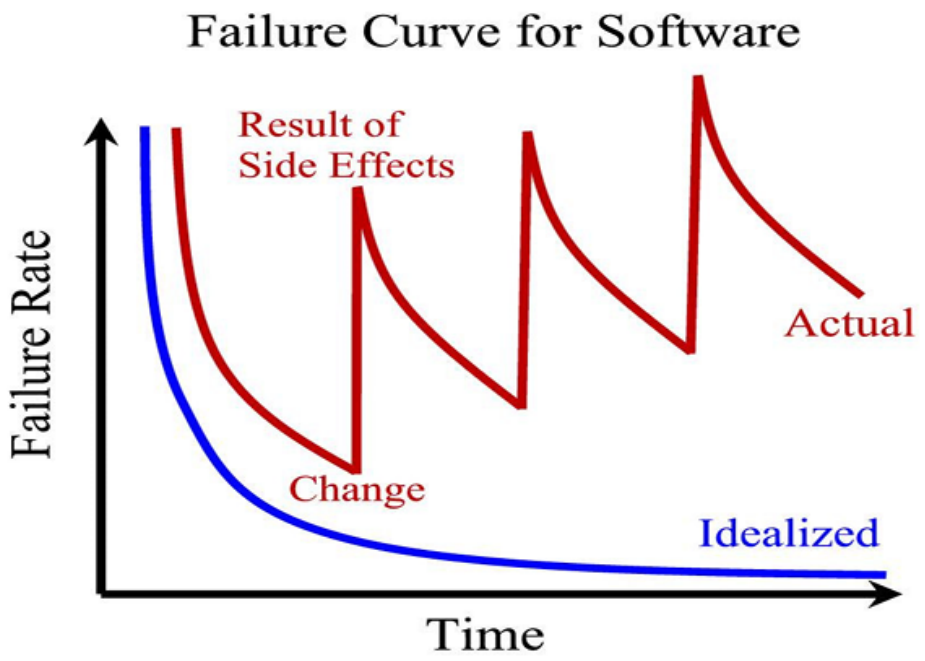
✓Software does not wear out.



Failure Curve for Hardware

Software Characteristics:

- ✓ Software is not manufactured
- ✓ Reusability of components
- ✓ Software is flexible

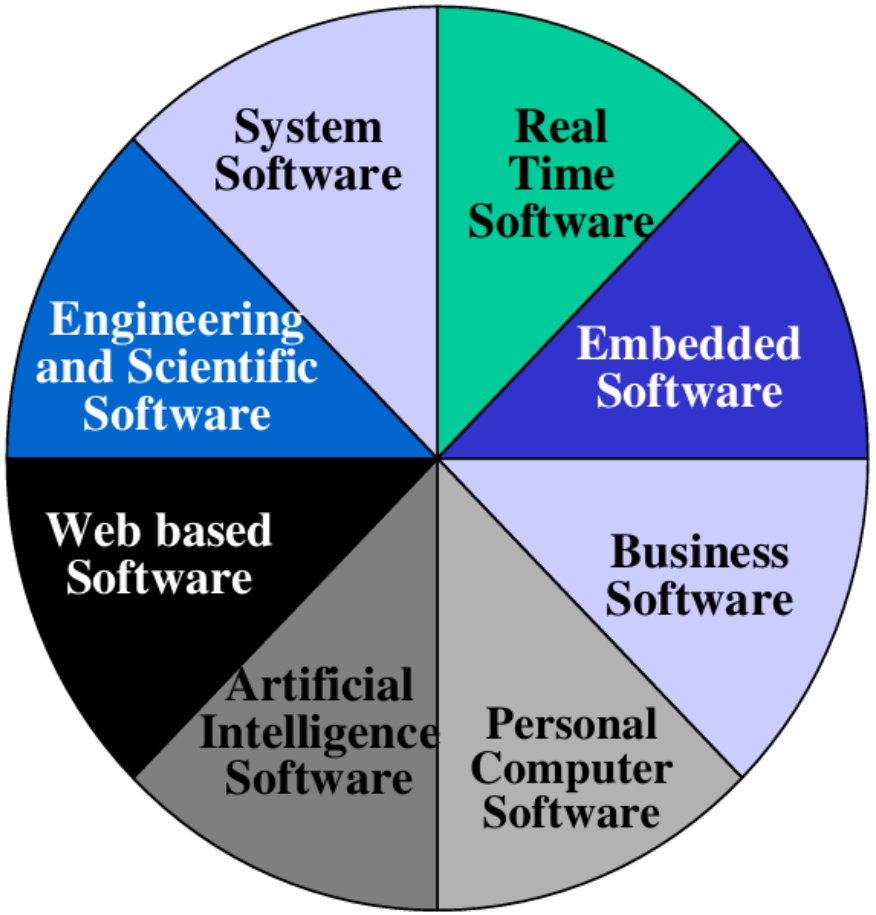


Software Characteristics:

Comparison of constructing a bridge vis-à-vis writing a program.

Sr. No	Constructing a bridge	Writing a program
1.	The problem is well understood	Only some parts of the problem are understood, others are not
2.	There are many existing bridges	Every program is different and designed for special applications.
3.	The requirement for a bridge typically do not change much during construction	Requirements typically change during all phases of development.
4.	The strength and stability of a bridge can be calculated with reasonable precision	Not possible to calculate correctness of a program with existing methods.
5.	When a bridge collapses, there is a detailed investigation and report	When a program fails, the reasons are often unavailable or even deliberately concealed.
6.	Engineers have been constructing bridges for thousands of years	Developers have been writing programs for 50 years or so.
7.	Materials (wood, stone,iron, steel) and techniques (making joints in wood, carving stone, casting iron) change slowly.	Hardware and software changes rapidly.

The Changing Nature of Software



Software Myths (Management Perspectives)

Software is easy to change

The reality is totally different.

Software Myths (Management Perspectives)

Computers provide greater reliability than the devices they replace

This is not always true.

Software Myths (Customer Perspectives)

Software with more features is better software

Software can work right the first time

Both are only myths!

Software Myths (Developer Perspectives)

Once the software is demonstrated, the job is done.

Usually, the problems just begin!

Some Terminologies

- Deliverables and Milestones

Different deliverables are generated during software development. The examples are source code, user manuals, operating procedure manuals etc.

The milestones are the events that are used to ascertain the status of the project. Finalization of specification is a milestone. Completion of design documentation is another milestone. The milestones are essential for project planning and management.

Some Terminologies

- Product and Process

Product: What is delivered to the customer, is called a product. It may include source code, specification document, manuals, documentation etc. Basically, it is nothing but a set of deliverables only.

Process: Process is the way in which we produce software. It is the collection of activities that leads to (a part of) a product. An efficient process is required to produce good quality products.

If the process is weak, the end product will undoubtedly suffer, but an obsessive over reliance on process is also dangerous.

