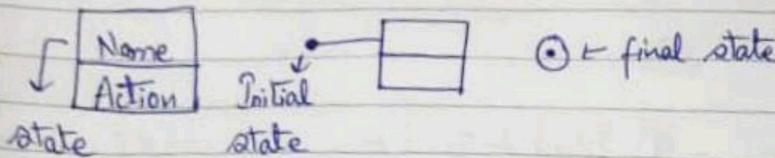
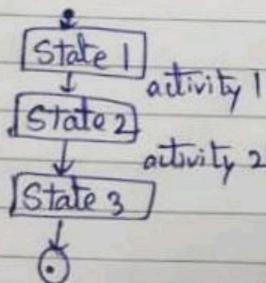


State Chart Diagram :- ① State ② Event ③ Transition ④ Decision Box
 ⑤ Initial / final state.

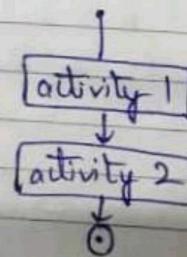


It is used to capture the behaviour of the software system. It can be used to model behaviour of class, package, subsystem or even an entire system. It is also called state machine | chart | transition diagram.

- ① State represent the condition of an object at instant of time. It is represented by rounded rectangle shape.
- ② The change from one state to another state is called transition.
- ③ A blank filled circle used to represent initial state. A blank circle with a dot used to represent final state.
- ④ Self Transition :- When state of object does not change upon occurrence of event called self transition state.



State Chart Diagram



Activity Diagram

→ The primary difference b/w state transition and activity diagram that state chart diagram used to model reactive system while activity used to

model nonreactive system.

- State chart focus on the internal and have to response to external events while activity diagram focus on sequence or activity involved in process and work flow.

Complex System → State Chart Diagram

Simple System → Activity Diagram

Types of Projects:

<u>Product</u>	<u>Service</u>
<ul style="list-style-type: none">• Creating a tangible item for customer.• Success is checked by product sold & profit made.• Planning includes market research & product design.• Management includes managing scope & budget.• Challenges includes product development delays & market competition.• Solid marketing strategy is must.	<ul style="list-style-type: none">• Provides intangible services to customers.• Success is measured by customer satisfaction & loyalty.• Planning includes consideration of customer needs & service delivery.• Management includes service quality & customer satisfaction.• Challenges includes service delivery delays & customer complaints.• Effective communication & pro-active problem solving is must.

what is burn up in Software engineering

A "burn-up chart" is a graphical representation that shows the progress of work completed over time in a software project, often used in Agile methodologies like Scrum. It consists of a graph with time on the horizontal axis and the amount of work completed on the vertical axis. As time progresses, the chart illustrates how the total work is accumulating. The goal is to "burn up" towards a predefined scope or target. Burn-up charts are valuable tools for tracking and communicating project progress, providing a visual representation to help identify trends, track velocity, and make informed decisions about project timelines and scope.

what is burn down in Software engineering

A "burn-down" chart in software engineering is a visual representation that illustrates the remaining work over time in a project. It helps teams track their progress by showing how the amount of work decreases (or "burns down") as time passes. This chart is commonly used in Agile methodologies like Scrum to monitor whether a team is on target to complete its tasks within a given timeframe.

velocity of sprint

Sprint velocity is a metric in Agile development that measures the amount of work a team can complete in a single sprint. It helps with planning and forecasting future sprints.

differentiate bw incremental and iterative delivery

Characteristic	Incremental Delivery	Iterative Delivery
Definition	Breaking down the project into smaller, functional parts.	Repetition of development cycles, refining the software.
Process	Sequential delivery of increments with added features.	Repetitive cycles of development, testing, and refinement.
Delivery	Each increment is a deliverable portion of the system.	Not every iteration necessarily results in a deliverable.
Feedback	Feedback is gathered after each increment.	Continuous feedback is sought and used for refining.
Goal	To deliver a working part of the system in each increment.	To continuously improve and refine the entire software product.

Product backlog and sprint backlog

Aspect	Product Backlog	Sprint Backlog
Scope	Encompasses all features, enhancements, and fixes that might be needed for the product in the future.	Contains the subset of items from the product backlog that the team commits to completing during a specific sprint.
Time Horizon	Long-term view, covers the entire project lifecycle.	Short-term view, typically covers a single sprint duration (usually 1-4 weeks).
Dynamic Nature	Evolves and changes as the product and project progress.	Fixed during the sprint planning and shouldn't change within the sprint. Changes can be made for the next sprint during sprint review and planning.
Ownership	Owned by the Product Owner, who prioritizes items based on business value and requirements.	Owned by the Development Team, which selects items from the product backlog during sprint planning based on capacity and priority.
Granularity	Items can be at a higher level (Epics, Features) and do not need to be fully detailed initially.	Items are broken down into smaller, more detailed tasks during sprint planning.
Visibility	Visible to the entire team, stakeholders, and often the entire organization.	Primarily visible to the development team, but stakeholders may have limited access.
Review and Adaptation	Regularly reviewed and adjusted based on feedback, changing priorities, and the evolving understanding of the product.	Reviewed at the end of each sprint during the sprint review, and adjustments are made for the next sprint during sprint planning.
Backlog Grooming	Ongoing process of refining and prioritizing items as the understanding of the product evolves.	Part of sprint planning, where the team reviews and refines selected items for the upcoming sprint.
Examples	"Improve user authentication," "Enhance reporting capabilities."	"Implement user login functionality," "Fix performance issue in module X."

