AGILE

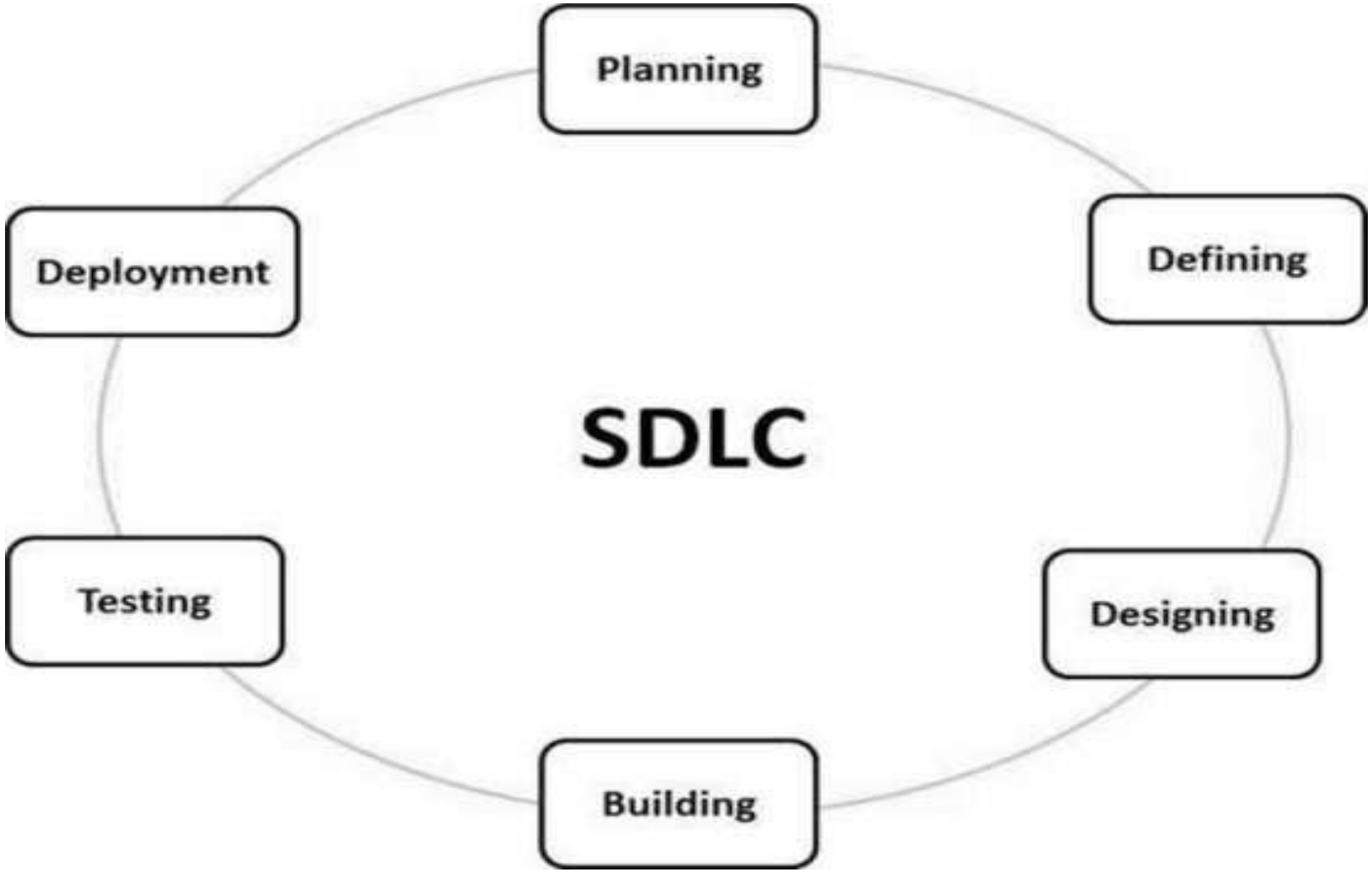
iamneo

SDLC(SOFTWARE DEVELOPMENT LIFE CYCLE)

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

What is SDLC

- SDLC is a process followed for a software project, within a software organization.
- It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software.


Planning
Defining
Designing
Building
Testing
Deployment
SDLC

PLANNING AND REQIUREMENT ANALYSIS

- Requirement analysis is the most important and fundamental stage in SDLC.
- It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry.
- This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.
- Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage.

Defining Requirements

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

Designing the Product Architecture

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

Building or Developing the Product

- The programming code is generated as per DDS during this stage.
- If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.
- Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code.
- Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding.
- The programming language is chosen with respect to the type of software being developed.

Testing the Product

- The testing activities are mostly involved in all the stages of SDLC.
- Product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Deployment in the Market and Maintenance

- Once the product is tested and ready to be deployed it is released formally in the appropriate market.
- Sometimes product deployment happens in stages as per the business strategy of that organization.
- The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).
- After the product is released in the market, its maintenance is done for the existing customer base.

SDLC Models

- There are various software development life cycle models defined and designed which are followed during the software development process.
- These models are also referred as Software Development Process Models".
- Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

- ❑ Waterfall Model
- ❑ Iterative Model
- ❑ Spiral Model
- ❑ V-Model
- ❑ Big Bang Model

