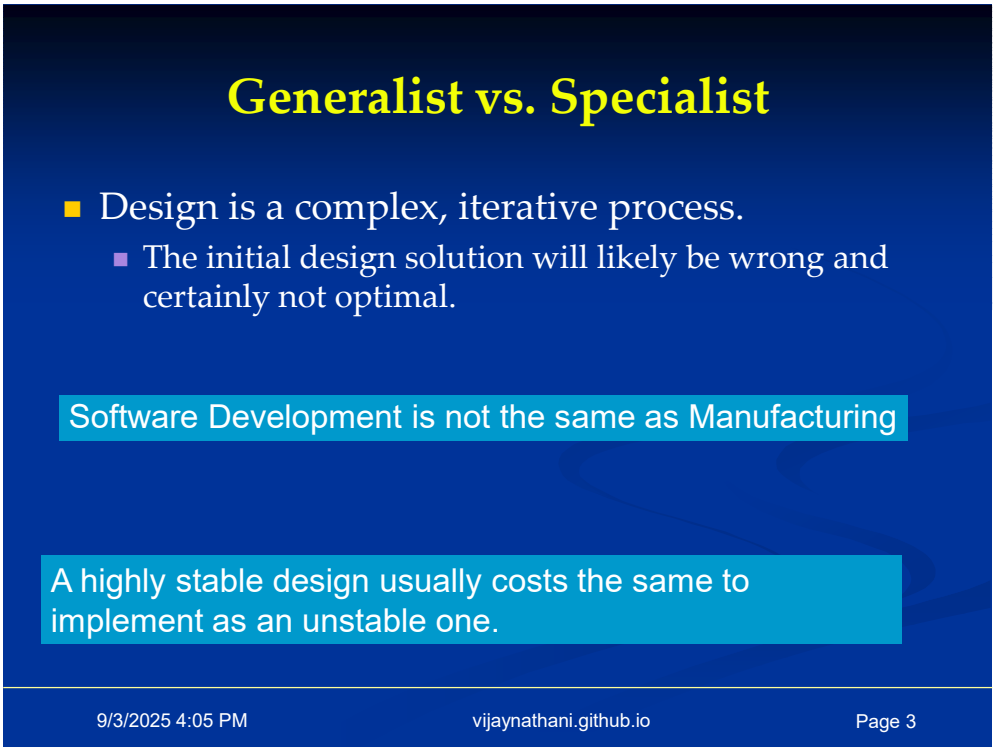
Agile Practices

A process is needed, when we would not do the same thing naturally.

Who owns the code?

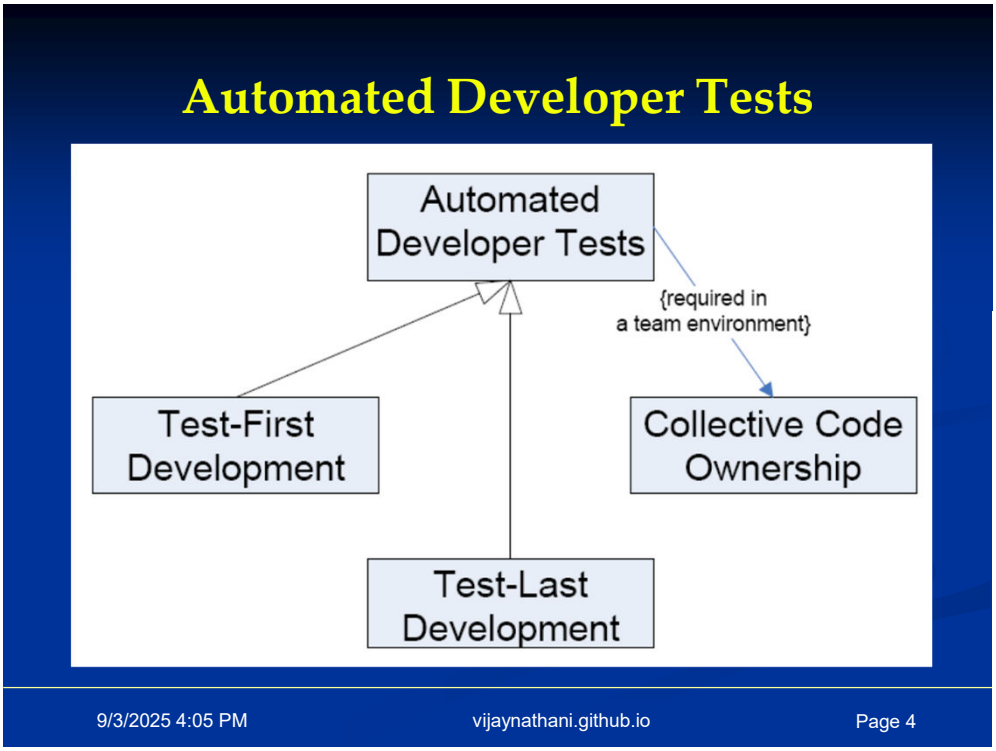
- Collective ownership is nothing more than an instantiation of the idea that products should be attributable to the team, not individuals who make up the team.
  - Strong Ownership / Loose Ownership are usually not good enough.



