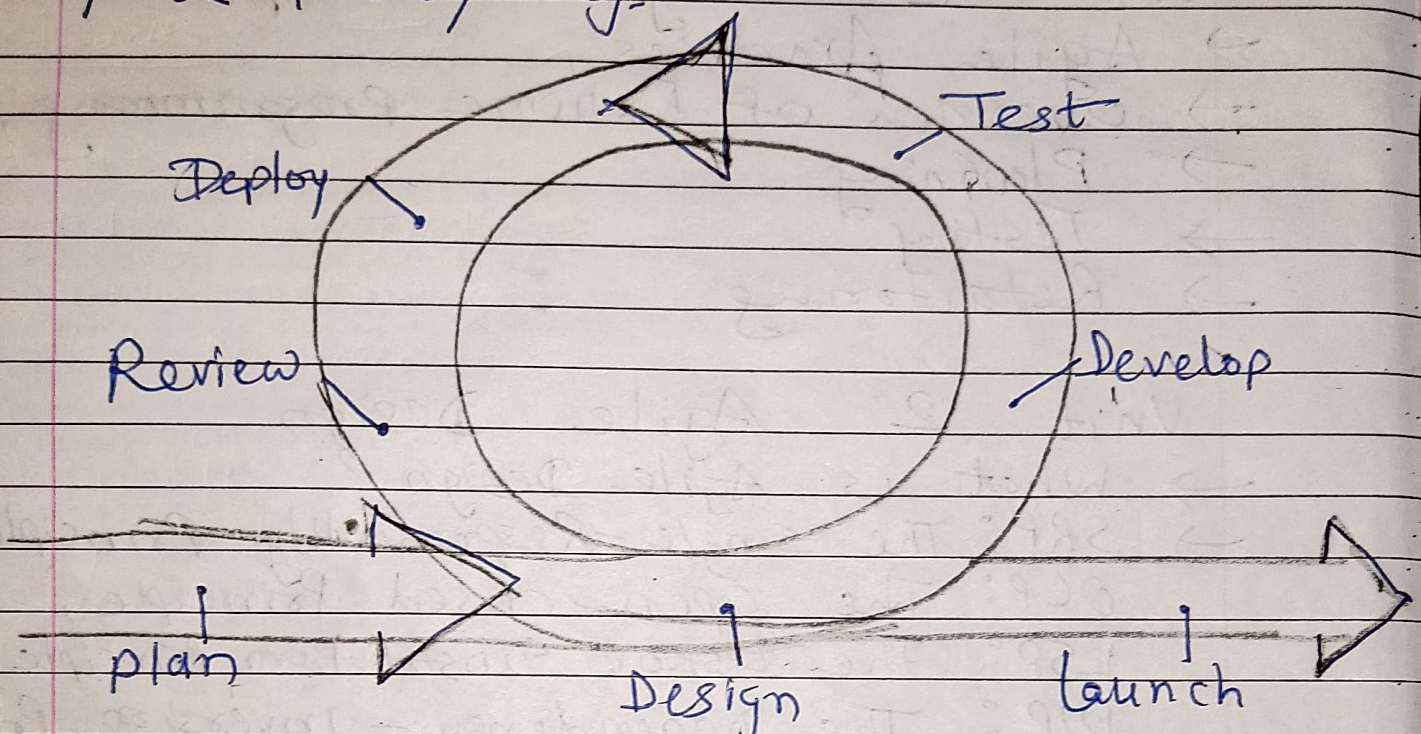
[Responsibility equal to all members, time-period wise modules are made & deploy to to clients/shakeholder] Client gives feedback for module, then it is incorporated in next timeframe]

Agile ⇒ ready to move in quick & easy way.



Why? & Where

Agile can ultimately be used on almost any large scale project in any industry. Any of these project teams can benefit from using Agile : Teams handling fast-changing deliverables, such as technology products. Teams working on projects that evolve or do not have clear scope & requirements at the beginning.

App's — Info. system, Payroll, Predictive modeling, Data analysis, QA & QA Automation

R. D. D. T. Deploy

What is Agile ?