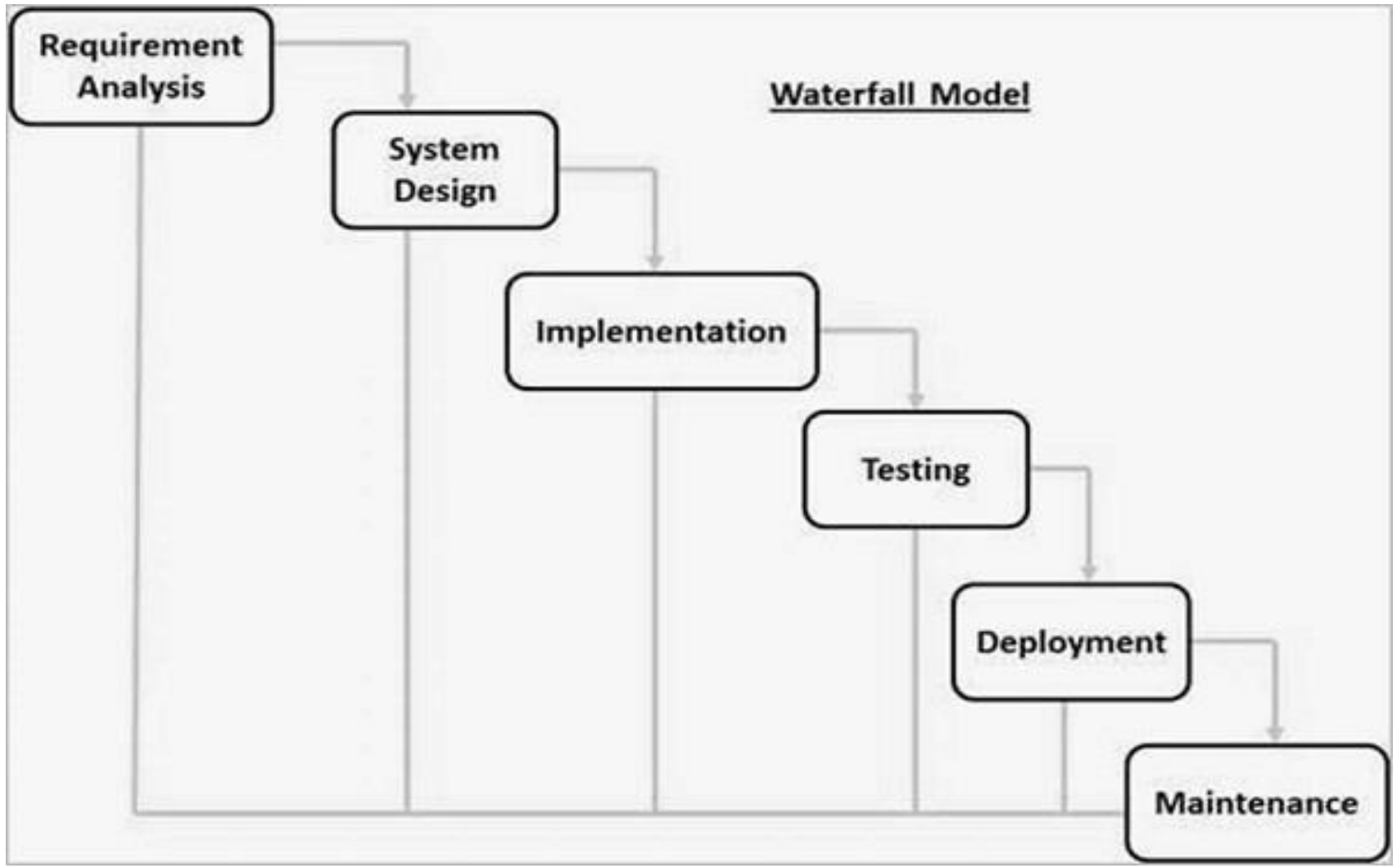
WATER FALL MODEL

- The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**.
- It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

- The Waterfall model is the earliest SDLC approach that was used for software development.
- The waterfall Model illustrates the software development process in a linear sequential flow.
- This means that any phase in the development process begins only if the previous phase is complete.
- The phases do not overlap.

Waterfall Model - Design

- Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project.
- In "The Waterfall" approach, the whole process of software development is divided into separate phases.
- In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.


Waterfall Model
Requirement Analysis
System Design
Implementation
Testing
Deployment
Maintenance

- **Requirement Gathering and analysis**

All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

- **System Design**

The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

- **Implementation**

With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

- **Integration and Testing**

All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

- **Deployment of system**

Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

- **Maintenance**

There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Waterfall Model - Application

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

ADVANTAGES

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

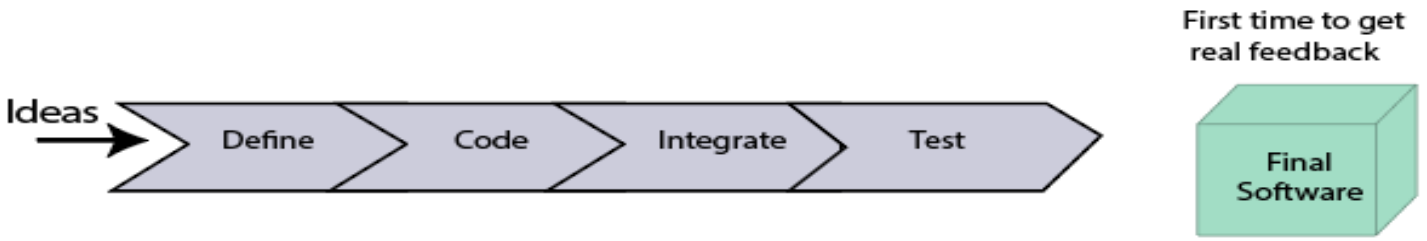
DISADVANTAGES

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

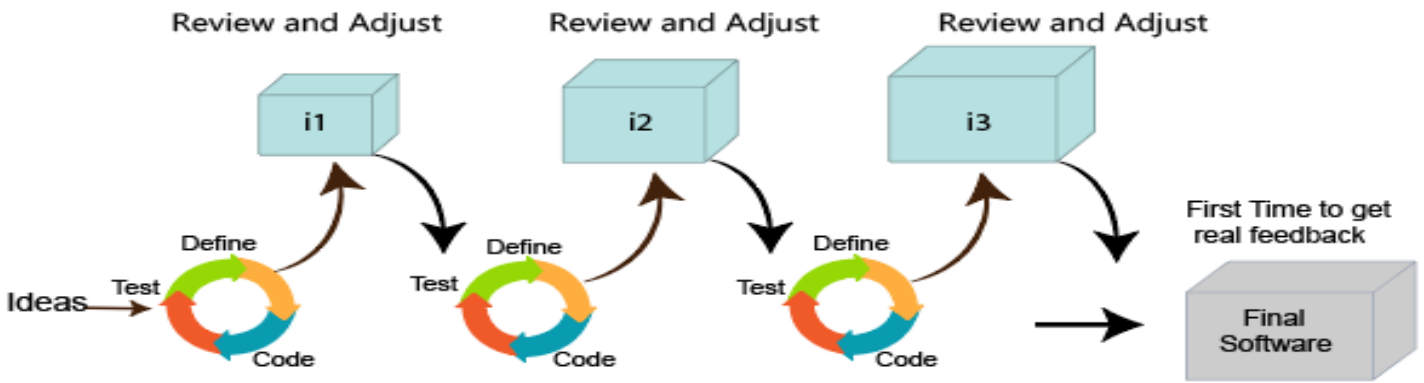
AGILE METHODOLOGY

- An agile methodology is an iterative approach to software development.
- Each iteration of agile methodology takes a short time interval of 1 to 4 weeks.
- The agile development process is aligned to deliver the changing business requirement.
- It distributes the software with faster and fewer changes.
- The single-phase software development takes 6 to 18 months.
- In single-phase development, all the requirement gathering and risks management factors are predicted initially.

AGILE METHODOLOGY



Traditional Method



Agile Method

iamneo

AGILE DEVELOPMENT MODEL PHASES

1. Requirements gathering
2. Design the requirements
3. Construction/ iteration
4. Testing/ Quality assurance
5. Deployment
6. Feedback

- **Requirements gathering:**

In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

- **Design the requirements:**

When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system

- **Construction/ iteration:**

When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

- **Testing:**

In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

- **Deployment:**

In this phase, the team issues a product for the user's work environment.

- **Feedback:**

After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

Agile Testing Methods:

- Scrum
- Crystal
- Dynamic Software Development Method(DSDM)
- Feature Driven Development(FDD)
- Lean Software Development
- Extreme Programming(XP)

Agile manifesto

- In February 2001, at the Snowbird resort in Utah, a team of 17 software developers met to discuss lightweight development methods. The result of their meeting was the following Agile Manifesto for software development.