Programmers shift from design to coding when the problem is decomposed to a level of "primitives" that the designer has mastered. If the coder is not the same person as the designer, the designer's primitives are unlikely to match the coder's primitives, and trouble will result.

Manufacturing is a popular metaphor for software development.

> One inference from this metaphor: highly skilled engineers design; less skilled laborers assemble the products.
>
> This metaphor has messed up a lot of projects for one simple reason—software development is all design.



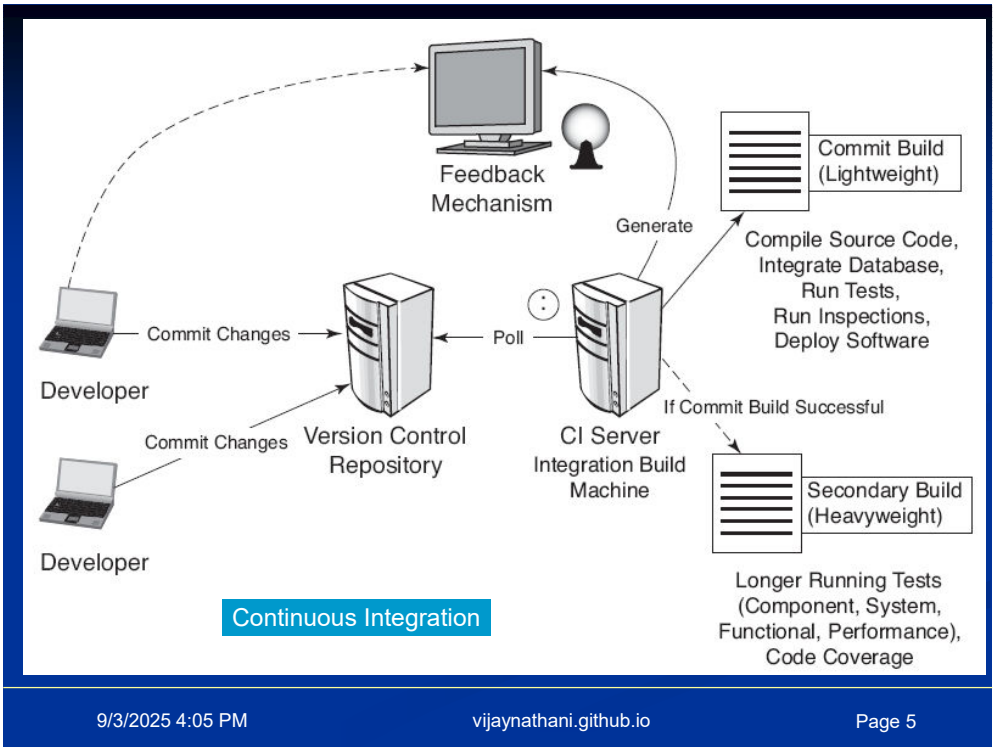
The way in which people run a 100m sprint is much different than running 10 Km race. In 100m, the runners are hefty and strong. In a 10 Km race, the runners are light and can change their running strategy.

Person who is ahead a the start in 10 KM will usually not win. Runners need to pace themselves. By cutting quality, we can go fast in the short run. Software is like 10 KM race.

Software development is like a 10 Km race.

We need to know whether we are done. We don't want to be almost complete. Project remain almost complete for many weeks / months.

Every failing test causes the line to stop.



For every check-in

Incremental Build

Fast tests (Unit)

Every Few hours

Full Build

Fast Tests

Slow Tests (Unit & Functional)

Information Radiator

==========

A build and test should take less than 10 minutes or else developers will avoid using it.

Code tested with build test suite is frequently separated from the database with mock objects to speed up execution.

These are fast tests

Acceptance tests are slow tests. They can run less frequently (but at least once a day)

CI means:

•All developers run private builds on their own workstations before committing their code to the version control repository to

ensure that their changes don’t break the integration build.

• Developers commit their code to a version control repository at least once a day.