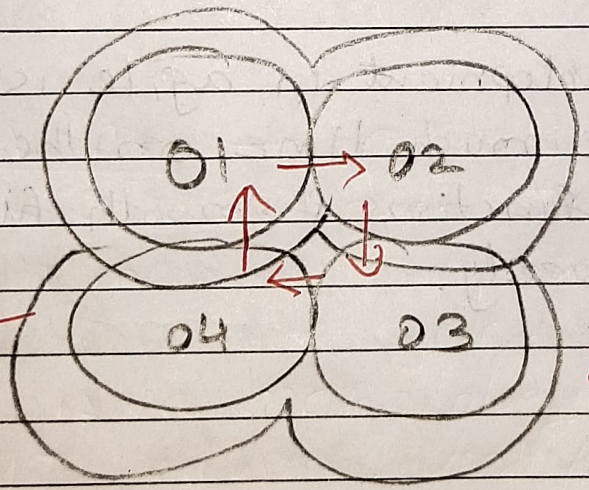
Agile is an Iterative approach to project management & s/w development that helps teams deliver value to their customers faster and with fewer headaches.

Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments.

Requirements, plans, and results are evaluated continuously so teams have a natural mechanism for responding to change quickly.

Agile Design

DADI



Agile is a type of methodology in which the work is divided into different subparts to make it easy to handle & increase the product's quality.

The agile design process includes :

Feedback : In this process, one can easily interact with the customer to know his requirements and also get the feedback of the client or customer regarding the product and make necessary changes required according to the feedback; it is a very helpful function in the agile design.

Changeable : During the agile design, it is easily doable if any changes are required in the design. There is the proper function for the changes, and s/w is very helpful for the alteration in the design.

Development : The development in agile is very fast; teams need not waste much time on the allied processes; its easier functions & smooth functioning make the process speedy.

Process of Agile Design

1. Design : The design of agile will be made by using any of the methodologies used above, but the best design is that which is customer or client-centric and gives fruitful results.

Also, the team that takes charge of making a better agile design pays proper vigil (awareness) on the project so that no time & resources will be wasted.

The design is a ~~best~~ hectic task at the inception of the project, which needs proper attention.

2. Analyze : When the design is completed and work is on track, a proper analysis is required from time to time so that the scope of the fault should be eliminated & the quality of the product is maintained. The analyses of the design are also helpful for the team to complete work on time.

3. Develop : Here, develop means to develop the required project management S/W, which is the prime moving force of the project after its completion. Management is very necessary after the proj. execution, so agile management is much required

4. Implement : The last stage of the process is to implement the agile project & its execution. The feedback of the design is most important so that the team can make changes if required. Also, the satisfaction of the client is much required things after the implementation.

Agile Practices

Agile is a proj. management methodology that consists of small development cycles, called "Sprints" to keep the focus on bringing continuous improvement in a product or service.

Sprint is a pre-determined timeframe for the team to complete a particular task.

Each sprint usually ends with a review where the team reviews their performance & discusses ways to improve the work.

Agile methodology is being widely adopted by numerous industries where the focus is to build products or services through a continuous cycle of small changes.

"Change is a constant, there is no need to fight it, and it's better to embrace it"
(Accept)

As opposed to the waterfall approach that uses a step-by-step technique for product development, Agile best practices focus more on bringing flexibility with constant updates throughout the process.

In other words, "Agile Best practices' is a parent term for a broad range of frameworks, & practices that guide these frameworks.

Some of the famous proj. management frameworks for agile include Scrum, XP.

Agile Best Practices

① Iterative Development

Bigger projects are broken down into smaller chunks and continuous tests are done in repetitive cycles. Through this practice, agile teams get a perspective on new features that need to be added to the final product or service and contribute towards more flexible product development.

② Regular Meetings

Regular meetings are key to agile implementation. These meetings should be short and concise, with each member of the team explicitly stating the progress of tasks & what needs to be done. This practice is a great way to monitor the performance of the team & check if there are any obstacles in the way of product development.

③ Using Professional Tools

Using proj. management tools for the implementation of agile methodology helps the team to better structure their workflows & improve team collaboration.

For proper documentation & meetings management, professional proj. management s/w can greatly reduce the effort it takes to manage your tasks otherwise.

[One such tool that you can easily use is nTask]

A comprehensive s/w, with the most intelligent features, provides a smart over-all coverage to all your agile project management needs.

