AGILE S/W ENG.

project using agile →
- small to medium size project with changing requirements
- incremental delivery of project

UNIT-1 eg:- IBM & Ltd Recruitment → set up goals & chose self teams

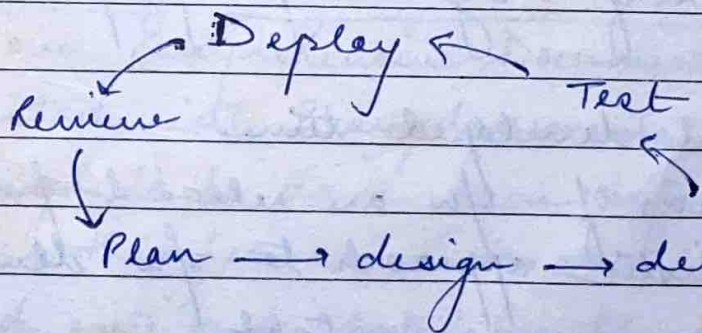
project ~~team~~ not using agile
a) large scale project with fixed scope eg:- home construction
b) Non s/w project
c) Project with stable requirement
d) Research oriented projects.

Things needed to be done in order

Genesis Of Agile → "Agile" → move quickly ...
↳ In agile large projects are broken into small chunks called "Iterations". Then work on those iteration parallely. ie. Develop, Test.
After testing we release it in market & then take feedback from users. Based on feedback we enhance & release it again.

#Agile Model → Agile means rapid incremental delivery of software ...
↳ It is mostly used s/w engineering model in today.
↳ Agile means "The ability to respond to changes from requirements, technology & people".
↳ It is an incremental & iterative process of s/w Dev.

#Working of Agile Model →
↳ divided requirements into multiple iterations
↳ Each iteration lasts for 2-3 weeks.
↳ Direct collaboration with customers.
↳ Rapid Project Development:



Note: When to use agile model ?

1) When project size is large.
2) When frequent changes are required
3) When a highly qualified & experienced team is available.
4) When a customer is ready to have a meeting with a s/w team all the time.
5) Projects with flexible timelines & Budget...

#Agile Principles →

1) Highest priority is to satisfy customers to early & continuous delivery of software.
2) Being flexible about changing requirements at any point.
3) Working on frequent & short deliveries
4) Transparency b/w business people & Developers.
5) F2F communication as the most effective way of communication
6) Continuous attention towards effective designing & technical Excellence.
7) Code Simplicity is the art of maximum result & less hardwork by removing unnecessary tasks.
8) Best Architecture, requirements and designs emerge from self-organising & experience teams.

#Advantages →

1) Reduced total developed time
2) Updated versions of S/w are released frequently.
3) provides a realistic approach to s/w development.
4) Anytime changes are acceptable.
5) Efficient Design & supports customer involvement
6) Little Planning Required
7) Strong communication of S/w teams with customer.

#Disadvantages → (1) Due to lack of proper documentation, once the project completes & developers are alloted to another project, maintainance of finished project can be a difficulty.

(2) Depends heavily on customer interaction so if customer is not clear, team can be driven in wrong direction.

#Agile Manifesto →

It has 4 values & 12 principles to make s/w dev. process more agile.

4 values are ⇒

i) Individuals & Interactions Over process & Tools: During the time of waterfall model, a lot of attention was given to processes being followed. So, no matter how detailed & well defined a process is, there will always come some barriers which couldn't have been anticipated by any process or a tool. Such barriers are overcomed when Individuals sit together, discuss & find solution of the problem.

ii) Working S/W over comprehensive Documents → Waterfall model used to start with heavy documentation, a lot of documentation used to be done even before starting to build the s/w. So this has some drawbacks:- what you have documented will be outdated by the time you are done, so Agile favours instead of wasting time in creating

long documents & then taking feedback, we should take feedback on the basis of working software.