Agile methodologies



ChatGPT

Agile methodologies are a set of principles and practices for software development that prioritize flexibility, collaboration, and customer satisfaction. Agile approaches emphasize iterative and incremental development, allowing teams to respond to changing requirements and deliver a product in a more adaptive and customer-centric way. Here are some key Agile methodologies:

1. Scrum:

- **Roles:** Product Owner, Scrum Master, Development Team.
- **Artifacts:** Product Backlog, Sprint Backlog, Increment.
- **Events:** Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective.
- **Iterations:** Sprints (typically 2-4 weeks).

2. Kanban:

- **Visualization:** Uses a Kanban board to visualize workflow.
- **Work in Progress (WIP) Limits:** Imposes limits on work items in progress.
- **Continuous Delivery:** Emphasizes a continuous flow of work.

Roles of product owner, scrum master and development team

SCRUM ROLES

- **Product Owner:** Represents the customer, prioritizes the product backlog, and ensures the team delivers maximum value.
- **Scrum Master:** Facilitates the Scrum process, helps the team remove impediments, and fosters continuous improvement.
- **Development Team:** Self-organizing, cross-functional group responsible for delivering product increments during sprints.

daily standup meetings in agile



ChatGPT

Daily standup meetings in Agile, also called Daily Scrums, happen every workday, last around 15 minutes, and involve team members answering three questions: What did I do yesterday? What am I doing today? Are there any obstacles? The goal is to synchronize work, identify issues, and maintain transparency. It's a quick, focused check-in to keep the team aligned and address challenges.

what type of project is suitable for agile ?

Agile is suitable for projects with changing or unclear requirements, a need for quick delivery, and high customer involvement. It works well in innovative and collaborative environments, especially with small to medium-sized teams. Projects benefit when there's a focus on technical excellence, flexibility in scope, and a willingness to adapt to evolving needs. The incremental nature of Agile is effective for projects where delivering value in phases is important, and it helps identify and address risks early on.

Aspect	Structured Programming	Procedural Programming
Focus	Emphasizes structured control flow and modularization.	Primarily concerned with procedures (functions or routines).
Control Flow	Relies on structured control flow constructs (sequences, selections, iterations).	Typically uses procedures or routines to define the flow of control. May use Goto statements in less structured forms.
Modularity	Encourages modular programming, breaking code into manageable and reusable modules.	Also supports modular programming but may not enforce it as strictly as structured programming.
Readability and Maintenance	Prioritizes clear, easy-to-read code for improved maintainability.	Focuses on creating procedures to encapsulate functionality, which can aid in code maintenance.
Abstraction	Uses functions or procedures as a form of abstraction.	Abstraction is achieved through the use of procedures and functions.
Goto Statement	Discourages the use of the Goto statement.	May use Goto statements for control flow, although this is discouraged in modern programming practices.
Examples	Structured Programming: C, Pascal.	Procedural Programming: Fortran, C, early versions of BASIC.

Aspect	Epic	User Story	Task
Scope	Large, overarching theme or feature	Smaller, focused on a specific functionality	Very specific, part of a user story
Size	Largest in size	Medium in size	Smallest in size
Level of Detail	High-level, often requires further breakdown	Provides details about a specific feature	Detailed, breaks down user story into tasks
Independence	Can consist of multiple user stories	Independent, typically stands alone	Part of a user story, not independent
Value to Customer	Delivers significant value to the customer	Delivers value but may require other user stories	Supports the completion of a user story
Development Time	Longer development time	Medium development time	Shorter development time
Ownership	Owned by product management or stakeholders	Owned by the development team	Owned by the individual responsible for the task
Examples	"Implement payment gateway," "Enhance security"	"User can log in," "Customer can add products"	"Create database schema," "Implement API endpoint"

AGILE ADVANTAGES AND DISADVANTAGES

- Advantages:
- 1) Frequent Delivery
 - 2) Face to face communication with client
 - 3) Changes
 - 4) Time

- Disadvantage:
- 1) Less documentation
 - 2) Maintenance Problem

Aspect	Functional Requirements	Non-Functional Requirements
Definition	Describes what the system should do and its behavior.	Describes how the system should perform its functions.
Focus	Concerned with the specific features and capabilities.	Concerned with qualities or attributes of the system.
Measurability	Generally more easily measurable and testable.	May be subjective and harder to quantify directly.
Examples	User authentication, data validation, reporting.	Performance, reliability, usability, scalability.
Change Frequency	May change more frequently as user needs evolve.	Typically more stable and changes less frequently.
Dependencies	Directly related to user interactions and expectations.	Affects the overall system behavior and user experience.
Testing Approach	Tested through functional testing methods.	Tested through non-functional testing methods.
Criticality	Failure to meet may result in a system that does not meet user needs.	Failure to meet may impact system performance, user satisfaction, or other quality aspects.
Examples (continued)	Search functionality, data input forms, calculations.	Response time, system availability, security measures.
Documentation	Typically documented in use cases and user stories.	Documented in Service Level Agreements (SLAs) and performance specifications.
User Visibility	More visible to end-users as features they can interact with.	Less visible but crucial for overall system quality.