Principles of agile

- **Customer Satisfaction:**

  Manifesto provides high priority to satisfy the costumer's requirements. This is done through early and continuous delivery of valuable software.

- **Welcome Change:**

  Making changes during software development is common and inevitable. Every changingrequirement should be welcome, evenin the late development phase. Agile process works to increase the customers' competitive advantage

- **Deliver the Working Software:**

Deliver the working software frequently, ranging from a few weeks to a few months with considering the shortest time period.

- **Collaboration:**

Business people (Scrum Master and Project Owner) and developers must work together during the entire life of a project development phase.

- **Motivation:**

Projects should be build around motivated team members. Provide such environment that supportsindividual team members and trust them. It makes them feel responsible for gettingthe job donethoroughly.

- **Face-to-face Conversation:**

Face-to-face conversation betweenScrum Master anddevelopment team and between the Scrum Master and customers for the most efficient and effective method of conveying information to and within a development team.

- **Measure the Progress as per the Working Software:**

The working software is the key and primary measure of the progress.

- **Maintain Constant Pace:**

The aim of agile development is sustainable development. All the businesses and users should be able to maintain a constant pace with the project.

- **Monitoring:**

  Pay regular attention to technical excellence and good design to maximize agility.

- **Simplicity:**

  Keep things simple and use simple terms to measure the work that is not completed.

- **Self-organized Teams:**

  The Agile team should be self-organized. They should not be depending heavily on other teams because the best architectures, requirements, and designs emerge from self-organized teams

- **Review the Work Regularly:**

The work should be reviewed at regular intervals, so that the team can reflect on how to become more productive and adjust its behavior accordingly.

AGILE Values

- There are four important values in the agile method.
    1. Individuals and interactions over processes and tools
    2. Working software over comprehensive documentation.
    3. Customer collaboration over contract negotiation.
    4. Responding to change over following a plan.

Individual and interaction over processes and tool

- There is no matter about tools and the process. It is the team work and the way you work together for successful product completion.
- The team and their ability to communicate efficiently and effectively is more important than the processes and the tools.

Working software over comprehensive documentation

- In the traditional product development processes often required extensive documentation before a single line of code was written.
- In this method getting software in the hands of customers is the highest priority.
- After this we will get feedback from the real users to develop the product features.

Customer collaboration over Contract Negotiation.

- In agile the importance of customer centric product development practices over product centric approaches.
- While contracts will always have their place in business, a list of the things you're offering your customer is no replacement for actually communicating with them about what their needs are and where their challenges are.
- Traditional product-centric processes allowed contracts to dictate what was delivered in the end, which left a lot of room for mismatched expectations. But in agile encourage building a continuous customer feedback loop into development cycles.

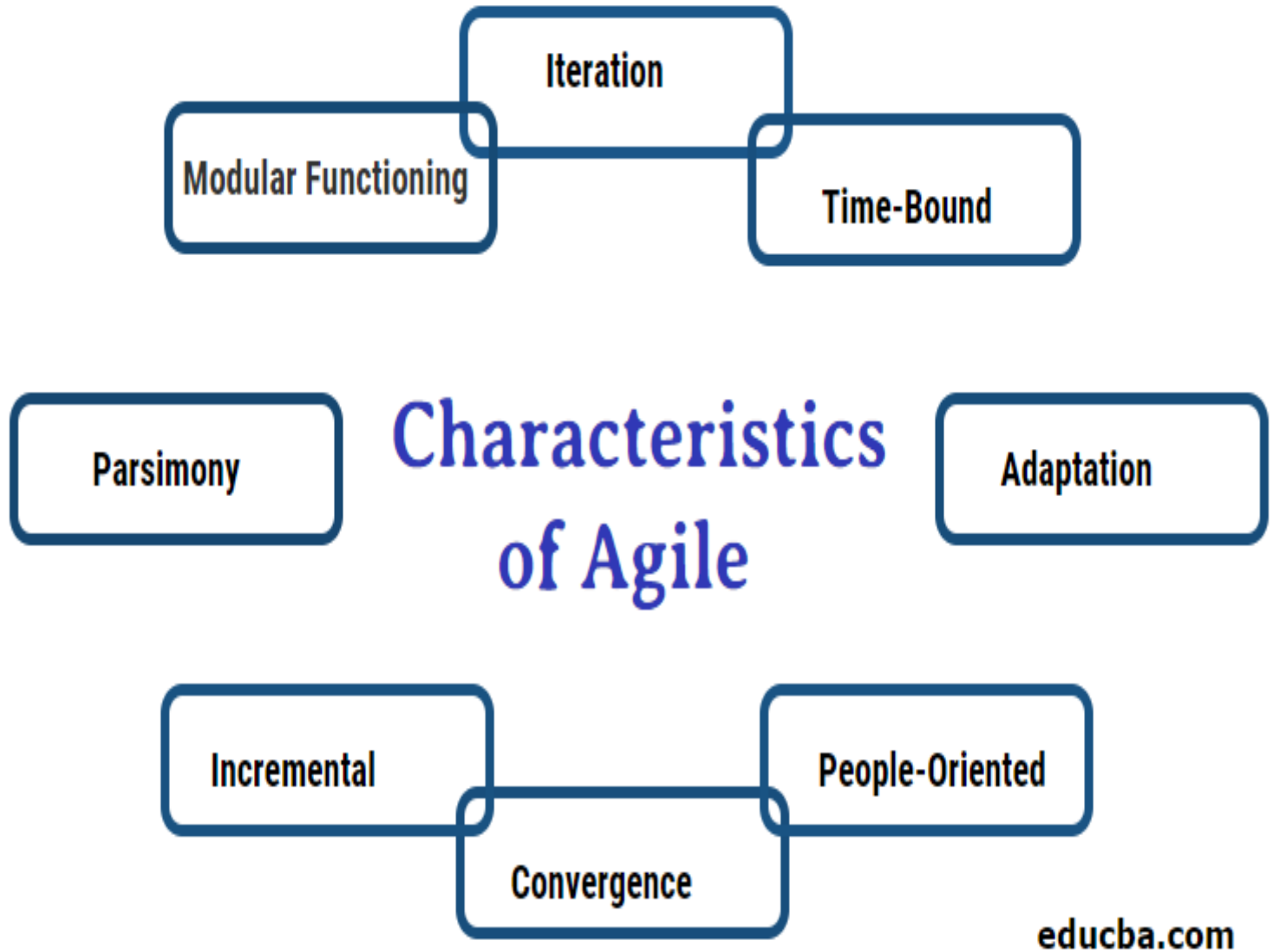
Responding to change over Following plan

- An important benefit of the agile methodology is that it encourages frequent reviewing and retooling of current plans based on new information that the team is continually gathering and analyzing.
- The agile methodology lets a product team adjust its priorities and plans whenever doing so makes strategic sense.
- These teams do not get stuck in an outdated plan simply because they have committed to seeing it through.