iii) Customer Collaboration over Contract Negotiation: In traditional method of sfw dev., customers were only involved at the beginning for Contract Negotiation, one time if changes to be made & finally at the end of the project. The development was more focussed on what is written in contract instead of being focussed on what customer wants, this neglected the value of customer's feedback which resulted in a project which couldn't satisfy customer. ∴ why agile focuses on taking consent / feedbacks from customer throughout the development process.

iv) Respond to change over following plan: Traditional sfw development methods were not in favour of change, this approach was to create detailed plans in the beginning & then follow it throughout the process which lead to an improper product & it also was missing the chance of change in sfw which can make the product better. ∴ why agile favours responding to change over following plans.

* Overview of Agile Methodologies → A 'methodology' is a body of methods, procedures, & rules for a particular discipline so agile methodology is a process or set of practices that define how software development is done.

Suppose you are developing a fw that is purely customer centric where you are giving regular updates to them reciving feedback & according to feedback you are making changes, such a process is known as 'Agile Methodology'.

Terminologies → (1) SPRINT → It is a cycle or a time interval ie in a particular time what amt. of work needs to be done, generally (2 weeks = 1 sprint)

(2) SPRINT PLANNING MEETING → ## sprint is of 2-3 weeks so work required to do in that time to discuss, a meeting is scheduled which is called sprint planning meeting.

Types of Agile Methodologies →

(1) SCRUM → We know that waterfall model ie

Plan ← Build ← Test ← Review ← Deploy

consists of a long time which is involved in planning ie week & month for all stages which may result in outdated product.

↳ Scrum solves the problem by breaking the project into smaller parts

1st → you start with just enough plans to get the project started.

2nd → you build the project with min. features

3rd → Test & Review for the same.

One this cycle is complete, we have a potentially shippable product & this process takes around 2 to 4 weeks. & is performed time & again, this way we end up with several different versions or incremental release.

↳ SCRUM is a framework within which people can address complex problem ~~which~~ while delivering products of highest possible value. (aimed at solving complex problems)

↳ SCRUM itself is a simple framework for effective team collaboration

↳ SCRUM promotes developing products through processes, techniques & practices with iterations & increments to deliver a product of max. value.

So, these steps of
1) Planning
2) Build
3) Test
4) Review } - Iteration

& we have.

∴ (1 → n) no. of iterations & these are called 'sprints' & at end of each sprint you have to launch a potentially deliverable software.

Note (Each sprint takes 2-4 weeks & repeated till whole product is developed)

↳ SCRUM follows "agile" approach to tackle complex problems & agile is NOT a methodology, framework or a process. Agile promotes Self learning & Self organisation

* Agile means not planning the whole entire process to create a potentially shippable product but to only plan enough to start out with & then react to responses you get with each sprint).

4th) Stakeholders → Anyt who has input to project, Not official scrum role but essential for scrum role & stakeholder to work closely throughout

5th) Agile Mentor → an experienced person on agile techniques & scrum framework helps scrum team with outsider's point of view.

0) Roles in Scrum →

↳ 1st we have 'Product Owner' with bright ideas for the product. & speaks business needs. It is a person not a committee

↳ 2nd we have 'Scrum Master' who is implementing agile & preparing team to follow agile approach & making them more efficient. / ensure scrum is played properly.

↳ 3rd we have 'Team' which consists of developers, testers etc, performs day to day work. Each member is multi skilled

Note: Agile Approach → 1) Creating a plan for Sprint i.e Sprint Backlog
2) Day to day work

problem: Suppose there are n no. of people in room & they have to queue up acc. to their height, taking min. amt of time.

Sol^{n} Approach I: Superior Approach → We have a supervisor, who arranges people one by one taking up time.

Approach II: Agile Approach → here we have a no. of ppl & a 'Scrum Master', the SM allows team to organise themselves. & in end he brings about whatever changes he deems necessary. In agile approach, requirements & solution evolve through collaborative effort of self organizing & cross functional teams consuming less time.