Some Terminologies

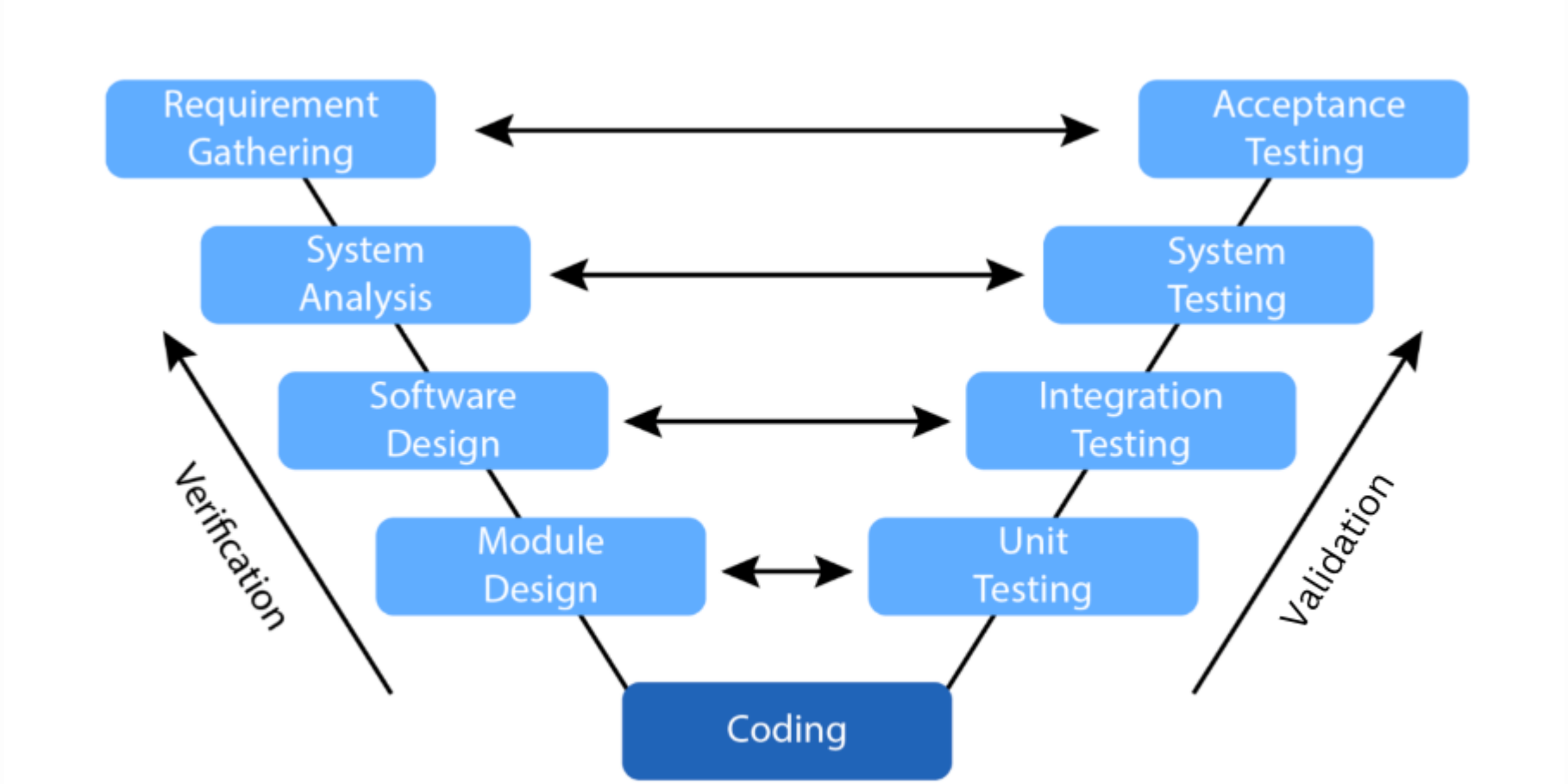
- Measures, Metrics and Measurement

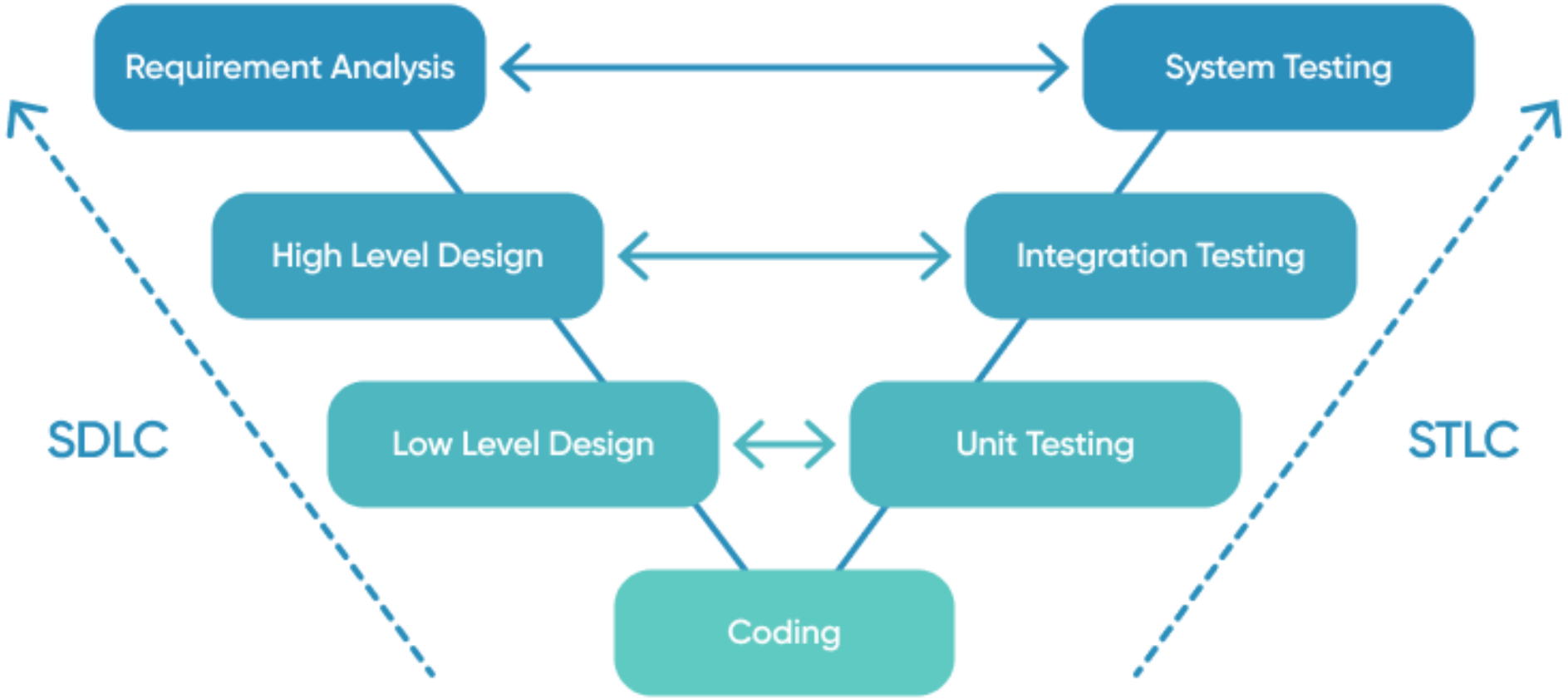
A measure provides a quantitative indication of the extent, dimension, size, capacity, efficiency, productivity or reliability of some attributes of a product or process.

Measurement is the act of evaluating a measure.

A metric is a quantitative measure of the degree to which a system, component or process possesses a given attribute.

Software Development Life Cycle




Requirement Analysis
System Testing
High Level Design
Integration Testing
Low Level Design
Unit Testing
Coding
SDLC
STLC

Software Life Cycle Models

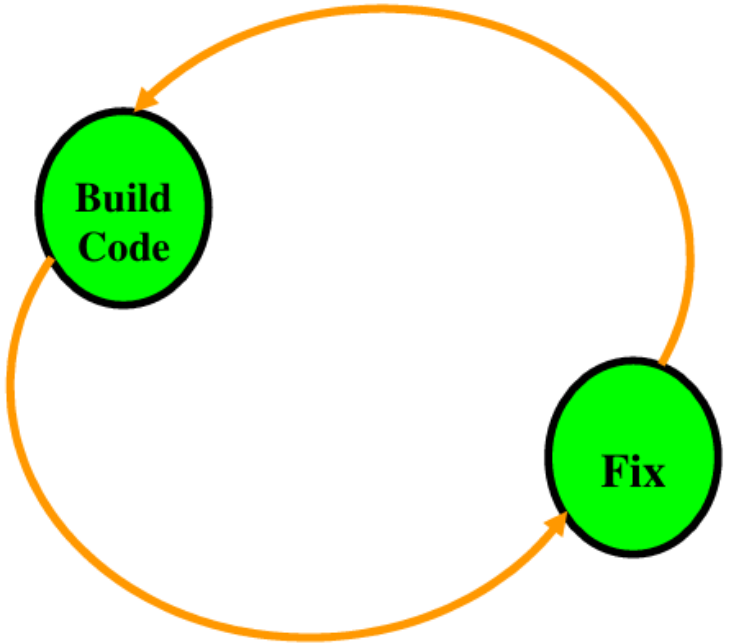
The goal of Software Engineering is to provide models and processes that lead to the production of well-documented maintainable software in a manner that is predictable.

Software Life Cycle Models

"The period of time that starts when a software product is conceived and ends when the product is no longer available for use. The software life cycle typically includes a requirement phase, design phase, implementation phase, test phase, installation and check out phase, operation and maintenance phase, and sometimes retirement phase".

Build & Fix Model

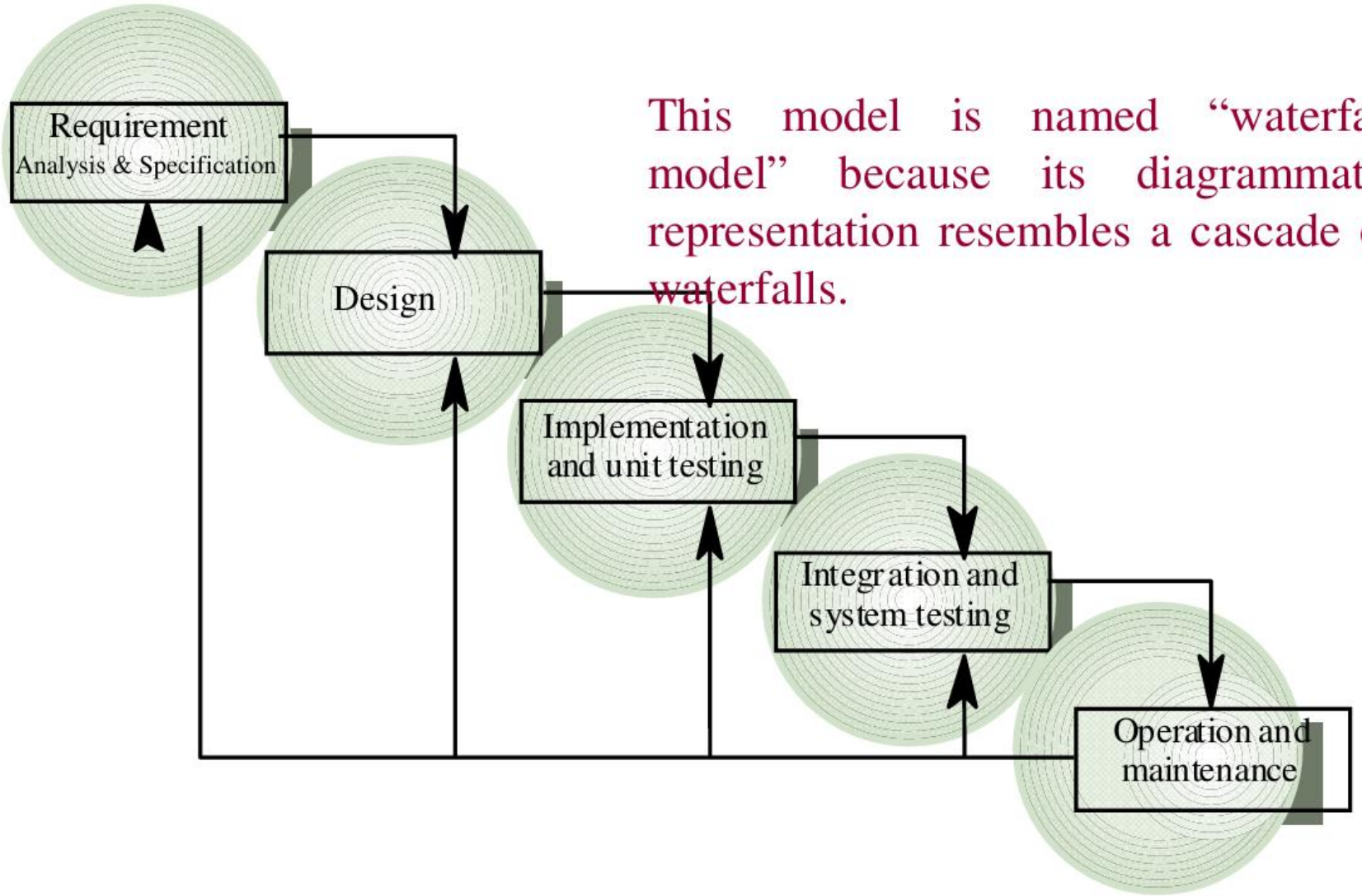
- Product is constructed without specifications or any attempt at design
- Adhoc approach and not well defined
- Simple two phase model



Build & Fix Model

- Suitable for small programming exercises of 100 or 200 lines
- Unsatisfactory for software for any reasonable size
- Code soon becomes unfixable & unenhanceable
- No room for structured design
- Maintenance is practically not possible

Waterfall Model



This model is named "waterfall model" because its diagrammatic representation resembles a cascade of waterfalls.