Agile Practices

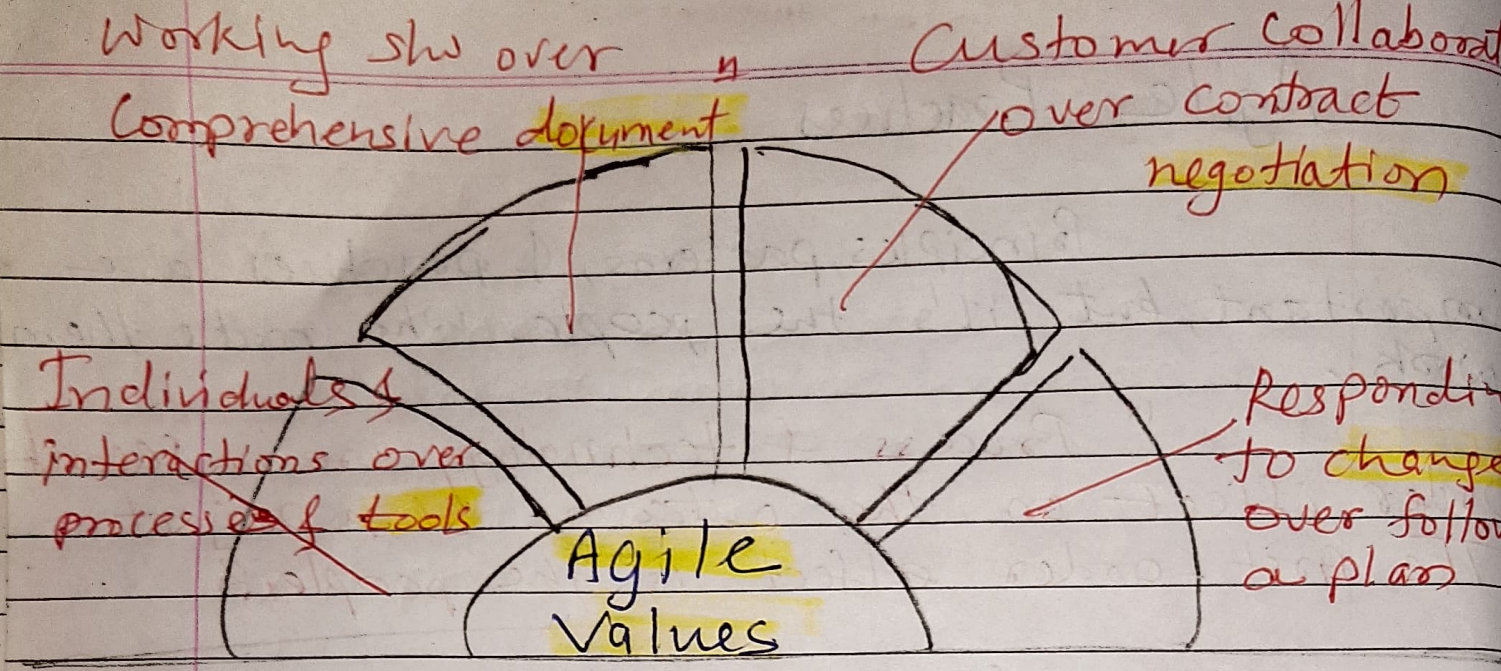
Principles, patterns, & practices are important, but it's the people who make them work.

"Process & technology are a second order effect on the outcome of a project. The first-order effect is the people."

(Policy or aim) Manifesto for Agile s/w development

Motivated by the observation that s/w teams in many corporations were stuck in difficulty of ever-increasing process, a group of industry experts calling themselves the Agile Alliance met in early 2001 to outline the values & principles that would allow s/w teams to develop quickly & respond to change.

Over the next several months, this group worked to create a statement of values.



① Individuals & interactions over processes & Tools

→ People are the most important ingredie of success. A good process will not save a project from failure if the team doesn't have strong players, but a bad process can make even the strongest of players ineffective. Even a group of stro players can fail badly if they don't work as a team.

→ the right tools can be very important to success. Compilers, interactive development environments (IDEs), source code control syste & so on, are all vital to the proper functioning o a team of developers. However, tools can be overemphasized. An overabundance of big, unwieldly tools is just as bad as a lack of tools

W S / C.D

→ Instead, work to create the team, & then let the team configure the environment on the basis of need.

② Working s/w over comprehensive Document
[produce no doc. unless its need is immediate & significant
→ s/w without documentation is a disaster
code is not the ideal medium for communicat the rationale & structure of a system.
Rather, the team needs to produce human-readable documents that describe the sys. & the rationale for design decisions.

→ However, too much documentation is worse than too little.