⇒ "Agile" is set of practice that promotes continuous iteration of development & Testing throughout the SDLC & "Scrum Master" is one who facilitates agile principles, he/she manages the process for how information is exchanged.

* Scrum Phases →

(i) Initiate → 1st we have to create project vision ie define what customer needs, his objective, his aim to deliver project.

↳ 2nd is to identify Scrum Master & Stakeholder.

↳ 3rd is to form Scrum team which consist of developers.

↳ 4th is to develop epic ie mean story that will be delivered

↳ 5th is to create Prioritized Product Backlog ie complete requirement for product priority wise, priorities are set by product owner.

↳ 6th is to conduct Release planning ie After every sprint we shall release the deliverable to owner.

(ii) Plan & Estimate → 1st is to create user stories ie functional requirements

↳ 2nd is to approve, estimate & commit user stories.

↳ 3rd is to create task ie you have to define task how you will develop, test etc.

↳ 4th is to estimate task ie how much time, cost required.

↳ 5th is to create Sprint Backlog, it is a doc. which is subset of product backlog (req. which will be delivered in sprint. eg:- If 40 task, then 10 to be delivered in sprint)

(iii) Implement → 1st is to create deliverables ie..

↳ 2nd is to conduct daily meetings, a min. meeting in which our team discusses what they have done yesterday, what to do today & tommorow along with risk fixed.

↳ 3rd is to groom prioritized product backlog.

iv) Review & Retrospect → 1st is to convince Scrum of a Scrum meeting. ie Suppose if it is a large product, so we make 5-5 team members for each team and form a Scrum master of each team, he will discuss issues & problemes with SM of another team so ie Scrum of a Scrum
↳ 2nd is to demonstrate & validate Sprint ie whatever we have developed, we showcase it to customer.
↳ 3rd is to retrospect Sprint ie a lesson learned in meeting in which we discuss the problems & issue we have faced, what things go wrong & what we have to avoid.

v) Release → ↳ 1st is to ship deliverables to customer
↳ 2nd is to retrospect project ie. post mortem of whole project ie what went wrong & right & what needs to be improved.

★ Scrum artifacts → helps scrum team & stakeholder with info about product under development & the activities undertaken in developing product & contains all crucial info required during product development stages covering to-do & done.

i) Product Backlog → ordered list of work to build & improve
↳ Single source of work ie contains user stories, task etc.
↳ 1 type of to-do list for scrum team (Product owner owns it)
↳ product team is responsible for product backlog:
ii) Sprint backlog → plan for sprint created & managed by developers
↳ list of task in a given sprint. PO & Dev Team select req. for sprint in sprint planning
↳ dev team breaks deep into tasks.

Only be changed by dev. team.

iii) Increment → done work within Sprint, entire Scrum team is responsible for creating a valuable, useful increment for each sprint.

UNIT-2 (More of the Scrum Part is covered in UNIT-1)

5 events of Scrum →

1) Sprint → They are short cycles (1-2 weeks) & in some case less than a day.
→ In each sprint, the scrum teams creates potentially shippable product.
→ Consistent sprint length reduces variance.
→ It gives scrum team an opportunity to make adjustments for continuous improvement rather that at end of project

2) Sprint Planning → Takes place at start of each sprint, Scrum team decides which goal, task will be part of sprint backlog.

3) Daily Scrum → Takes place daily (<15 min). During DS, dev team makes 3 statements:
- What team member completed yesterday
- What the team will work on today
- A list of items impeding team members.