Waterfall Model

This model is easy to understand and reinforces the notion of "define before design" and "design before code".

The model expects complete & accurate requirements early in the process, which is unrealistic

Waterfall Model

Problems of waterfall model

i. It is difficult to define all requirements at the beginning of a project

ii. This model is not suitable for accommodating any change

iii. A working version of the system is not seen until late in the project's life

iv. It does not scale up well to large projects.

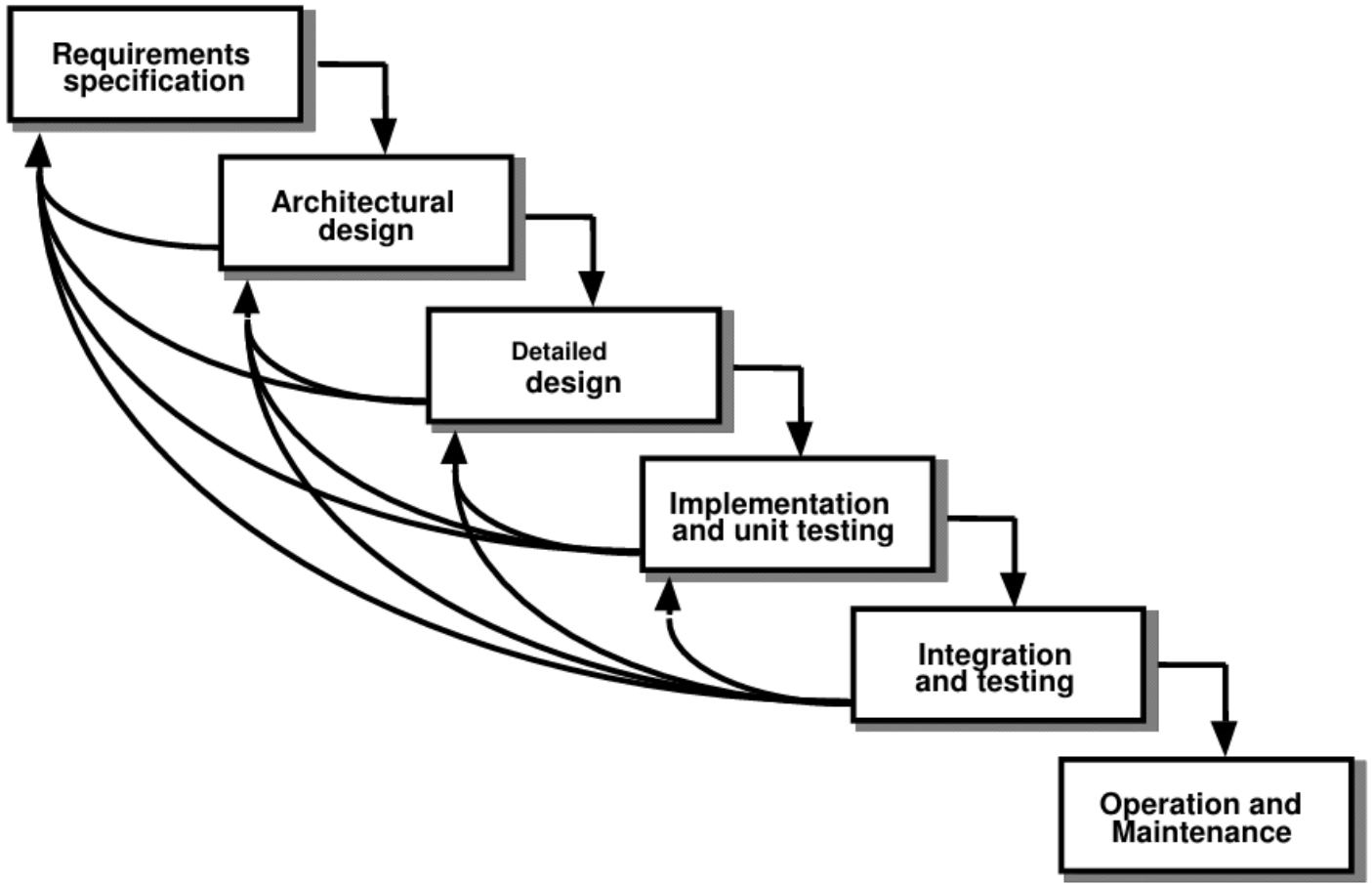
v. Real projects are rarely sequential.

Iterative Enhancement Model

This model has the same phases as the waterfall model, but with fewer restrictions. Generally the phases occur in the same order as in the waterfall model, but they may be conducted in several cycles. Useable product is released at the end of the each cycle, with each release providing additional functionality.

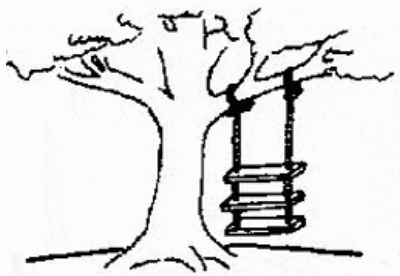
- ✓ Customers and developers specify as many requirements as possible and prepare a SRS document.
- ✓ Developers and customers then prioritize these requirements
- ✓ Developers implement the specified requirements in one or more cycles of design, implementation and test based on the defined priorities.

Iterative Enhancement Model

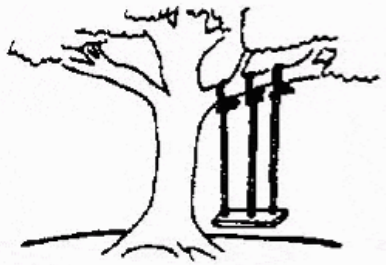


The Rapid Application Development (RAD) Mode

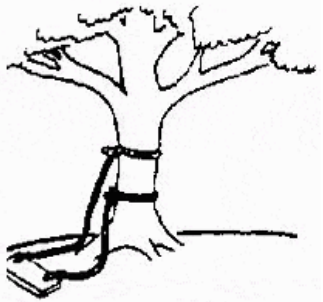
- Developed by IBM in 1980
- User participation is essential



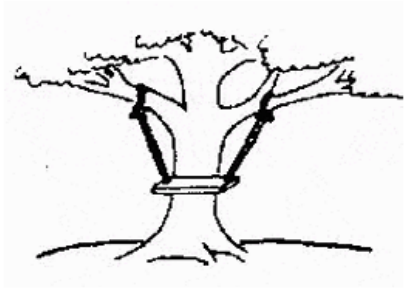
The requirements specification was defined like this



The developers understood it in that way



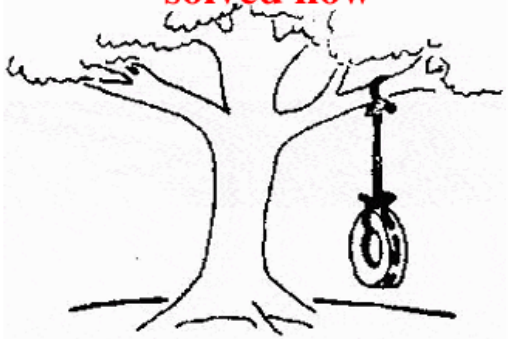
This is how the problem was solved before.



This is how the problem is solved now

That is the program after debugging

This is how the program is described by marketing department

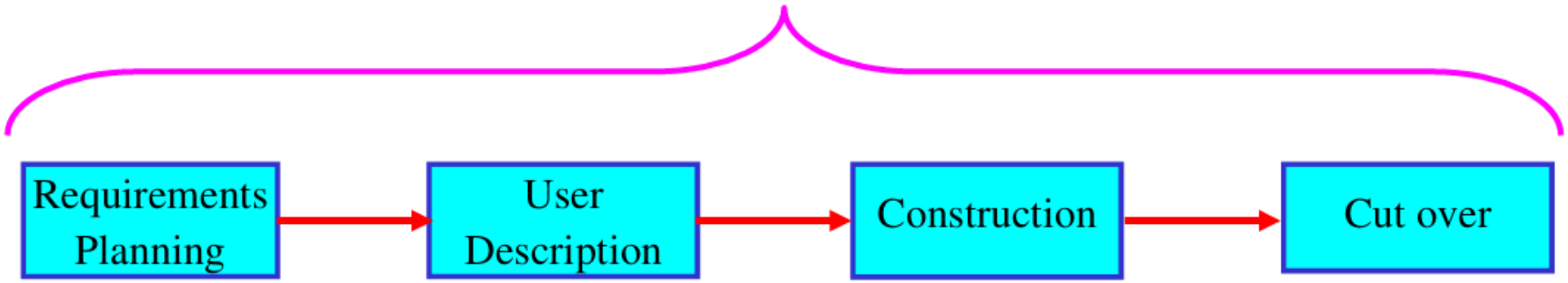


This, in fact, is what the customer wanted …

The Rapid Application Development (RAD) Model

- Build a rapid prototype
- Give it to user for evaluation & obtain feedback
- Prototype is refined

With active participation of users



The Rapid Application Development (RAD) Model

Not an appropriate model in the absence of user participation.

Reusable components are required to reduce development time.