Aspect	Traditional (Waterfall)	Agile
Project Lifecycle	Sequential, with distinct phases (requirements, design, implementation, testing, deployment).	Iterative and Incremental, with continuous feedback and adaptation.
Flexibility	Less flexible, changes are difficult once the project has progressed beyond the initial phases.	Embraces changes, welcomes requirements modifications even late in the development process.
Planning	Extensive upfront planning before the start of development. Detailed project documentation.	Adaptive planning, planning is done incrementally throughout the project. Emphasis on collaboration over documentation.
Delivery Time	Longer delivery cycles due to sequential phases.	Shorter delivery cycles with potentially shippable increments at the end of each iteration.
Customer Involvement	Limited customer involvement until the end of the project.	Continuous customer involvement and feedback throughout the development process.
Risk Management	Risks are addressed at the beginning of the project.	Risks are continually assessed and addressed throughout the project.
Testing	Testing is a separate phase that occurs after development.	Continuous testing is integrated throughout the development process.
Team Structure	Hierarchical team structure with well-defined roles.	Cross-functional, self-organizing teams with flexible roles.
Documentation	Extensive documentation is a priority.	Emphasis on working software over comprehensive documentation, though documentation is still valued.
Adaptability	Less adaptable to changes in requirements.	Highly adaptable to changing requirements and customer needs.
Feedback	Limited customer feedback until the end of the project.	Continuous feedback from customers and stakeholders is encouraged and incorporated.
Success Criteria	Success is often measured by adherence to the plan and meeting predefined requirements.	Success is measured by the ability to deliver value to the customer and respond to changing needs.

Sprint

In Scrum, a "sprint" is a time-boxed iteration typically lasting one to four weeks, with two weeks being common. During a sprint, a cross-functional team works to deliver a potentially shippable product increment. The work is determined at the start in Sprint Planning, forming the "Sprint Backlog." The team focuses solely on this work during the sprint, fostering stability. A Sprint Review showcases completed work, and a Sprint Retrospective allows the team to reflect. Sprints provide an iterative approach in Scrum, enabling adaptability to changing requirements and continuous delivery of incremental value.

Software Design Approaches

Function Oriented Design

- System is designed from a functional viewpoint.
- Top-down decomposition
- Divide & Conquer approach
- DFD is used

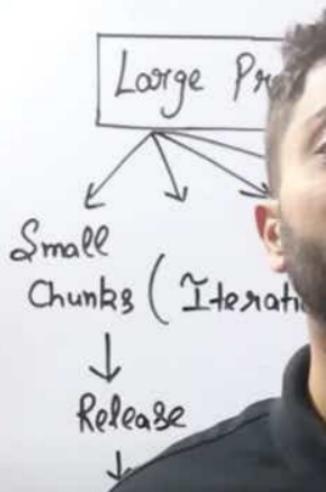
Object Oriented Design

- System is viewed as a collection of objects (i.e., entities)
- Bottom-up approach
- UML is used





* Agile (Move Quickly)



- Advantages:
- 1) Frequent Delivery
 - 2) Face to face communication with client
 - 3) Changes
 - 4) Time

- Disadvantage:
- 1) Less documentation
 - 2) Maintenance Problem

Aspect	Scrum	Kanban
Methodology	Agile framework with defined roles, events, and artifacts	Agile methodology emphasizing flexibility and continuous improvement
Iterations	Sprints (time-boxed iterations)	No predefined iterations, work flows continuously
Roles	Scrum Master, Product Owner, Development Team	No specific roles; encourages team collaboration
Planning	Sprint planning, backlog grooming	Continuous flow, no specific planning meetings
Work Items	Backlog items broken into user stories/tasks	Visualizes work as a flow, often using cards or boards
Meetings	Daily Stand-ups, Sprint Review, Sprint Retrospective	Continuous improvement meetings; no specific required meetings
Focus	Emphasizes delivering a potentially shippable product increment	Focuses on workflow visualization and reducing lead time
Flexibility	Less flexible during sprints due to predefined goals	Highly flexible, adapts to changes instantly
Visualization	Sprint board, burndown charts	Kanban board, cumulative flow diagram

Advantages

- Freedom & Adaption
- High-quality, low-risk product.
- Reduce the development time up to 40%
- Scrum customer satisfaction is very important.
- Reviewing the current sprint before moving to new one.

Disadvantages:

- More efficient for small team size.
- No changes in the sprint.

A photograph of a person's hand holding a black marker, writing on a whiteboard. The hand is positioned on the left side of the frame, with the marker pointing towards the center where the word 'Advantages' would be written.

SUBSCRIBE