→ Huge s/w documents take a great deal of time to produce & even more time to keep in sync with the code.
[

→ It is always a good idea for the team to write & maintain a short rationale & structure document. But that doc. needs to be short & salient.
Short, → Mean one or two dozen pages at most
Salient → Mean that it should discuss the overall design rationale & only the highest-level structures in the sys.

C C over C N

(3) Customer Collaboration over Contract Negotiation

→ SW cannot be ordered like a commodity. You cannot write a description of the SW you want & then have someone develop it on a fixed schedule for a fixed price.

→ Successful projects involve customer feedback on a regular & frequent basis. Rather than depending on a contract, or a statement of work, the customer of the SW works closely with the development team, providing frequent feedback on its efforts.

→ A contract that specifies the req. schedule & cost of a proj. is fundamentally flawed. In most cases, the terms it specifies become meaningless long before the proj. is complete, sometimes even long before the contract is signed!
The best contracts are those that govern the way the development team & the customer will work together.

RC / fp

(1) Responding to change over following Plan

→ The ability to respond to change often determines the success or failure of a s/w project.
When we build plans, we need to make sure that they are flexible & ready to adapt to changes in the business & technology.

→ The course of a s/w proj. cannot be planned very far into the future.

→ First, the business environment is likely to change, causing the requirements to shift. Second, once they see the sys. start to fn, customers are likely to alter the requirements.

→ Finally, even if we know what the requirements are & are sure that they won't change, we are not very good at estimating how long it will take to develop them.

Overall | 12 principles of Agile

→ Customer satisfaction — C S — W-S/W

→ welcome change — W C — C.P

→ deliver frequently — D F — S.D

→ Working together — W T — Simp

→ Motivated team — M T — S.O

→ Face-to face — F-F — R&A

→ Working S/W

→ Constant pace

→ Good design

→ Simplicity

→ Self organization

→ Reflect & Adjust

(1) Our highest priority is to satisfy the customer through early & continuous delivery of valuable s/w.

(2) Welcome changing req, even late in developments. Agile processes harness change for the customer's competitive advantage.

(3) Deliver working s/w frequently, from a couple of weeks to a couple of months with a preference to the shorter timescale.

(4) Business people & developers must work together daily throughout the project.

(5) Build projects around motivated individuals. Give them the environment & support they need, and trust them to get the job done.

(6) The most efficient & effective method of conveying info. to & within a development team is face-to-face conversation.

XP → one step ahead of best practices (innovative) that stretched the limits of methods like unit testing, pair prog., customer acceptance tests.

(7) Working shw is the primary measure of progress.

(8) Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

(9) Continuous attention to technical excellence & good design enhances agility.

(10) Simplicity — the art of maximizing the amount of work not done — is essential.

(11) The best architectures, requirements, and designs emerge from self-organizing teams.

(12) At regular intervals, the team reflects on how to become more effective, then tunes & adjusts its behavior accordingly.

So Agile's real utility is giving people a common foundation for making decisions about the best way to develop shw.

Extreme m
on the end
customer nee

famous
up of
These p
that

→ custom
person

→ Whole
& deve
so th
problem
problem

→ User
mnema
about
the cu
of a
estima

→ Shor

each
stake

Extreme methodology is used mainly for sw dev. & focuses on the end product, & ensure that it should be as far as the customer needs

Extreme Programming

Extreme programming is the most famous of the agile methods. It is made up of a set of simple, yet interdependent practices. These practices work together to form a whole that is greater than its parts.

→ Customer — The customer of an XP team is the person or group that defines & prioritizes features.

→ Whole team — We want customers, managers, & developers to work closely with one another so that they are all aware of one another's problems & are collaborating to solve those problems.

→ User stories — A user story is a mnemonic token of an ongoing conversation about a requirement. It is a planning tool that the customer uses to schedule the implementation of a requirement based upon its priority & estimated cost.

→ Short cycles — An XP proj. delivers working sw every two weeks
— The Iteration Plan
— The Release Plan
At the end of each iteration, the sys. is demonstrated to the stakeholders in order to get their feedback.

The iteration plan — It is usually two weeks in length & represents a minor delivery that may or may not be put into production.

During this time, the developers are free to cut the stories up into tasks & to develop the tasks in the order that make the most technical & business sense.

The release Plan — XP teams often create a release plan that maps out the next six or so iterations. That plan is known as a release plan.