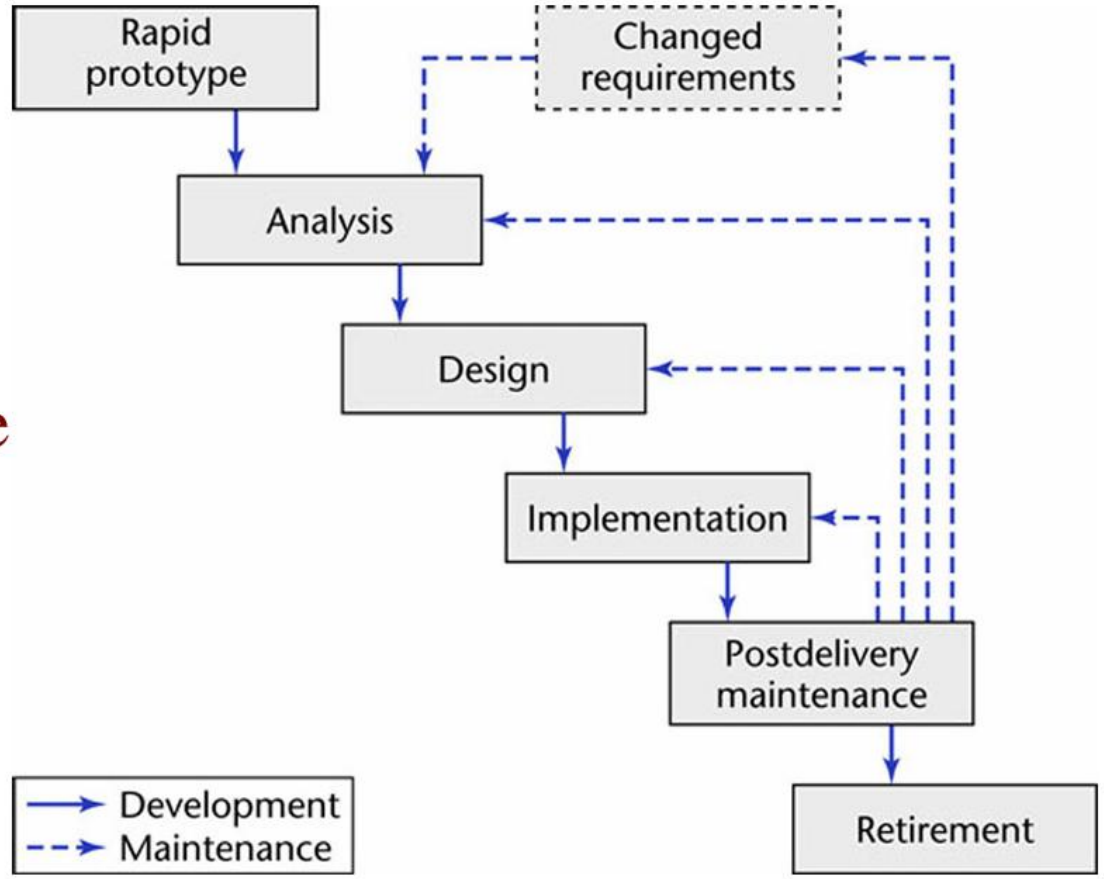
Highly specialized & skilled developers are required and such developers are not easily available.

Prototyping Model

- The prototype may be a usable program but is not suitable as the final software product.
- The code for the prototype is thrown away. However experience gathered helps in developing the actual system.
- The development of a prototype might involve extra cost, but overall cost might turnout to be lower than that of an equivalent system developed using the waterfall model.

Prototyping Model

- Linear mode
- "Rapid"



Spiral Model

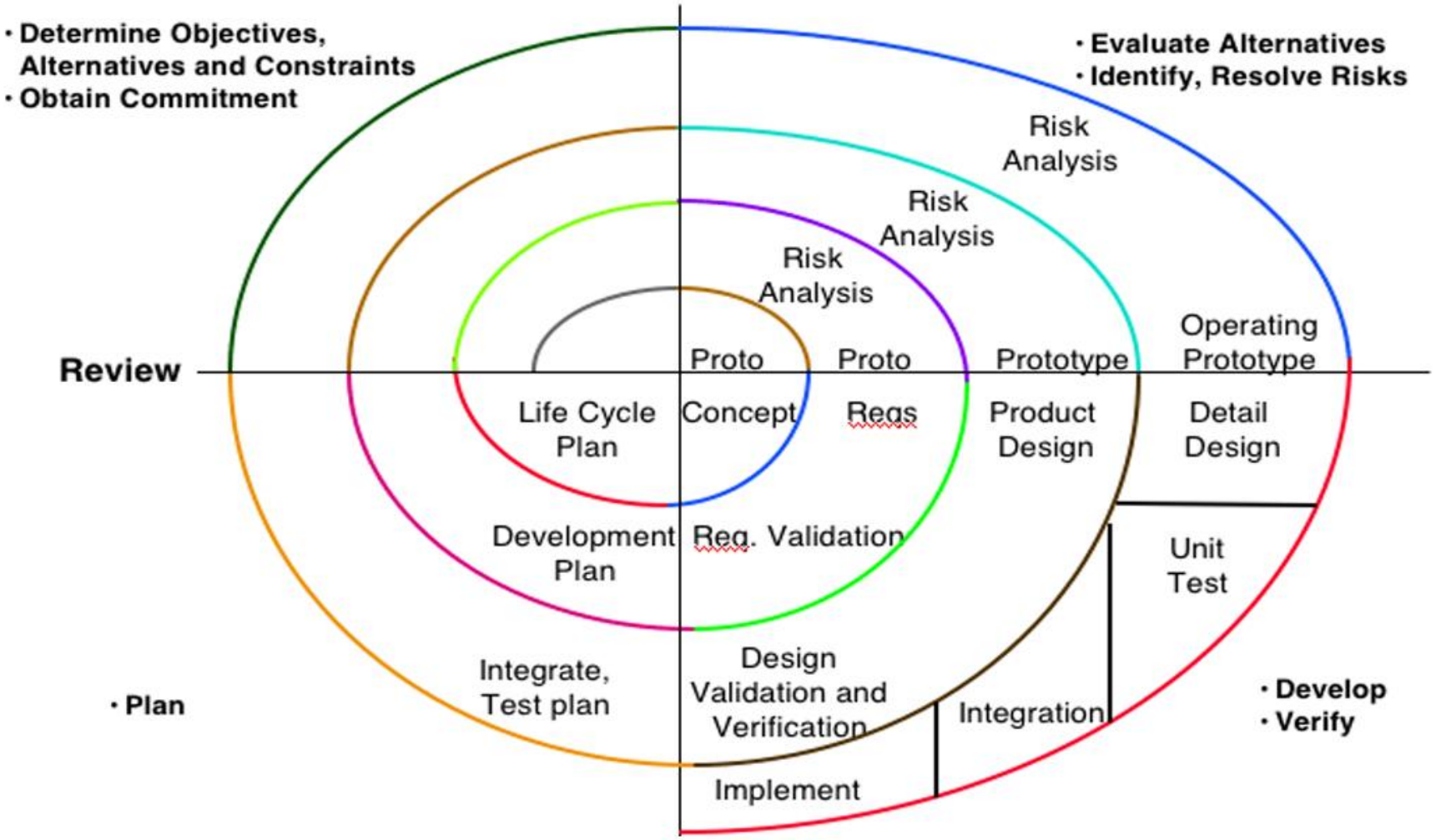
Models do not deal with uncertainly which is inherent to software projects.

Important software projects have failed because project risks were neglected & nobody was prepared when something unforeseen happened.

Barry Boehm recognized this and tired to incorporate the "project risk" factor into a life cycle model.

The result is the spiral model, which was presented in 1986.

Spiral Model



Spiral Model

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 360º represents one phase. One phase is split roughly into four sectors of major activities.

- **Planning:** Determination of objectives, alternatives & constraints.
- **Risk Analysis:** Analyze alternatives and attempts to identify and resolve the risks involved.
- **Development:** Product development and testing product.
- **Assessment:** Customer evaluation

Spiral Model

- An important feature of the spiral model is that each phase is completed with a review by the people concerned with the project (designers and programmers)
- The advantage of this model is the wide range of options to accommodate the good features of other life cycle models.
- It becomes equivalent to another life cycle model in appropriate situations.

The spiral model has some difficulties that need to be resolved before it can be a universally applied life cycle model. These difficulties include lack of explicit process guidance in determining objectives, constraints, alternatives; relying on risk assessment expertise; and provides more flexibility than required for many applications.

Selection of a Life Cycle Model

Selection of a model is based on:

a) Requirements

b) Development team

c) Users

d) Project type and associated risk

Based On Characteristics Of Requirements

Requirements	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
Are requirements easily understandable and defined?	Yes	No	No	No	No	Yes
Do we change requirements quite often?	No	Yes	No	No	Yes	No
Can we define requirements early in the cycle?	Yes	No	Yes	Yes	No	Yes
Requirements are indicating a complex system to be built	No	Yes	Yes	Yes	Yes	No

Based On Status Of Development Team

Development team	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
Less experience on similar projects?	No	Yes	No	No	Yes	No
Less domain knowledge (new to the technology)	Yes	No	Yes	Yes	Yes	No
Less experience on tools to be used	Yes	No	No	No	Yes	No
Availability of training if required	No	No	Yes	Yes	No	Yes

Based On User's Participation

Involvement of Users	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
User involvement in all phases	No	Yes	No	No	No	Yes
Limited user participation	Yes	No	Yes	Yes	Yes	No
User have no previous experience of participation in similar projects	No	Yes	Yes	Yes	Yes	No
Users are experts of problem domain	No	Yes	Yes	Yes	No	Yes