Agile Characteristics

- The software development life cycle is known for trying different approaches based on requirements for project development.
- Software development mainly considerate two points that are an emphasis on process and the quality of the software and process itself.

- Agile development method works by dividing the task into small sub-tasks termed as increments and builds the project's ultimate deliverables in small increments by repeating basic steps over and over.
- It requires less planning during the development process, it mainly works for short term projects, the team efforts make a project successful.
- The agile development process is lightweight and provides incremental and continuous delivery.
- Agile development teams are working on a full-time basis and persist project to project.


Iteration
Modular Functioning
Time-Bound
Parsimony
Characteristics of Agile
Adaptation
Incremental
People-Oriented
Convergence
educba.com

MODULAR FUNCTIONING

- Modularity is considered one of the key elements of a good process.
- Modularity is the element that allows the components to break down and that broken component is called activities.
- The software development process is just the set of activities that frames or transforms the vision of the software system into reality.
- Agile Software development process makes use of good tools and is wielded with good software craftsman who is well known to apply those at the right place and right time.
- These can not be utilized for the production line for manufacturing software products.

ITERATION

- The agile software development process acknowledges the working on attempting wrong before its correct.
- In agile processes focus on small cycles.
- Each cycle has a task of defined activities must be completed in a perfect manner.
- These task have a time slot of a week, from beginning to completion of a project.
- The iteration that is single cycle may or may not get a 100% correct element.
- Due to this reason one short cycle repeated many times until getting the correct result.

Time-Bound

- Software development comes with time limits.
- The developing team will give a delivery date of the project to the customer for tracking the status of the project.
- Iteration plays a good role in keeps time limit between the 1 to 6 weeks.
- There are more chances to it may schedule all the activities in a single iteration else wise only the activities to reach the goals which were set at the beginning of the iteration.
- Rescheduling or functionality reduction can be done to deliver the project on time, on the allotted time.

Parsimony

- Agile software development is considered an upgraded version of the traditional approach with time constrains add on.
- Impossible deadlines are not attempted for rapid delivery, each phase of development is kept in mind as this attempt may take away the quality from the product and that’s a big NO.
- Instead, agile approach focus on parsimony keeps the activities to minimal and only necessary to mitigate risks and achieve their goal.

ADAPTION

- In the development or iterations there are higher chances of unknown risks they may be exposed.
- The agile approach is introduced for overcome the unknown risk.
- If there are changes in different results during the functionality, new activities or functionality can be added to reach the goal.

INCREMENTAL

- Agile system is not built entirely at once, the system is partitioned and look out for increments that can be parallelly developed, at a different time and a different rate.
- Each increment is tested independently and if found ok then all are integrated into the one system for the result.

CONVERGENCE

- It means that the risks are attacked actively because it is worth to know the risks.
- This takes the system closer to the results.
- Risks solving during each iteration is one of the great processes that leads to a successful iteration.

PEOPLE ORIENTED

- The process is known for its priority of the customers over process and technology.
- The involvement of the customer is done organically.
- The developers evolve through adaptation and are empowered to raise their productivity and performance.
- These developers are very aware of dealing with the changes in the system at every stage.

COLLABRATION

- It has a very practical approach for face-to-face discussion.
- Discussion may between customers or with a team members itself.
- Good communication may perform a important role in the success of the software development field.
- The risk may occur the miscommunication is higher when the system is developed into pieces.
- This is important for every member should know how to fit the two piece together for creating a product.

Advantages of agile model

- In agile methodology the delivery of software in unremitting.
- The customers are satisfied because every sprint working feature of the software is delivered to them.
- Customers can have a look of the working feature which fulfilled their expectation.
- In agile methodology the daily interacctions are required between the business people and the developers.
- In this methodology attention is paid to the good design of the product.
- Changes in the requirements are accepted even in the later stages of the devolpment.
- An agile/scrum approach can improve organizational synergy by breaking down organizational barriers and developing a sprit of trust and partnership around organizational goals.

Disadvantages of Agile model

1. It is not useful for small development projects.
2. There is a lack of intensity on necessary designing and documentation.
3. It requires an expert project member to take crucial decisions in the meeting.
4. Cost of Agile development methodology is slightly more as compared to other development methodology.
5. The project can quickly go out off track if the project manager is not clear about requirements and what outcome he/she wants.

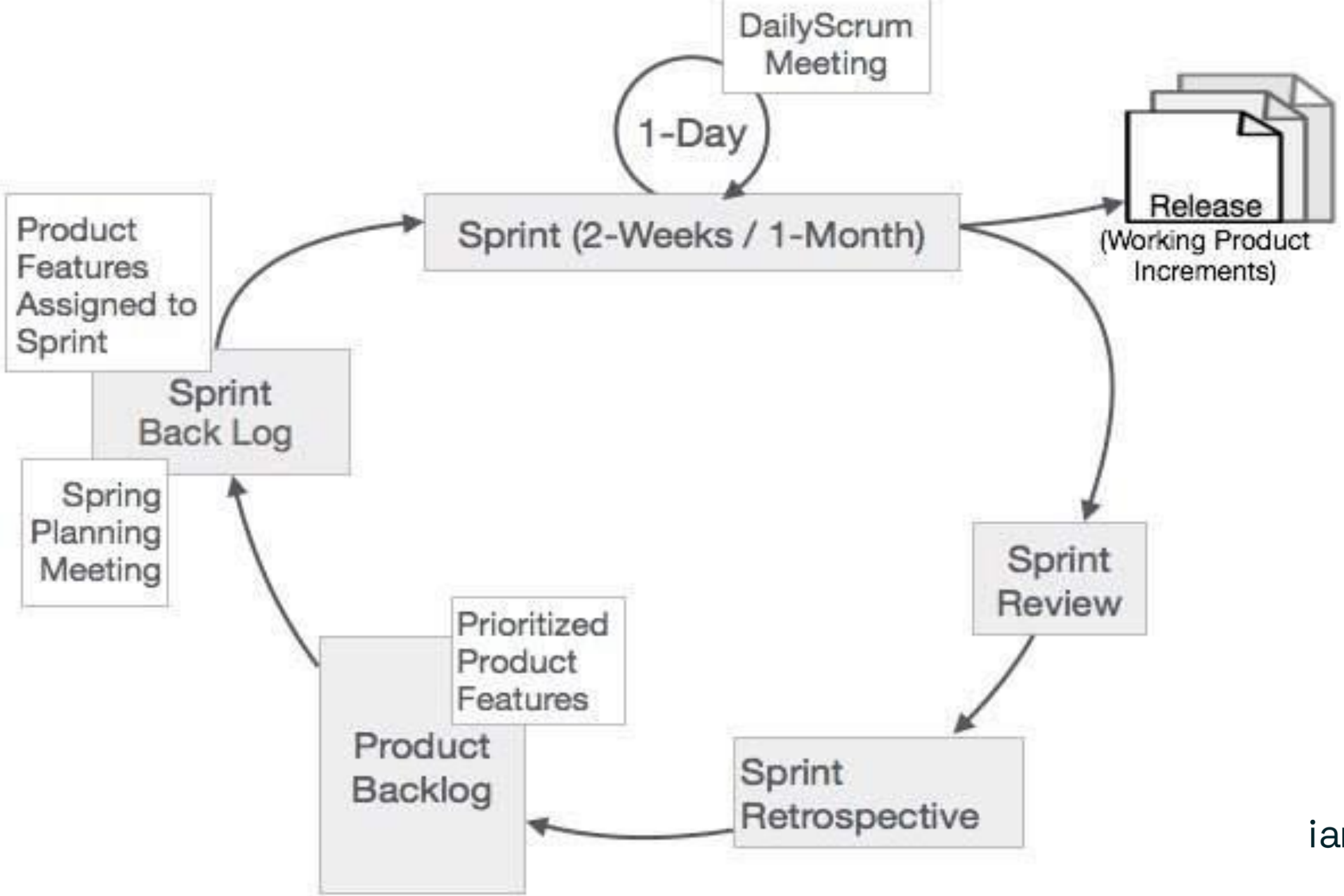
Difference between agile and waterfall model