A release plan consists of prioritized collections of user stories that have been selected by the customer according to a budget given by the developers.

Releases are not cast in stone. The business can change the release content at any time. The business can cancel stories, write new stories, or change the priority of a story.

However, the business should strive or try not to change an iteration.

Acceptance Tests :

The details about the user stories are captured in the form of acceptance tests specified by the customer.

Together, they act to verify that the sys. is behaving as the customers have specified.

Acceptance tests are written by business analysts, quality assurance specialists, & testers during the iteration.

~~It is from these tests that the prog~~

These tests become the true requirements document of the project.

Every detail about every feature is described in the acceptance tests, & those tests are the final authority as to whether those features are done & correct.

Once an acceptance test passes, it is added to the body of passing acceptance tests and is never allowed to fail again.

Pair Programming

Code is written by pairs of programmers working together at the same workstation.

One member of each pair drives the keyboard & types the code. The other member of the pair watches the code being typed, finding errors & improvements. Both are completely engaged in the act of writing s/w.

The roles change frequently.

A reasonable goal is to change pair ~~partner~~ partners at least once per day so that every programmer works in two different pairs each day.

Pair programming dramatically increases the spread of knowledge throughout the team.

These have suggested that pairing does not reduce the efficiency of the programming staff but does significantly reduce the defect rate

Test-Driven Development (TDD)

All production code is written in order to make a failing unit test pass.

The test cases & code evolve together, with the test cases leading the code by a very small fraction.

Programming a pair that makes a small change can run the tests to ensure that nothing has broken. This greatly facilitates refactoring.

The principles of object-oriented design (OOD) play a powerful role in helping you with this decoupling.

Collective Ownership

A pair has the right to check out any module & improve it.

Nobody has more authority than anybody else over a module or a technology.

Continuous Integration

The programmers check their code in & integrate several times per day.

This means that programmers are allowed to check any module out at any time, regardless of who else may have it ~~checked out~~ Checked out.

→ They first make sure that all the tests run. They integrate their new code into the existing code base.

→ If they broke anything that used to work, they fix it. Once all the tests run, they finish the check-in.

So XP teams will build the system many times each day.

They build the whole sys. from end to end.

If the final result of the sys. is an active Web site, they install that web site, probably on a testing server.

Sustainable Pace

A S/w proj. is not a sprint; it is a marathon.

In order to finish quickly, the team must run at a sustainable pace; it must conserve its energy & alertness.

The XP rule is that a team is not allowed to work overtime.

Open Workspace

The team works together in an open room. Tables are set up with workstations on them. The sound in this room is a buzz of conversation.
Each has the opportunity to hear when another is in trouble. Each knows the state of the other.
The programmers are in a position to communicate intensely

Working in a "room" environment may increase productivity by a factor.

The planning game

The essence of the planning game is the division of responsibility between business and development.

The business people customers decide how important a feature is & the developers decide how much that feature will cost to implement.

At the beginning of each release & each iteration, the developers give the customer a budget.

Simple Design

An ~~XP~~ team makes its designs as simple & expressive as they can be.

→ team first act will be to get the first batch of stories working in the simplest way possible.
Then team will add the infrastructure.

3 XP ~~mantras~~ mantras

① XP teams always try to find the simplest possible design option for the current batch of stories.

② once & only once - XPers don't tolerate duplication of code. whenever they find it, they eliminate it.

→ When we find those, we eliminate them by creating a fn or a base class.

→ We can turn those into fns. or use the TEMPLATE METHOD pattern.

③ The Act of eliminating redundancy for the team to create many abstractions & further reduce coupling.

cost-effective

An XP team seriously considers what will happen if it resists the temptation to add infrastructure before it is strictly needed.

Refactoring

Code tends to rot. As we add feature after feature & deal with bug after bug, the structure of the code degrades.

Left unchecked, this degradation leads to a tangled, unmaintainable mess.

XP teams reverse this ~~degradation~~ degradation through frequent refactoring.

Refactoring is the practice of making a series of tiny transformations that improve the structure of the system without affecting its behavior.

Refactoring is done continuously rather than at the end of the project, the end of the release, or the end of the iteration, or even the end of the day.

Metaphor Metaphor is the only XP practice that is not concrete & direct. It is the least well understood of all the practices of XP.

A metaphor guides all the developers to choose appropriate names, select appropriate locations for fns, create appropriate new classes & methods, & so on.