Based On Type Of Project With Associated Risk

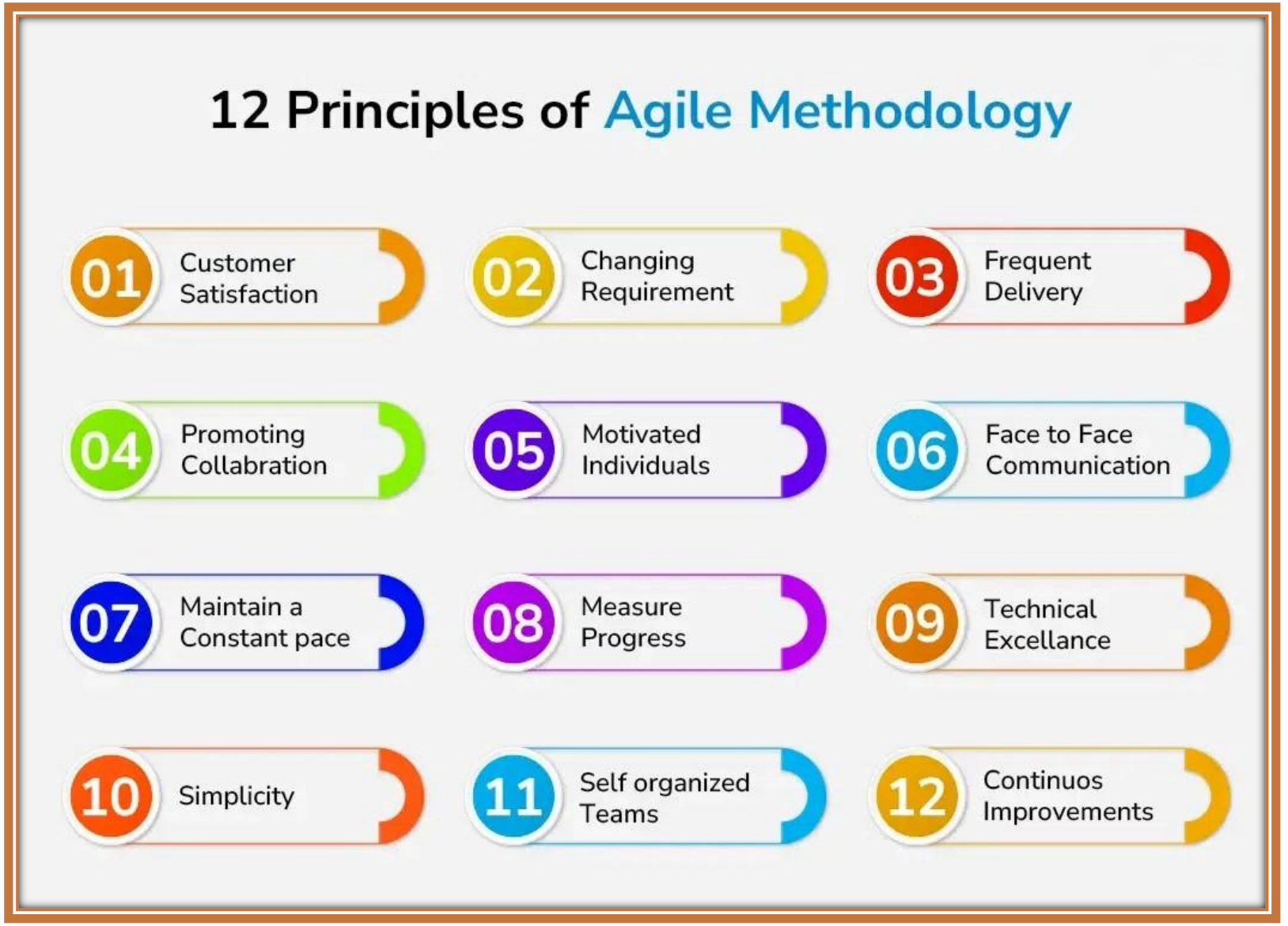
Project type and risk	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
Project is the enhancement of the existing system	No	No	Yes	Yes	No	Yes
Funding is stable for the project	Yes	Yes	No	No	No	Yes
High reliability requirements	No	No	Yes	Yes	Yes	No
Tight project schedule	No	Yes	Yes	Yes	Yes	Yes
Use of reusable components	No	Yes	No	No	Yes	Yes
Are resources (time, money, people etc.) scare?	No	Yes	No	No	Yes	No

What is Agile?

- Agile is a Project Management and software development approach that aims to be more effective.
- It focuses on delivering smaller pieces of work regularly instead of one big launch.
- This allows teams to adapt to changes quickly and provide customer value faster.

What is the Agile Methodology?

- Agile Methodology is a way to manage projects by breaking them into smaller parts.
- It focuses on working together and making constant improvements. Teams plan, work on the project, and then review how things are going in a repeating cycle.
- They prioritize flexibility, collaboration, and customer satisfaction.
- Major companies like Facebook, Google, and Amazon use Agile because of its adaptability and customer-focused approach.


12 Principles of Agile Methodology
01 Customer Satisfaction
02 Changing Requirement
03 Frequent Delivery
04 Promoting Collabration
05 Motivated Individuals
06 Face to Face Communication
07 Maintain a Constant pace
08 Measure Progress
09 Technical Excellance
10 Simplicity
11 Self organized Teams
12 Continuos Improvements

Agile Development Models

- The main difficulties included handling customer change requests during project development and the high cost and time required to incorporate these changes.
- In the mid-1990s, the Agile Software Development Model was proposed to overcome these drawbacks of the Waterfall Model.
- The Agile Model was primarily designed to help a project adapt quickly to change requests.
- So, the main aim of the Agile model is to facilitate quick project completion. To accomplish this task, it's important that agility is required.
- Agility is achieved by fitting the process to the project and removing activities that may not be essential for a specific project.

Agile Development Models

- **The Agile Model is a combination of iterative and incremental process models. The phases involve in Agile (SDLC) Model are:**

- **Requirement Gathering**
- **Design the Requirements**
- **Construction / Iteration**
- **Testing / Quality Assurance**
- **Deployment**
- **Feedback**

Agile Development Models

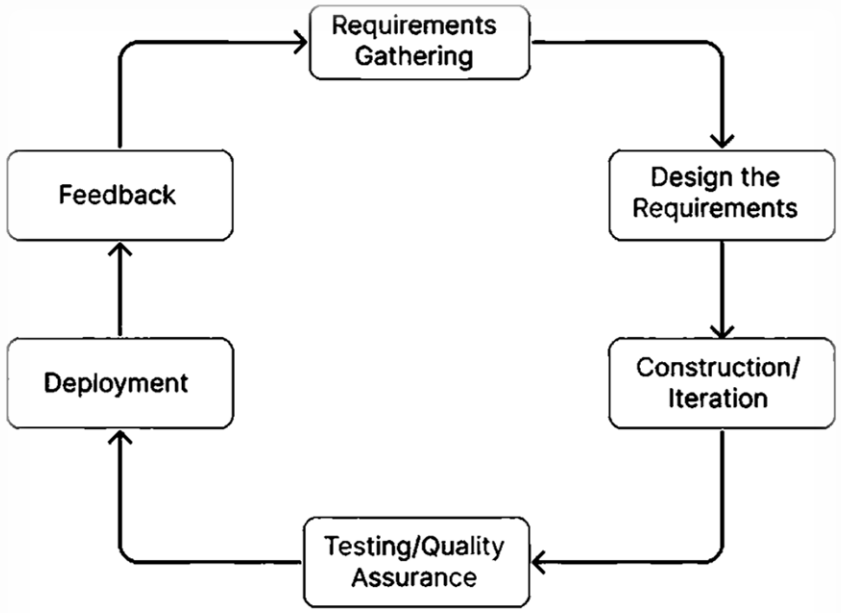
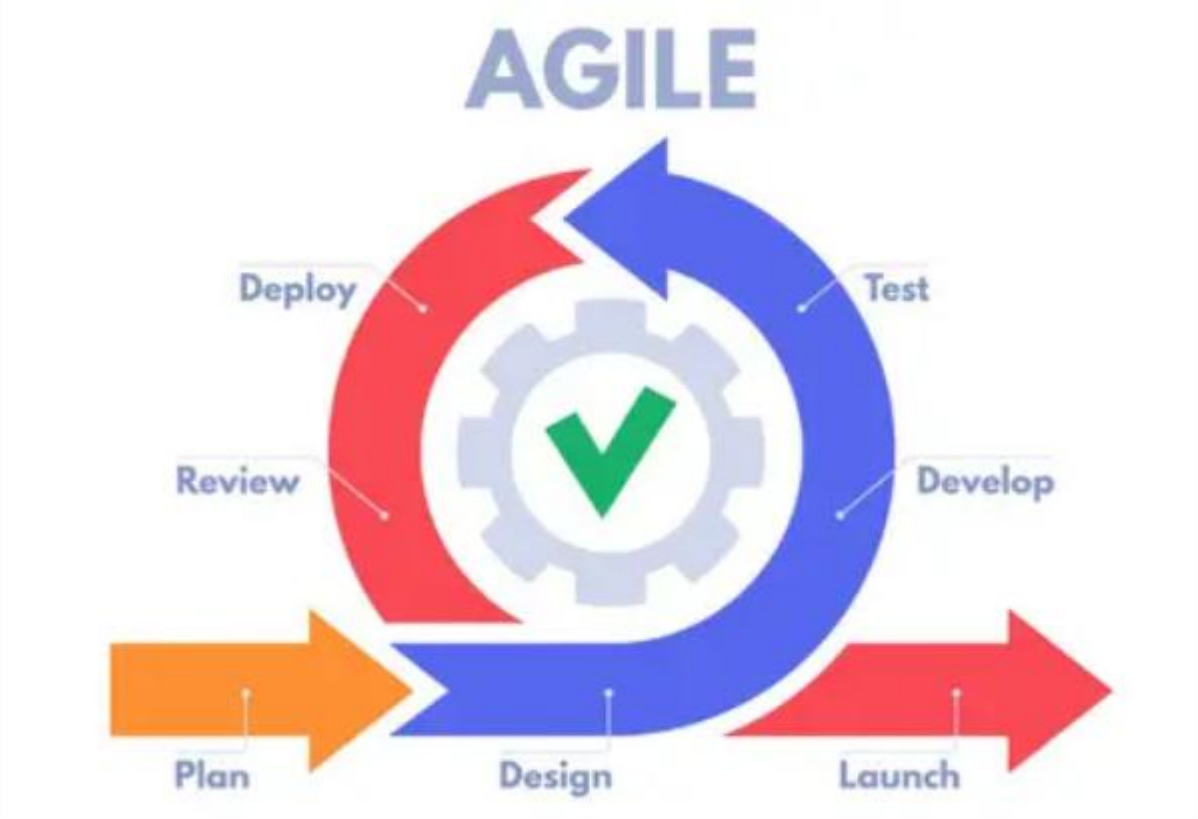


Fig:- Steps in Agile SDLC Model



Agile Development Models

1. 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.**

2. 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.**

Agile Development Models

3. Construction / Iteration

- In this step, development team members start working on their project, which aims to deploy a working product.
- Each cycle typically consist between 1-4 weeks, and at the end, the team delivers a working version of the software.