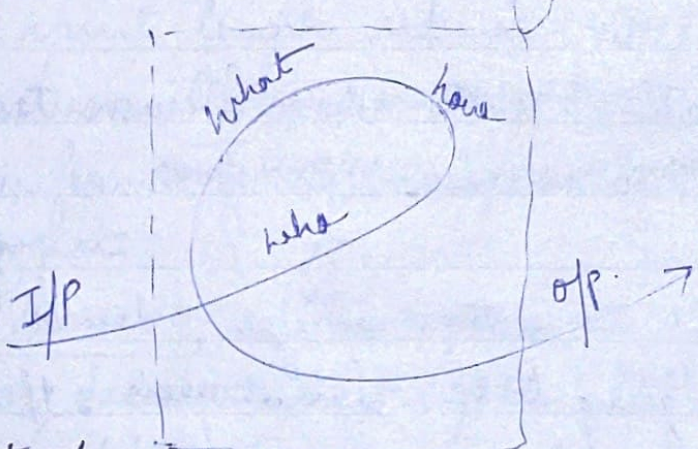
Agile Project Velocity, burn down chart, Sprint planning, Daily Scrum, Case Study, Tools.

4) Sprint Review → Takes place at end of each sprint dev team demonstrated to stake holders, the accepted parts of product, the team has completed during sprint.
↳ The key is to collect feedback from stakeholders which informs product owner how to update product backlog & consider next sprint goal. → plan ways to increase quality & effectiveness

5) Sprint retrospective → Takes place at end of each sprint. It is an internal team meeting in which Scrum team discuss what went well during sprint, what didn't work well & how they can make improvements. This meeting is action oriented & ends with improvement plans for next sprint.

Sprint planning → purpose is to define what can be delivered in sprint & how that work will be achieved.
↳ Done in collaboration with whole Scrum Team.
↳ In scrum, sprint is an interval of time in which a work is done.
↳ However 1st you have to setup ① sprint, ② decide ③ sprint goal. If done correctly, it also create an environment where team is motivated & can be successful.
↳ Sprint planning is an event in scrum that kicks off sprint

- What → product owner describes goal of sprint. Scrum team decides what can be done in coming sprint & what they will do in sprint to make that happen.
- How → dev. team plans ~~as~~ work to deliver sprint goal. The resulting sprint plan is a negotiation b/w dev team & product owner based on value & effort.
- Who → Can do sprint planning w/o PO & dev team.
  PO defines goals based on values they seek. dev team needs to understand how they can deliver goal. If either is missing, it makes planning impossible
- Inputs → Starting point of sprint plan is "Product Backlog" as it provides a list of 'stuff' that could be part of current sprint.
- Outputs → The outcome of sprint planning is that team can describe goal & how it will work towards the goal. This is mentioned in "Sprint Backlog"



- Time limit for Sprint Planning →
- should be not more than 2 hrs for each week of sprint.
- This is called "time boxing" / setting max time for team to complete task. Timebox is max. time allowed; there is no min time.
- Focus on outcome (goal) not the work
- Estimates are required but don't pretend you know more than you actually do...

★ "Daily Scrum" → Purpose is to inspect progress toward Sprint Goal & adapt Sprint Backlog.
↳ It is a 15 min. event for Developers of Scrum Team
↳ To reduce complexity, it is held at same time & place every working day of sprint.
↳ If Product Owner / Scrum Master are actively participating ~~on~~ on items on Sprint Backlog, they act as developers.
↳ Developer select structure / Technique they want as long as Daily Scrum focuses on progress towards Sprint Goal & produces an action plan for next day.
This creates focus & improves self management.
↳ Daily Scrum improves comm", identify impediments, promote quick decision making.
↳ Developers use Daily Scrum to inspect how progress is tending towards completion of work in Sprint Backlog.
↳ Daily scrum optimizes the probability that Dev. will meet Sprint Goal.
↳ Team members often meet immediately after Daily Scrum for detailed discussion, to adapt/plan rest of work.
↳ Daily Scrum improves Dev. Team's level of knowledge.

ROLE OF SCRUM MASTER :→

o) ensures that meeting happens, but Dev. are responsible for conducting Daily Scrum.
o) teaches Dev team to keep meeting within 15 min time box.
o) ensures other roles/members do not disturb meeting.

iii) Increment → done work within sprint, entire Scrum team is responsible for creating a valuable, useful increment for each sprint.