- Waterfall is a Linear Sequential Life Cycle Model, whereas Agile is a continuous iteration of development and testing in the software development process.
- In Agile vs Waterfall difference, the Agile methodology is known for its flexibility, whereas Waterfall is a structured software development methodology.
- Comparing the Waterfall methodology vs Agile, which follows an incremental approach, whereas the Waterfall is a sequential design process.
- Agile performs testing concurrently with software development, whereas in Waterfall methodology, testing comes after the “Build” phase.
- Agile allows changes in project development requirements, whereas Waterfall has no scope of changing the requirements once the project development starts.

SCRUM

- Scrum is a process framework that has been used to manage complex product development since the early 1990s.
- Scrum is an efficient framework within which you can develop software with teamwork. It is based on agile principles.
- Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.
- Scrum is not a process or a technique for building products; rather, it is a framework within which you can employ various processes and techniques.

- Scrum makes clear the relative efficacy of your product management and development practices so that you can improve.
- The Scrum framework consists of Scrum Teams and their associated roles, events, artifacts, and rules.
- Each component within the framework serves a specific purpose and is essential to Scrum’s success and usage.
- The rules of Scrum bind together the events, roles, and artifacts, governing the relationships and interaction between them.


DailyScrum Meeting
1-Day
Release
(Working Product Increments)
Product Features Assigned to Sprint
Sprint (2-Weeks / 1-Month)
Sprint Back Log
Spring Planning Meeting
Sprint Review
Prioritized Product Features
Product Backlog
Sprint Retrospective

ROLES

There are three main roles in SCRUM

1. SCRUM MASTER
2. PRODUCT OWNER
3. DEVELOPMENT TEAM

SCRUM MASTER

- Making the process run smoothly
- Removing obstacles that impact productivity
- Organizing and facilitating the critical meetings
- Acts as the middle man between on-shore team and off-shore team

PRODUCT OWNER

- Expressing Product Backlog items clearly.
- Ordering the Product Backlog items to best achieve goals and missions.
- Optimizing the value of the work the Team performs.
- Ensuring that the Product Backlog is visible, transparent, and clear to all, and shows what the Team will work on further.
- Ensuring that the Team understands items in the Product Backlog to the level needed.

DEVELOPMENT TEAM

- Help in sprint planning and goal setting
- Lend expertise to program, design, or improve products
- Use data to find best practices for development
- Test products and prototypes, plus other forms of quality assurance

SCRUM MASTER SERVICES TO PRODUCT OWNER

- Finding techniques for effective Product Backlog management.
- Helping the Scrum Team understand the need for clear and concise Product Backlog items.
- Understanding product planning in an empirical environment.
- Ensuring that the Product Owner knows how to arrange the Product Backlog to maximize value.
- Understanding and practicing agility.
- Facilitating Scrum events as needed.

ScrumMaster Services to the Scrum Team

- Coaching the Scrum Team in self-organization and cross-functionality.
- Helping the Scrum Team to create high-value products.
- Removing impediments to the Scrum Team’s progress.
- Facilitating Scrum events as requested or needed.
- Coaching the Scrum Team in organizational environments in which Scrum is not yet fully adopted and understood.

ScrumMaster Services to the Organization

- Leading and coaching the organization in its Scrum adoption.
- Planning Scrum implementations within the organization.
- Helping employees and stakeholders understand and enact Scrum and empirical product development.
- Causing change that increases the productivity of the Scrum Team.
- Working with other Scrum Masters to increase the effectiveness of the application of Scrum in the organization.

SCRUM ARTIFACTS

- Scrum Artifacts provide key information that the Scrum Team and the stakeholders need to be aware of for understanding the product under development, the activities done, and the activities being planned in the project.
- Process framework
  - Product Backlog
  - Sprint Backlog
  - Burn-Down Char
  - Increment

PRODUCT BACKLOG

- The Product Backlog is an ordered list of features that are needed as part of the end product and it is the single source of requirements for any changes to be made to the product.
- The Product Backlog lists all features, functions, requirements, enhancements, and fixes that constitute the changes to be made to the product in future releases.
- Product Backlog items have the attributes of a description, order, estimate, and value.
- These items are normally termed as User Stories.
- The Product Owner is responsible for the Product Backlog, including its content, availability, and ordering.

SPRINT BACKLOG

- The Sprint Backlog is the set of Product Backlog items selected for the Sprint, plus a plan for delivering the product Increment and realizing the Sprint Goal.
- The Sprint Backlog is a forecast by the Team about what functionality will be made available in the next Increment and the work needed to deliver that functionality as a working product Increment.
- The Sprint Backlog is a plan with enough detail that can be understood but the Team to track in the Daily Scrum.

INCREMENT

- The Increment is the sum of all the Product Backlog items completed during a Sprint combined with the increments of all previous Sprints.
- At the end of a Sprint, the new Increment must be a working product, which means it must be in a useable condition.
- It must be in working condition regardless of whether the Product Owner decides to actually release it.

Sprint Burn-Down Chart

- At any point in time in a Sprint, the total work remaining in the Sprint Backlog can be summed.
- The Team tracks this total work remaining for every Daily Scrum to project the likelihood of achieving the Sprint Goal.
- By tracking the remaining work throughout the Sprint, the Team can manage its progress.

Scrum user stories

- The User Stories are commonly used to describe the product features and will form part of the Scrum Artifacts – **Product Backlog** and **Sprint Backlog**.
- In Scrum projects, the Product Backlog is a list of user stories. These User Stories are prioritized and taken into the Sprint Backlog in the Sprint Planning Meeting.