4. Testing / Quality Assurance

- Testing involves Unit Testing, Integration Testing, and System Testing, Which help in the agile development models.

Agile Development Models

5. Deployment

- In this step, the development team will deploy the working project to end users.
- Once an iteration is finished and fully tested, the software is ready to be released to the end users.
- In Agile, deployment isn't a one-time event it's an ongoing process.
- Updates and improvements are rolled out regularly, making sure the software keeps evolving and getting better with each release.

6. 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.

Characteristics of the Agile Process

- The Agile process is all about being flexible, working together, and focusing on delivering real value to customers.
- 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.

Disadvantages of the Agile Model

- 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.

Disadvantages of the Agile Model

- 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.

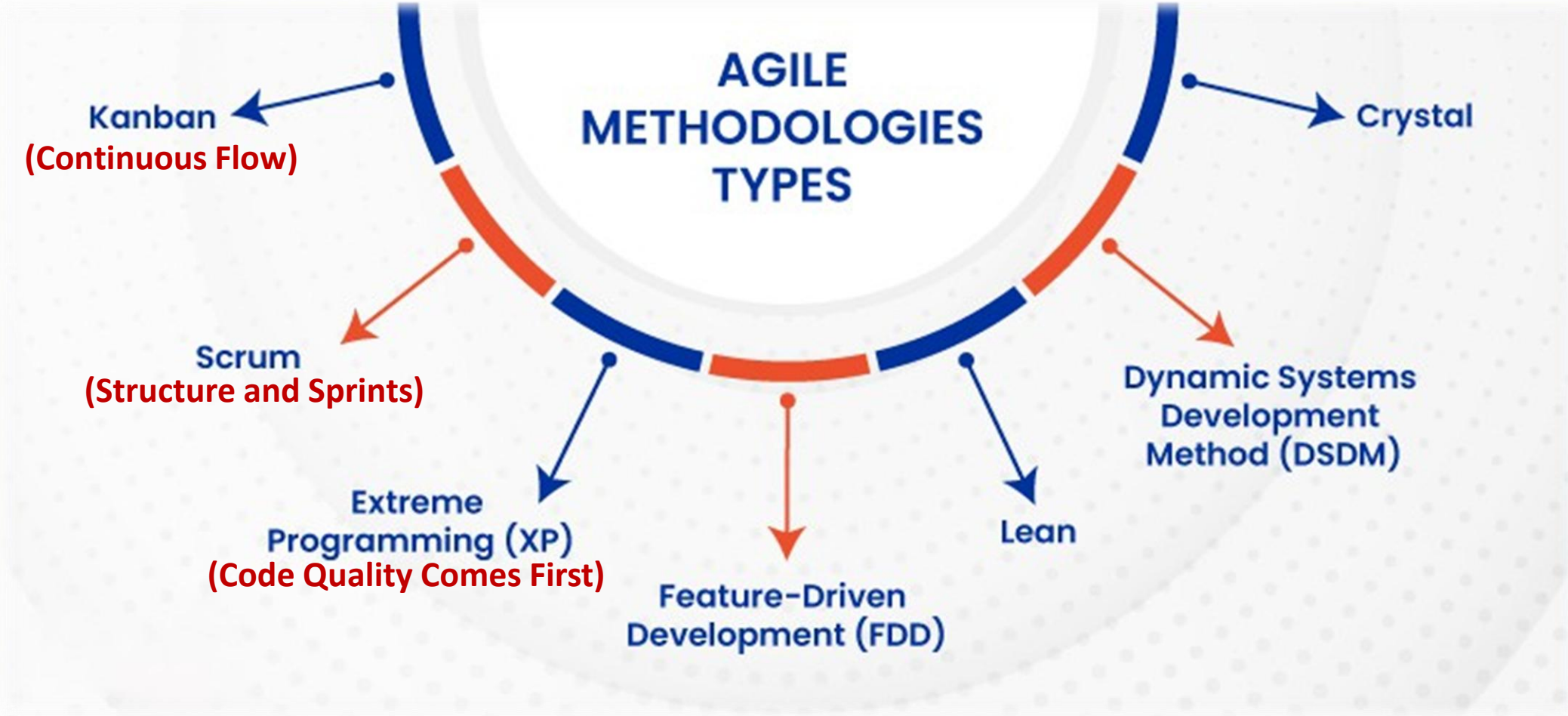
- Due to the absence of proper documentation, when the project completes and the developers are assigned to another project, maintenance of the developed project can become a problem.

Various Agile methodologies

- 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.