(2) Extreme Programming (XP) → It is light weight, efficient, low risk, flexible, predictable & scientific way to develop a s/w.

Application → (1) (Continuous modification in requirement) (2) (Small projects) (3) (New Technology or Research projects)

↳ small to medium size teams that work under vague & rapid changing requirements.

↳ 5 values are: Communication, Courage, Feedback, Respect & Simplicity

↳ follows object oriented approach

↳ frequent iteration

↳ Principle → Rapid feedback (1), Simplicity (2) (follow YAGNI You Ain't Gonna Need it) & DRY (Don't Repeat Yourself)

(3) Incremental change → Small change better than big

(4) Embracing change → Should change acc. to client needs always.

(5) Quality Work

Phases → i) Planning → user stories ie requirements of product owner ie product owner gives their requirement in "Index Card" preference wise & when that card reaches developers they make 'Iteration plan' ie what user story to achieve in a specific iteration.

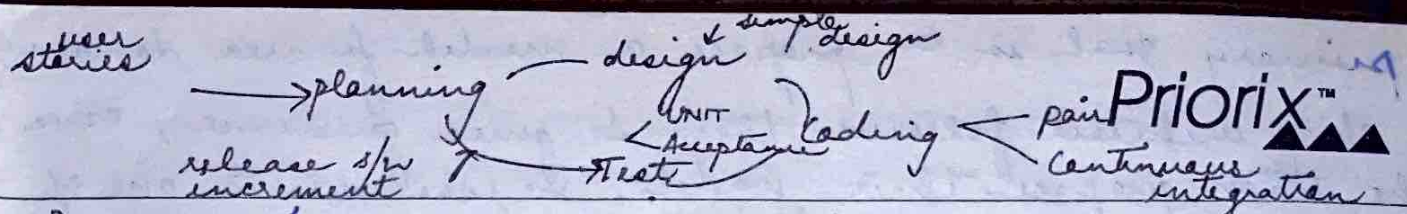
↳ It also has acceptance test criteria to judge whether s/w is according to requirements of product owner.

ii) design → 'Simple design' ie to keep design simple & easy to understand.

↳ "Spike Sol" is a very simple program to explore potential sol" difficult design should be modelled using prototype.

iii) coding → we follow 'pair programming' ie instead of ~~coding~~ single person, duo will be there to help in giving better feedback.

↳ 'Continuous Integration' after testing.



iv) Test → Acceptance testing w.r.t customer requirement

#Advantages → 1) fewer documentation required
2) Collaboration with Customers
3) Flexibility to developers
4) Easy to Manage.
5) Simplicity

v) Feedback

#Disadvantage →

1) depends heavily on customer interaction.
2) transfer to technology to new team members may be quite challenging due lack of documentation.

(3) Feature driven development (FDD) → practical agile approach suitable for long term complex projects.

↳ FDD is an agile framework that organizes s/w dev. around making progress on features.

→ FDD is customer centric, incremental & iterative to deliver tangible s/w results efficiently.

↳ FDD employs status reporting at all level, helps to track result & progress.

↳ allows team to update project regularly & identify errors quickly

↳ FDD methodology primary goal is to develop real, working s/w & meet deadlines systematically.

FDD methodology contains 5 steps:

(1) Developing an overall model → In first stage, the dev. team cooperates & builds an object model of domain problem.

↳ primary goal is to propose a model for area domain.
↳ chief architect follows them & gives guidance, once the team propose their model, he/she selects one of these models or a merge of models & it becomes the model created for that domain area. As a result team gets a clear image of entire project.

(2) Building a feature list → After establishment of model, team select the feature that the client prefers, these features are used as building barriers of project.