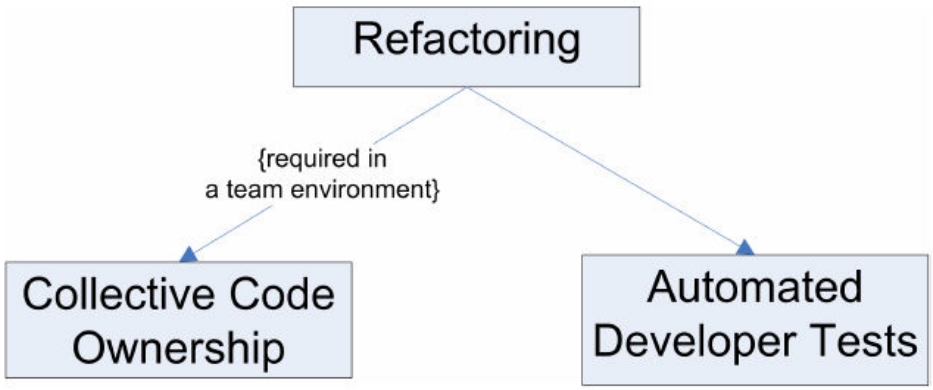
• Integration builds occur several times a day on a separate build machine.

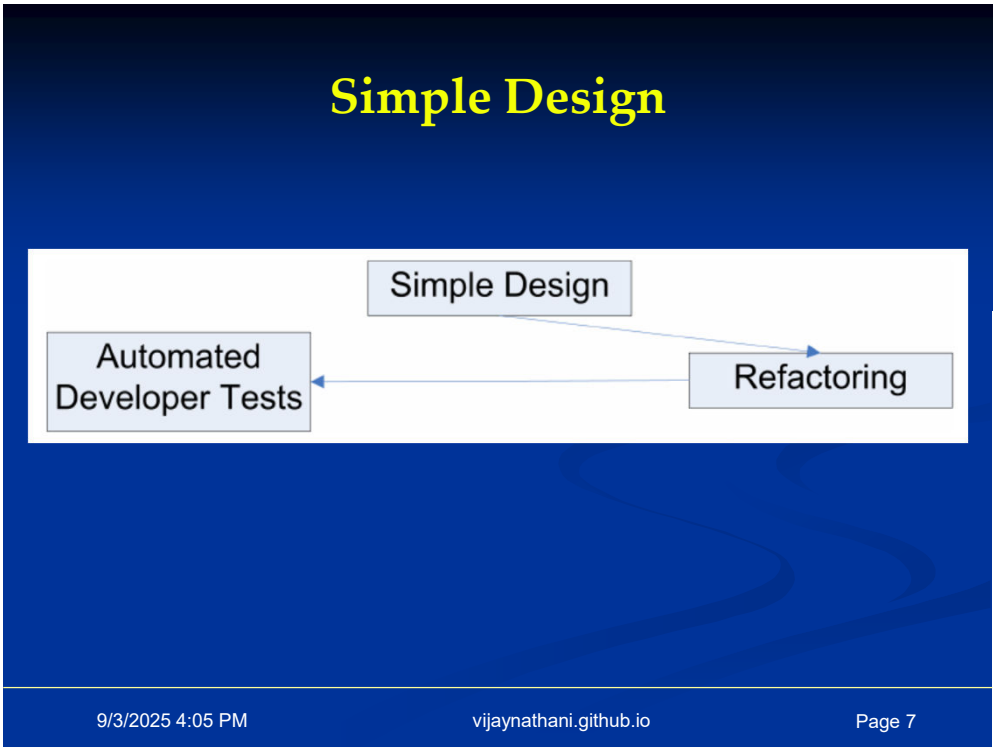
• 100% of tests must pass for every build.

• A product is generated (e.g., WAR, assembly, executable, etc.) that can be functionally tested.

• Fixing broken builds is of the highest priority.

• Some developers review reports generated by the build, such as coding standards and dependency analysis reports, to seek areas for improvement.