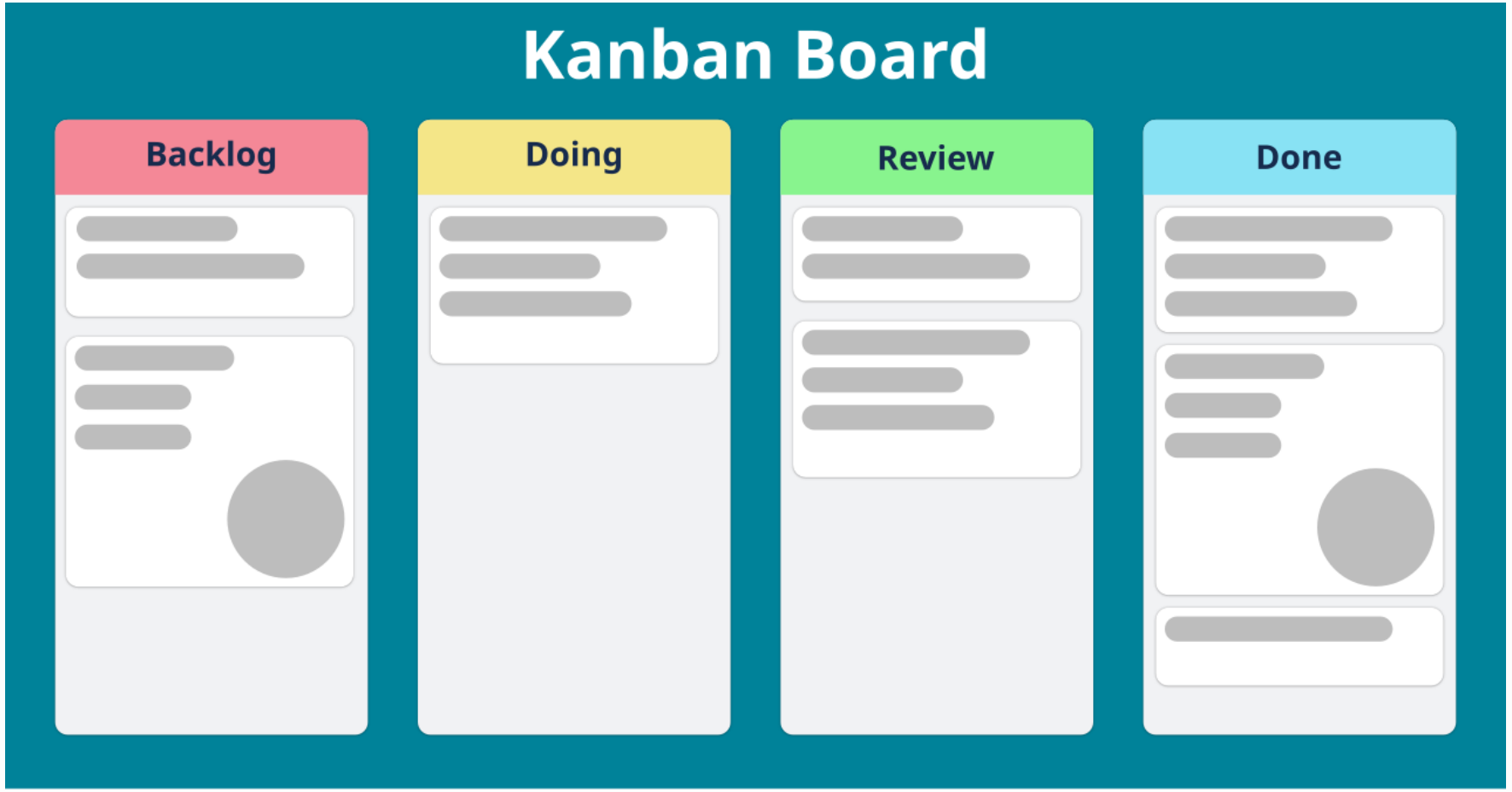
Various Agile methodologies



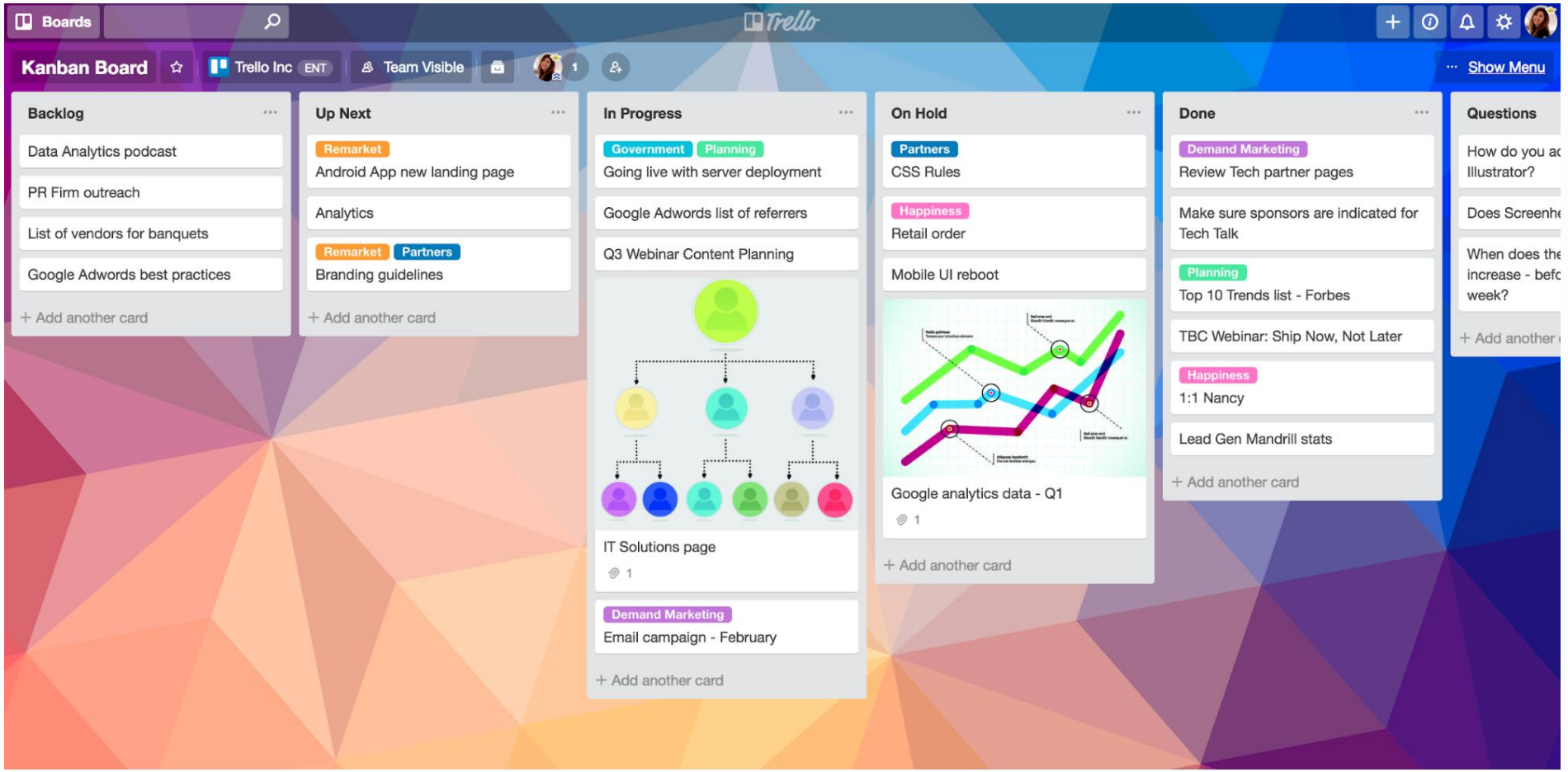
Kanban

- A type of Agile framework that has originated from the Japanese language, the word 'Kanban' means a signboard or a visual board.
- The Kanban concept is connected to the Just In Time methodology.
- Back in the day, Kanban was primarily introduced as a Lean Manufacturing System, and then it was slowly introduced into Agile software development procedures.
- The Kanban framework uses visual elements for managing & developing projects.

Kanban



Kanban


Boards
Trello
Kanban Board
Trello Inc ENT
Team Visible
Show Menu
Backlog
Data Analytics podcast
PR Firm outreach
List of vendors for banquets
Google Adwords best practices
+ Add another card
Up Next
Remarket
Android App new landing page
Analytics
Remarket Partners
Branding guidelines
+ Add another card
In Progress
Government Planning
Going live with server deployment
Google Adwords list of referrers
Q3 Webinar Content Planning
IT Solutions page
1
Demand Marketing
Email campaign - February
+ Add another card
On Hold
Partners
CSS Rules
Happiness
Retail order
Mobile UI reboot
Google analytics data - Q1
1
+ Add another card
Done
Demand Marketing
Review Tech partner pages
Make sure sponsors are indicated for Tech Talk
Planning
Top 10 Trends list - Forbes
TBC Webinar: Ship Now, Not Later
Happiness
1:1 Nancy
Lead Gen Mandrill stats
+ Add another card
Questions
How do you ac
Illustrator?
Does Screenhe
When does the
increase - befc
week?
+ Add another

Kanban

- Projects that are created with Kanban are overseen by the Kanban Board.
- The Kanban Board is divided into multiple columns (which segments tasks into three columns: “To Do,” “Doing,” and “Done.”) to properly depict the process flow for the development of software.
- Such a process helps in enhancing the visibility among the team members because each team member can see what is happening and what isn't so that they prepare themselves for the upcoming task for final delivery of the product.
- It should be known this method will require thorough transparency as well as interaction between the team members so that proper development of the product can take place.

Kanban

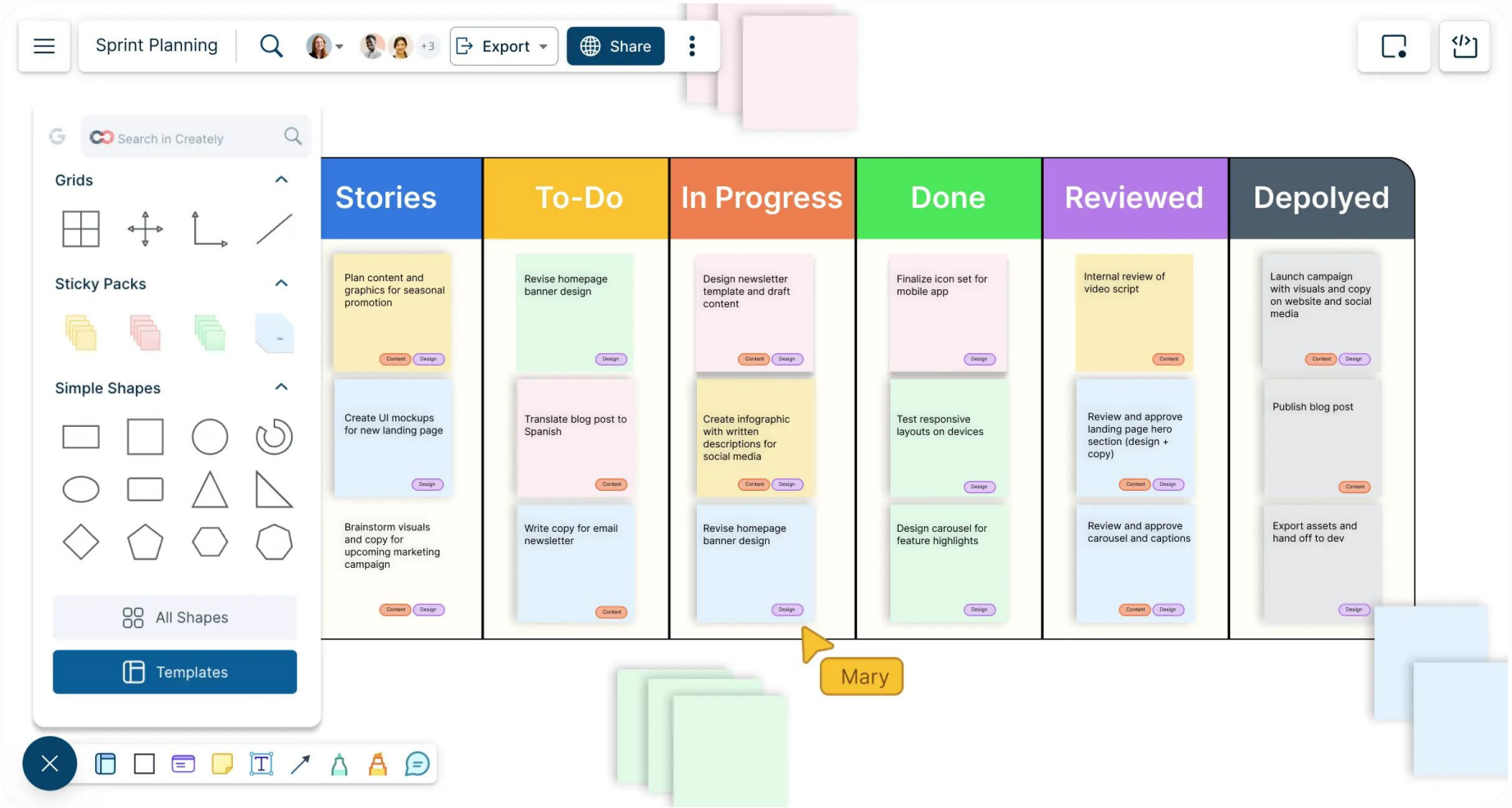
Kanban works on the three following principles:

1. **Visualizing the work that is to be performed at any given time.**
2. **Creating boundaries regarding the amount of work that is to be done or completed, so that the team doesn't over-commit additional work.**
3. **Boosting workflow when a specific task is about to be completed so that the next item can be added to the queue.**