(3) planning by feature → To manage features & the way dev. team tends to implement them.
↳ its essential to consider the team workloads, risks & other aspects to prevent any issue.

(4) Designing by feature → chief programmer uses knowledge from first modelling phase to select all team's features & identify classes of domain.
↳ Immediately team starts working on the project, domain expert analyzes & design a sol^n to each feature.

(5) Building By feature → The final step is to put all necessary items into action to support design. In other words, once team develops, test & inspects the code, they start development of s/w.

FDD v/s SCRUM →
↳ FDD values documentation more than other methods.
↳ In FDD teams don't meet frequently as they rely on documentation to communicate imp. info. whereas in Scrum, team meet on daily basis.
↳ FDD takes shorter sprint length than ~~FDD~~ Scrum.
↳ FDD is class ownership where Scrum is shared code ownership

↳ FDD specifies engineering process practices ie design/code, inspection & tests where Scrum doesn't specify any engineering practices.
↳ FDD is domain driven, while Scrum focuses on its not.
↳ In FDD, teams have recognised roles (project manager, chief architect, dev manager, chief prog, domain expert) whereas Scrum is a self organising team.

* FDD advantages →
↳ It is simple 5 step process that allows rapid development, enabling larger team to move products forward with continuous success.
↳ provides team with opportunity to connect more efficiently.

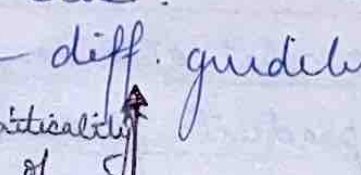
* FDD disadvantages →
↳ not efficient for small projects.
↳ doesn't work with project with only 1 developer.
↳ high dependency on chief programmer.
↳ No written doc for owner.

(4) Crystal Methodology →
↳ Crystal is a family (5) of methodologies
↳ Which project will be suitable for we project depends on 3 dimensions :- . Team Size, Criticality, What priority of project
↳ We know Agile project management approaches guides us only to organize project acc. to priority & availability of project partners in such (ie availability of project partner) we use "Crystal"...

↳ Crystal is a light weight & adaptable approach to S/w dev. based on our need, priority & availability of resources & manpower the there are diff forms of crystal project management:

1) Crystal clear. (team size: 1-6) for short term proj: feedback taken from real users. involves automated testing to solve bugs
2) " Yellow (7-20) team is split acc to skills. project lasts for 1-2 years & release is required every 3-4 months.
3) " Orange (21-40) team divided acc. to req.
4) " Red (41-80) involves large sized projects where methods are diff. & as per req. of s/w.
5) " Maroon (81-200)
6) Blue.

note: there are diff. guidelines for each one of these crystal.



↳ Criticalities are based on what might be lost because of a failure in produced system.

	L6	L20	L40	L80	L200
Life					
essential money	E_6	E_{20}	E_{40}	E_{80}	E_{200}
Directionary money	D_6	D_{20}	D_{40}	D_{80}	D_{200}
Comfort	C_6	C_{20}	C_{40}	C_{80}	C_{200}

1-6 clear | 7-20 yellow | 21-40 orange | 41-80 red | 81-200 maroon

Note: L(80) → means team size (41-80). So, our project is of moderate to large & because of L (life lost) our project is in critical level, if we fail at this level then life of project maybe lost.

* Frequent delivery → regular delivery products, least code to users.
* Osmotic comm → info to flow b/w teams in osmosis.
* Technical tools → contains specific tools to be used.
* Regular feedback.
* advantages → 1) 'value people' ie consider ppl at most imp,
2) 'adaptive model' ie n/o a set of prescribed tool & techniques 5) most effective 6) is expandable
3) 'light weight' ie n/o too much Documentation, management or reporting. 4) for short term projects
* disadvantages → 1) scalability is limited
2) Not suitable for developing high critical systems.
3) Lack of an unambiguous process.
4) over dependence on inter-human communication.
2) Crystal focuses on individuals & interaction. The methods are colour coded - to signify risk to human life.
5) Dynamic Software Development Method (DSDM) →
→ It is an iterative, incremental approach that was first coined in 1994, to impose "rapid application development".
↳ 3 key things: fixed cost, fixed quality, fixed time.
→ it also uses MoSCoW prioritization ie.
Must Should Could Won't (This Time)
↳ in respect to agile principles.
- small team
- Iterative
- Transparency
- Continuous learning
- Communication
- Team empowerment