User Story

Title:	Priority:	Estimate:
User Story: As a [description of user], I want [functionality] so that [benefit].		
Acceptance Criteria: Given [how things begin] When [action taken] Then [outcome of taking action]		

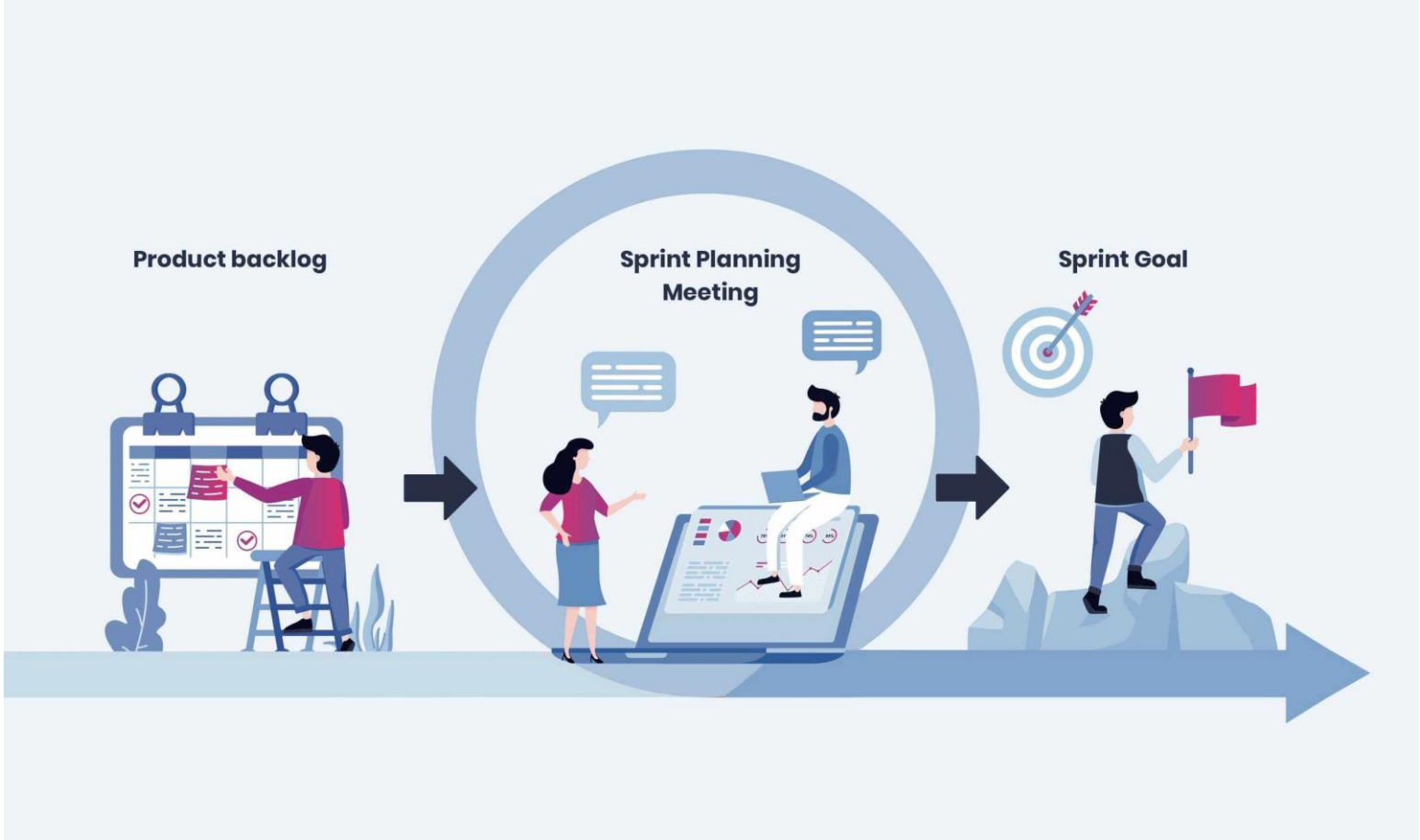
Release planning

- Release planning is the creation of a plan that describes how a product will be released into the market.
- The plan should take into account all of the factors that could affect the success of the product launch, including team availability, deadlines, and dependencies.
- By creating a plan that takes into account all the potential variables, developers can ensure that their product is released on time and meets all the necessary requirements.

SPRINT PLANNING MEETING

- In the Scrum agile framework, a sprint planning meeting is an event that establishes the product development goal and plan for the upcoming sprint, based on the team's review of its product backlog.
- In this there are two important strategic items:
  - **The sprint goal**
  - **The sprint backlog.**

SPRINT PLANNING MEETING:



iamneo

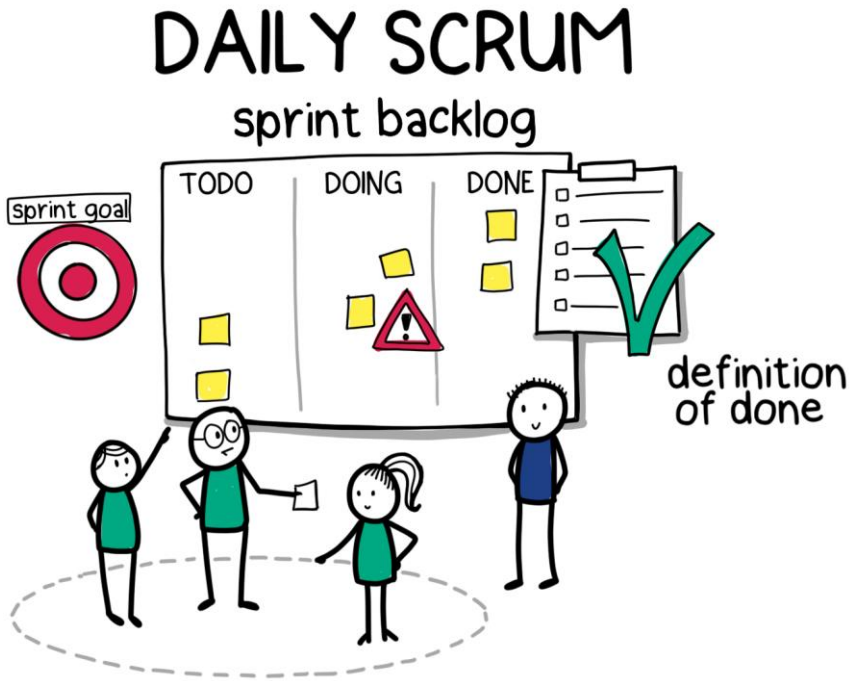
- The objective of sprint planning is to work out the key details regarding the team's planned work during the next sprint. With that in mind, the sprint team should plan to address at least the following issues during this meeting.

In this we have some meeting agenda.