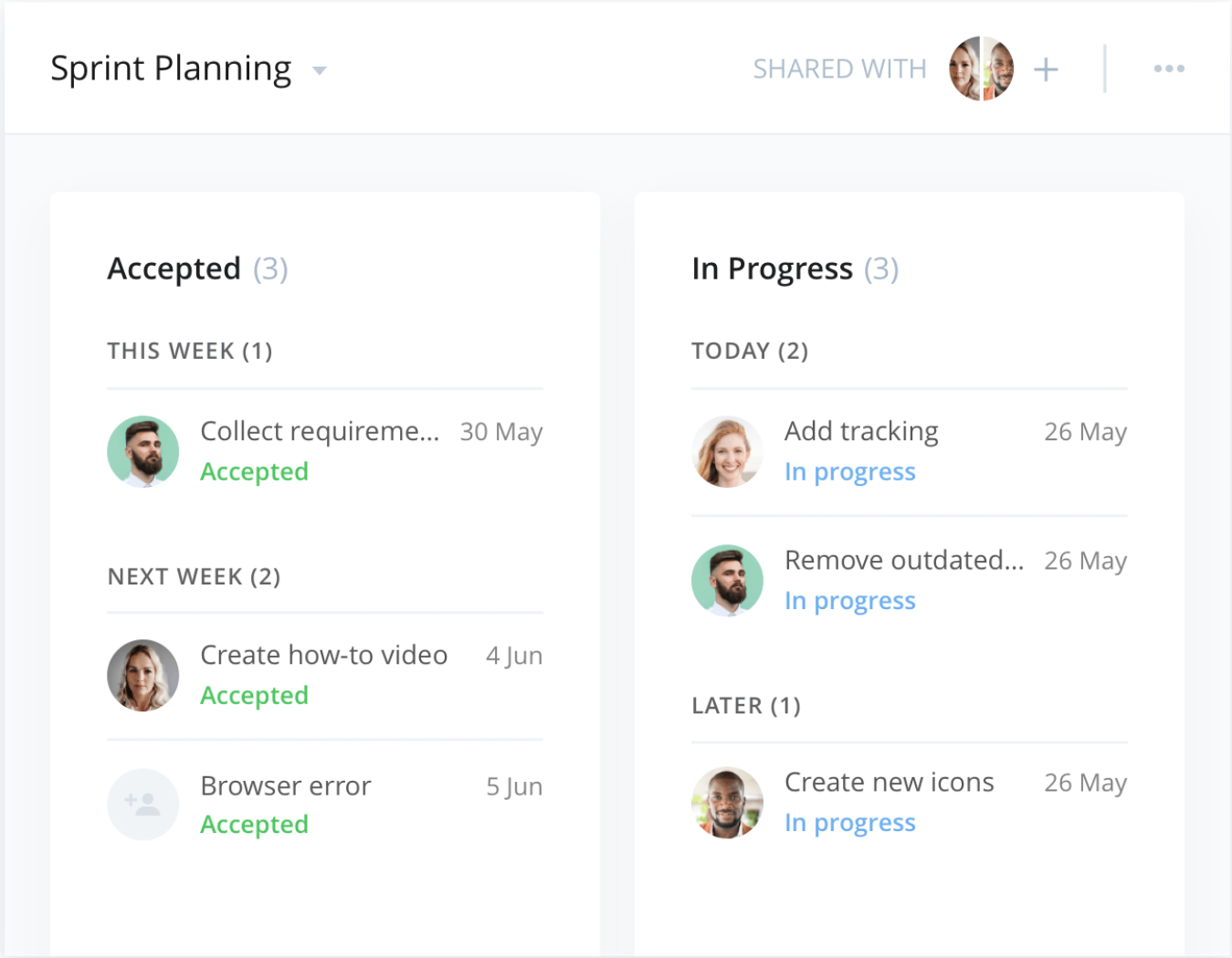
Scrum

- Scrum is one of the most popular Agile frameworks out there.
- However, unlike Kanban, Scrum focuses on breaking down a single project into multiple parts, known as 'Sprints', where only one Sprint will be planned & managed at a time.
- Scrum also includes unique project roles such as Product Owner, Scrum Master, and Developers.
- Scrum generally utilizes a Scrum Board, which is similar to the Kanban Board, and thereby group various tasks into multiple columns based on their overall progress.

Scrum


Sprint Planning
Export
Share
Search in Creately
Grids
Sticky Packs
Simple Shapes
All Shapes
Templates
Stories
To-Do
In Progress
Done
Reviewed
Depolyed
Plan content and graphics for seasonal promotion
Create UI mockups for new landing page
Brainstorm visuals and copy for upcoming marketing campaign
Revise homepage banner design
Translate blog post to Spanish
Write copy for email newsletter
Design newsletter template and draft content
Create infographic with written descriptions for social media
Revise homepage banner design
Finalize icon set for mobile app
Test responsive layouts on devices
Design carousel for feature highlights
Internal review of video script
Review and approve landing page hero section (design + copy)
Review and approve carousel and captions
Launch campaign with visuals and copy on website and social media
Publish blog post
Export assets and hand off to dev
Mary

Scrum


Sprint Planning
SHARED WITH
Accepted (3)
THIS WEEK (1)
Collect requireme... 30 May
Accepted
NEXT WEEK (2)
Create how-to video 4 Jun
Accepted
Browser error 5 Jun
Accepted
In Progress (3)
TODAY (2)
Add tracking 26 May
In progress
Remove outdated... 26 May
In progress
LATER (1)
Create new icons 26 May
In progress

Scrum

The process of the Scrum framework is as follows:

- **The Product Owner will create an estimated wish list, which will be identified as the Product Backlog.**
- **The Scrum Team will take a few items from the top of the list and makes a plan how to turn them into product Increments; thereby denote it as the Sprint Backlog. The team will then work towards completing the current Sprint Backlog.**
- **The daily progress of the work will be synchronized via a meeting known as Daily Scrum.**
- **The Scrum Master will be responsible for maintaining the focus of the team.**
- **At the end of each Sprint, the Increments will be reviewed and feedback will be provided before the next Sprint is initiated.**

Extreme Programming (XP)

- Extreme Programming is one of the foremost Agile frameworks out there.
- In this procedure, both the Developers and customers should exhibit a high degree of participation.
- The customers will be responsible for inspiring additional development of a product by showcasing the most useful features, via multiple testimonials.

- On the other hand, the Developers will base every set of software updates according to the feedback received from the customer while also testing new innovative features every few weeks.

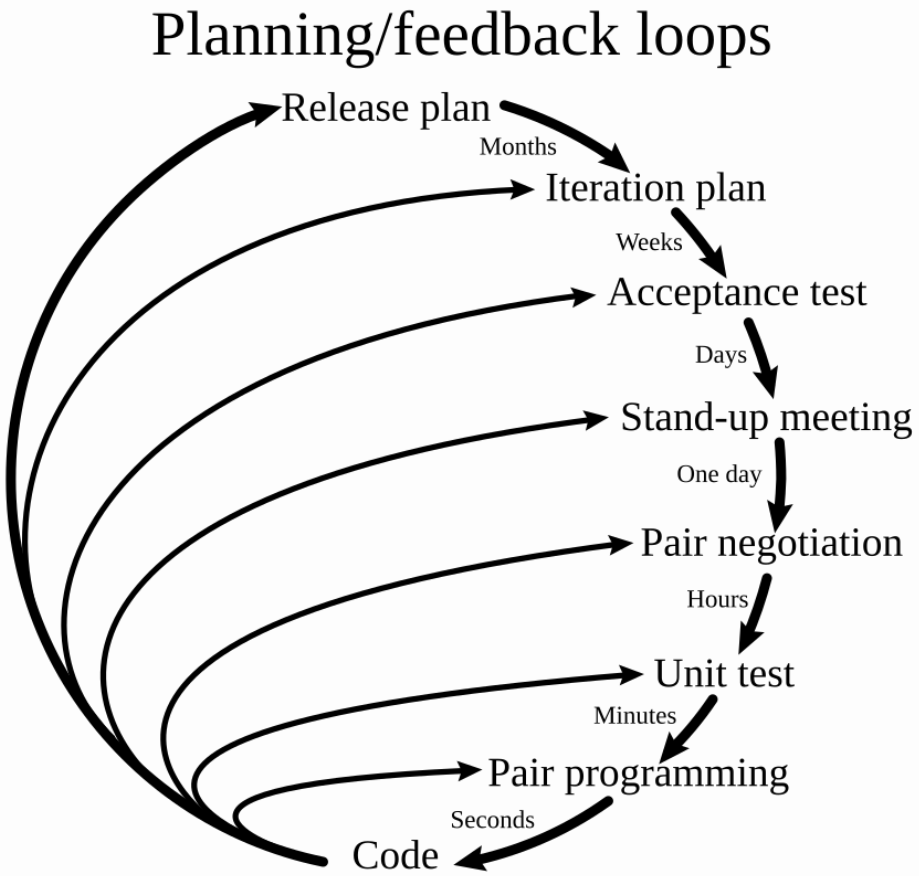
Extreme Programming (XP)

- XP has its share of benefits & drawbacks. On the meritorious side, XP will always involve a higher level of collaboration along with a very minimal amount of documentation. Moreover, it's a persistent & efficient delivery model.

- XP suits teams with various skill levels, mainly when junior and senior programmers collaborate. It also benefits teams facing tight deadlines, limited budgets, and frequent changes in agile project management software requirements.

- However, XP may not be the best choice for remote teams. It works best with small teams that are co located in the same physical location.

Extreme Programming (XP)



Planning and feedback loops in extreme programming

Lean Software Development (LSD)

- Lean Software Development is an agile framework that makes a specialty of handing over costs to clients with the aid of eliminating waste and maximizing efficiency.

- It originated in the manufacturing enterprise however has been adapted to software development. Lean emphasizes non-stop development, glide, and pull-based structures.

- It additionally promotes a culture of respect for humans, continuous getting-to-know, and consumer consciousness.

Lean Software Development (LSD)

Features of Lean

- **Eliminating Waste:** Focuses on casting off non-fee-including sports and lowering inefficiencies.
- **Value Stream Mapping:** Identifies and optimizes the price circulate, from customer want to product shipping.
- **Pull System:** Work is pulled based totally on consumer calls to prevent overproduction.
- **Kaizen (Continuous Improvement):** Teams always are trying to find ways to improve tactics and decrease waste.

Features of Lean

- Pair Programming is an Agile software development practice where two programmers work together at one workstation:
- Driver → writes the code
- Navigator → reviews, suggests improvements, and thinks strategically

Real world case studies

Agile Methodology	Scope (Where It Is Best Used)	Real-World Case Study	Application in Practice	Key Result
Scrum	Product development with evolving requirements, team collaboration, iterative delivery	Spotify	Organized teams into Scrum squads working in short sprints	Faster feature delivery and adaptability
XP (Extreme Programming)	Software projects requiring high quality, frequent updates, strong technical practices	IBM	Used pair programming and test-driven development	Improved code quality and fewer defects
Lean	Process improvement, cost reduction, large organizational workflows	Toyota IT	Removed non-value activities and optimized development flow	Reduced development time and cost
Kanban	Continuous work environments like maintenance and support systems	Microsoft	Visualized workflow using Kanban boards with WIP(Work In Progress Limit) limits	Better task management and quicker issue resolution


thank
you