o phases in DSDM →
(i) pre-project → Getting an understanding whether there's value within this project.
(ii) pre-project feasibility → more understanding, value analysis ensuring that project will continue.

(3) Business Study → going through requirements, understanding what needs to be built

4) Iterative development → Continuous iterative development

5) deployment → deploy the project

6) post project → finding area of improvement for future project

Test Driven Development → ↳ iterative dev. process.

↳ Every iteration starts with set of tests.

↳ Test cases are created before code is written.

↳ TDD instructs developers to write new code only if automated test has failed.

* advantages → 1) Avoids duplication of code

2) Refactoring improves code.

3) Clean code

4) Simple code

5) Bug free code

6) Small regression suite

X P uses it

* disadvantages → (1) Requirement change is paid

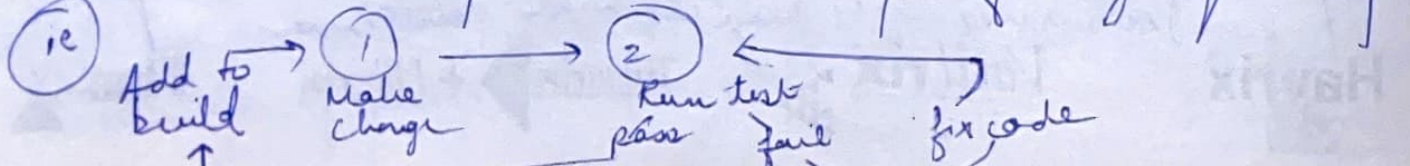
(2) Early test cases are heavy in maintainance.

9) Continuous Integration → CI is intended to minimize time & effort needed to integrate new code from multiple developers.

↳ Instead of merging at end, team continuously merges & test new code.

→ This process is automated to help developer to update code & provide version control for final product.

→ CI reduces risk & allows catching of bugs quickly



↳ Developer keep the system fully integrated.
↳ XP team make iterative dev to another level ↑ bcoz they commit code multiple times
↳ Programmer discuss which part of code can be reused shared.
↳ The policy of shared code helps eliminate integration problem
↳ Automated Testing allows developers to detect & fix errors before deployment

6) Refactoring
↳ It means simplifying, making it efficient w/o affecting external behaviour ie internally it gets better & good
↳ all the test cases passing before refactoring must pass even after refactoring.
→ eg:- Basic Refactoring transformation on OOP code:
- add a class / method, attribute
- Rename it
- Move an attribute up or down the hierarchy
- Extract chunks of code into seperate method

↳ The goal is to continuously improve code.
↳ It is about removing redundancy, eliminating unnecessary things & increasing code coherency.
↳ Keep your code clean & simple so you can easily understand & modify it would be advice of XP TM.

6) Pair programming → is an Agile s/w dev. technique in which 2 programmer work together on a same code.
→ While 1st programmer focuses on working, other one reviews, suggest improvement & fixes code. Such teamwork results in

high quality s/w. but takes 15% more time.
→ So pair programming is suggested for long term projects

5) Simple Design → We want to avoid creating a complicated design at begining of project. Instead design & work on the features that are implementing to meet requirement & then you search for enough knowledge & refactor incrementally to implement simple design.