1. **Decide on the team's overall strategic objective for the next sprint. (This will be represented as the one- or two-sentence sprint goal.)**
2. **Review the product backlog and discuss which items belong on the next sprint backlog and why.**
3. **Call for a team consensus on the proposed sprint goal and backlog items (led by the scrum master).**
4. **Discuss team capacity.**
5. **Discuss known issues that could disrupt or slow progress on the sprint backlog.**
6. **Assign the new sprint backlog's tasks, according to skill sets, capacity, and other relevant criteria.**
7. **Estimate the timeframes for each of the tasks assigned and agree on what "done" will look like for each item.**
8. **Confirm the timeframe of the upcoming sprint.**
9. **Open the meeting to sprint-related questions. (The product owner should be responsible for coordinating this step, to ensure the discussion stays on track.)**

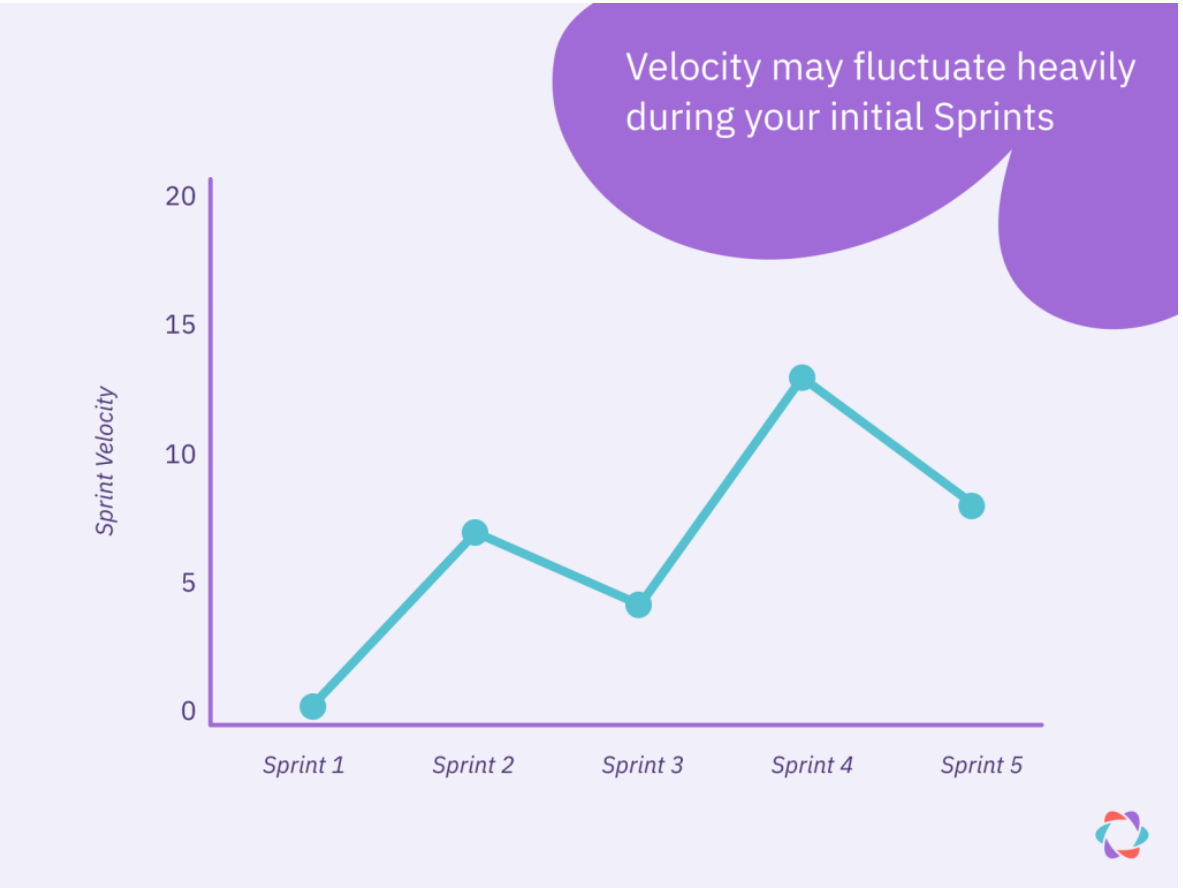
Daily scrum

- The purpose of the Daily Scrum is to inspect progress toward the Sprint Goal and adapt the Sprint Backlog as necessary, adjusting the upcoming planned work.
- The Daily Scrum is a 15-minute event for the Developers of the Scrum Team. To reduce complexity, it is held at the same time and place every working day of the Sprint.
- If the Product Owner or Scrum Master are actively working on items in the Sprint Backlog, they participate as Developers.



SPRINT VELOCITY:

- Velocity is **the number of story points completed by a team in one Sprint**.
- Some teams use different measurements, like hours or stories completed, to calculate their velocity. Whatever data you use, the concept remains the same.
- Velocity indicates how much work the team has finished in a Sprint.
- How do you calculate velocity in Scrum?
  Actual velocity is calculated by **dividing the total Story Points completed by the team by the number of Sprints**. For instance, if the Scrum Team has finished a total of 80 points over 4 Sprints then the actual velocity of the team would be 20 points per Sprint.



SPRINT VELOCITY = TOTAL STORY POINTS COMPLETED/NUM OF SPRINTS

Why do teams measure sprint velocity?

- Velocity can help teams make more accurate plans for their Sprints. That's because it helps teams understand how many story points they can generally complete in one Sprint.
- The number can also serve as a conversation starter when you set the stage in your retro meetings or be a warning signal for teams to investigate.

Velocity helps plan your Sprints:

During Sprint Planning, you can use velocity to calculate how much work you can reasonably take on for one Sprint. Many teams find value in calculating their average velocity over the past three Sprints. This can then serve as a cap for the number of story points bookmarked for the upcoming Sprint.

What is a sprint review meeting?

- In Agile project management, a sprint review is an informal meeting held at the end of a sprint, in which the Scrum team shows what was accomplished during this period.
- This typically takes the form of a demonstration of new features, with the goal of creating transparency, fostering collaboration, and generating feedback.

Definition of **DONE**

- It has been defined three stages
  1. User story
  2. Iteration
  3. Product release

USER STORY

- User story is a requirement which is formulated into few sentences.
- The user requirement is the everyday language of user.
- This user story should be completed within iteration
  1. All the related code and documentation have been checked-in.
  2. The product passed all the processes of unit test.
  3. All the processes of the acceptance test case have been moved.
  4. The product owner must have accepted the story.
  5. The help text (documentation) is written.

  With these process only the user story will done.