XP	Scrum
1) Shorter iterations	1) Longer sprint
2) Flexible with changes	2) No change within sprint.
3) Focus on technical practices	3) Focuses on managerial aspect.
4) Customer determines order of feature development	4) Self organised teams decide what features to work on first

* Agile Testing → It is a new age approach which focuses on testing smarter rather than more efforts, delivering high quality product.
↳ It is a testing practice that follows principles of agile s/w dev. It is an iterative methodology.

1) principles of Agile Testing →
1) Testing is Continuous ✓
2) Continuous feedback
3) Test performed by whole team
4) Simplified Code → fixed in same iteration
5) less documentation → use reusable checklist
6) Test Driven →

2) advantages → 1) saves time
2) saves money
3) flexible of changes
4) adaptable of changes
5) less Documentation
6) Regular feedback from end user.

Agile Testing Methods →

1) Test Driven Development → It begins with test itself. TDD varies on repetition of dev. cycle.
1st step is to create unit test case. Next, we will be designing code that fits test caste in order to execute test cases. Whole code is designed until unit test passes. Generally TDD is executed by using automated testing tools & implement on units of components of code.

2) Behaviour Driven Dev. → It enhances comm" b/w projt stakeholders & understand all components before dev.
→ code is developed as per test case.
→ primary purpose is to identify the needs of outputs.
→ Steps:
- Describe behaviours
- Generate test case
- Write code as per test case
- Continue process until code passes test case.

3) Exploratory Testing → Testers have freedom to explore code & create effective tcs.
→ helps discover unknown risk
→ tester creates various test cases, executes it & record it to learn & understand.
→ Steps →
- Exploring s/w in all possible way.
- Understanding flow of application
- Preparing a test document.
- Testing the application.

✱ Agile Tools →

i) requirement management tools → jira, MS Excel.
ii) Source Code Management → Gitlab, Github, VSS.
iii) Code Development (IDE) → Eclipse, Visual Studio, Netbeans.
iv) Unit Testing → TestNg, Selenium, Junit
v) Continuous Integration → Jenkins, Docker, Maven, Bamboo
vi) Code Deployment → Jenkin, Bamboo, AWS Code Deploy.

Tools For Agile Project Management →

1. ProofHub → Smart & feature rich, used by leading agile proj mang. s/w organisations. Teams can effortlessly share ideas, compile doc, start discussion & move forward.
   ↳ It comes integrated with all necessary tools & features that needs to make our project successful using Agile methodology.
   ↳ For organising tasks you can use ProofHub's Kanban Board & Gantt Charts.
   - Kanban Board → able to define workflow stages of project & see tasks moving across diff. stages in form of a card.
   - Gantt chart → ables to visualize timeline of a project & ensure timely completion of project.
   ↳ Other work are as follows:
   → Create Task
   → Divide Task into subtasks
   → Set Start & End dates for each task
   → Select one or more assignees for a task
   → Attach files & add comments
   → Add labels to prioritize task &
   ↳ Proofhub consists of built in chat interface to connect with other members of your team making it perfect.
   ↳ ProofHub will make Agile Methodology more meaningful & will provide enhanced o/ps.

(2) Wrike → ideal for centralizing & connecting multiple projects & increasing team's efficiency.

↳ Although Wrike is upto date with their releases, & their mobile versions lag in process. This might frustate user if he/she wants to see the changes in mobile which is happening in Laptop.

(3) Smartsheet → defines how team collaborate on projects such as managing o/p, tracking & planning events.

↳ It considers the various solutions for business

↳ has free & open collaboration

↳ Allows to create team w/o limitations.

~~(4) Version One →~~

(4) Version One → It is all in 1 Agile Tool that supports agile s/w development methodologies.

→ Features are agile portfolio management, quality management, business intelligence, collaboration & development.

→ We can establish link b/w global product & portfolio team with its collaboration feature.

(5) Agilean → is a SaaS enterprise workflow automation & project management s/w for IT enterprises.

↳ features including → project plan, execution, SCRUM & Kanban board

↳ easy to use & customizable.

↳ removes bottlenecks from process.