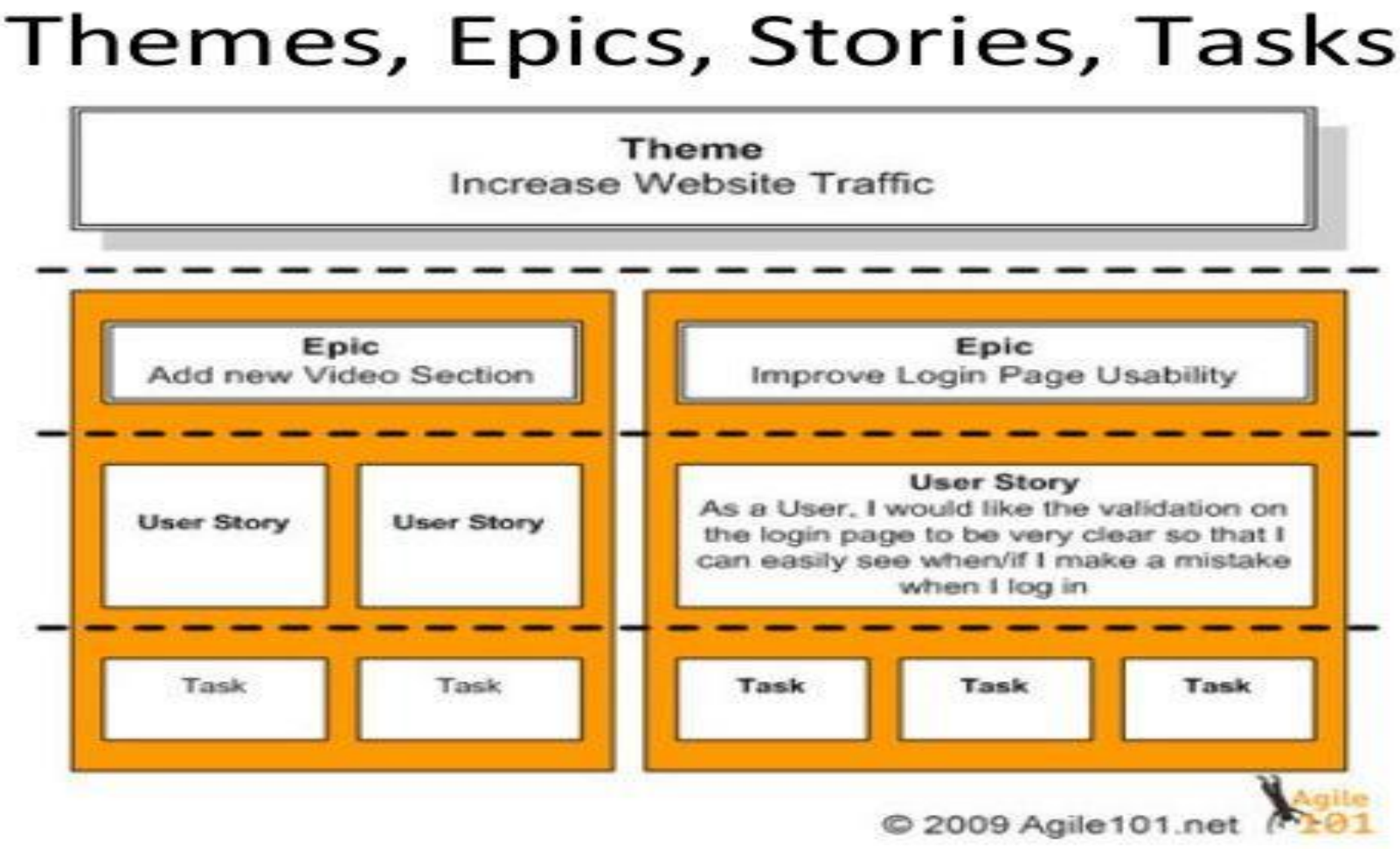
Iteration

- An iteration is a time-based collection of a user story.
- It works on the defected product and accepted within the release of a product.
- Iteration is defined at the time of the iteration planning meeting and completed within the iteration demo and review meeting.
- The iteration is also known as **Sprint.**
  - Performance of the product has been tested.
  - Product backup is complete.
  - User requirement has been accepted or moved for next iteration.
  - Defect product has been fixed or postponed for the next iteration.
- The repetition will happen because these reasons.

Release

- The product release is a major occasion that represents an internal and external delivery of work.
- It also tests the version of the product or system.
- The product release is done when:
  - The system is stress tested.
  - Performance is high.
  - Contain the security validation in the product.
  - Disaster recovery plan is tested.

Splitting user story into Task



iamneo

Task

- Task is a piece of activity that is required to get the story done.
- There are five efficient ways to split the story into task.
  1. Create meaningful task.
  2. Use the definition of done as a checklist.
  3. Create tasks that are right sized.
  4. Avoid explicitly outlining a unit testing task.
  5. Keep your task small.

Why to split user story into task?

- Allows the team to determine what is essential now.
- Prioritize and deliver other data later or not at all.
- Data sources may trigger complex integration.
- Data my be messy or unavailable.
- Data may need to be created by the team.
- Data may imports may require transformation or to support to complex format.
- External data dependencies can be complexities in the story.

Scrum task board

- Used to keep track and tasks and subtasks.
- Continually updated.
- Team members volunteer for tasks.
- Separated into 4 columns minimum.
  - User stories
  - To-Do
  - In-progress
  - Done

User stories

- Short descriptions of a feature from the perspective of the intended user.
- Explains the big tasks that will be split into subtasks in the other columns
- Has a specific format, shown below.
- “As a (user), I want (goal) so that (reason).”

Mediums

- There are mediums for a scrum task board.
- A task board can be physical, with sticky notes or index cards representing each item.

Planning Poker®

- Planning Poker® is a consensus-based estimating technique.
- Agile teams around the world use Planning Poker to estimate their product backlogs.
- Planning Poker can be used with story points, ideal days, or any other estimating unit.

What is planning poker

• Planning poker (also known as **Scrum** poker) is a consensus-based, gamified technique for estimatir mostly used to estimate effort or relative size of development goals in software development.

iamneo

Process in planning poker

1. To start a poker planning session, the product owner or customer reads an **agile** user story or describes a feature to the estimators.

For example:

“Customer logs in to the reservation system”
“Customer enters search criteria for a hotel reservation”

1. Team members of the group make estimates by playing numbered cards face-down to the table without revealing their estimate (Fibonacci values: 1,2,3,5,8,13,20,40)
2. Cards are simultaneously displayed
3. The estimates are then discussed and high and low estimates are explained
4. Repeat as needed until estimates converge

Fibonacci series and planning poker

- Planning Poker uses of the Fibonacci sequence to assign a point value to a feature or user story.
- The Fibonacci sequence is a mathematical series of numbers that was introduced in the 13th century and used to explain certain formative aspects of nature, such as the branching of trees.
- The series is generated by adding the two previous numbers together to get the next value in the sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, and so on.

For agile estimation purposes, some of the numbers have been changed, resulting in the following series: 1, 2, 3, 5, 8, 13, 20, 40, 100

Card(s)	Interpretation
0	Task is already completed.
1/2	The task is tiny.
1, 2, 3	These are used for small tasks.
5, 8, 13	These are used for medium sized tasks.
20, 40	These are used for large tasks.
100	These are used for very large tasks.
<infinity>	The task is huge.
?	No idea how long it takes to complete this task.
<cup of coffee>	I am hungry ☺

How do we estimate in story points

- While making a story point estimation during story mapping developers must consider three factors.

1. Complexity. How hard will it be to develop this feature?
2. Risk. This includes random changes, dependency on third parties, vague demands, and other uncertainties.
3. Repetition. How familiar are team members with the feature, and how monotonous will the required tasks be?

What goes into story points?

- Because story points represent the effort to develop a story, a team's estimate must include everything that can affect the effort. That could include:
    - The amount of work to do
    - The complexity of the work
    - Any risk or uncertainty in doing the work
- When estimating with story points, be sure to consider each of these factors. Let's see how each impacts the effort estimate given by story points.