Refactoring
{required in
a team environment}
Collective Code
Ownership
Automated
Developer Tests

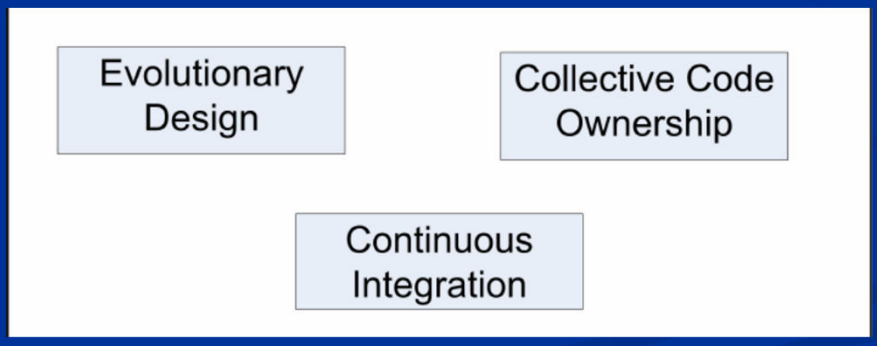


BDUF can be even worse than wasted time because of incorrect guesses. BDUF can also lead to self-fulfilled prophesies.

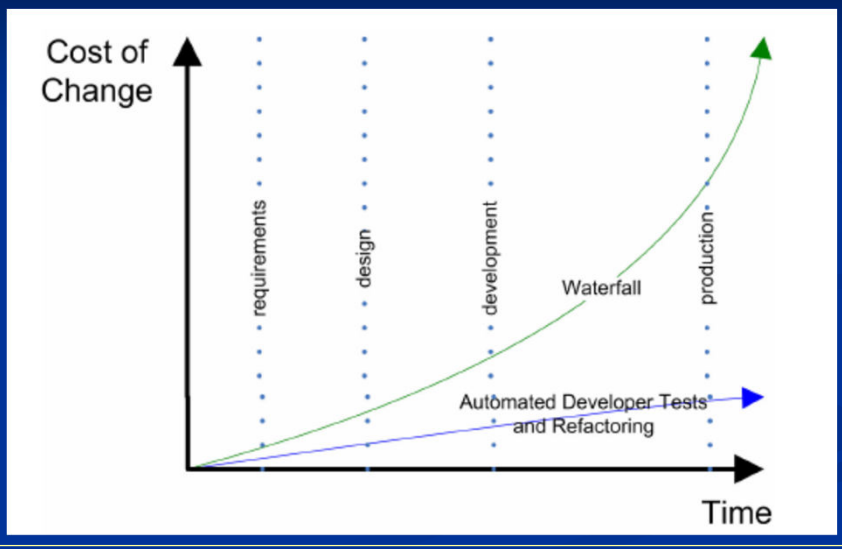
It's pretty well known that a Big Design Up-Front (BDUF) has some big problems. At the same time, most often we know some things from day one. It's a matter of balance.

Finally, a last remark regarding DDD and TDD: Domain Models are very suitable for TDD. Sure, you can also apply TDD with more database-oriented design, but I haven't been able to apply it as gracefully and productively as when I'm working with Domain Models.

Simple Design



Cost of Change in Waterfall & TDD

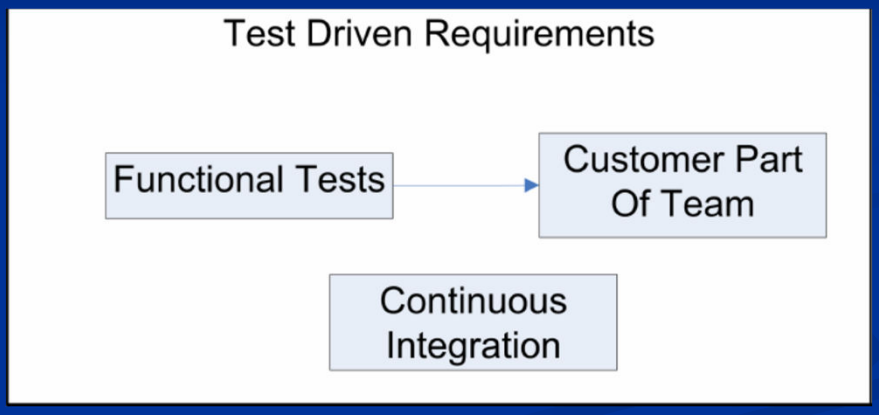


No matter where you are in the system life cycle, the system will change, and the desire to change it will persist throughout the life cycle. - *Bersoff, et al, 1980*

On an average, 35% of the requirements change during project execution

It is estimated that 25% to 50% of effort in waterfall goes in discussing / managing change requests.

Automated Functional Tests



Thoughtworks is primarily a XP shop.

Why is it Extreme?

- Because we take good practices to extreme levels (turning the knobs up to 10!)
  - Code reviews – pair programming
  - Testing – CI
  - Design – TDD
  - Feedback in minutes

If code reviews are good, we'll review code all the time (pair programming).

If testing is good, everybody will test all the time (unit testing), even the customers (functional testing).

If design is good, we'll make it part of everybody's daily business (Refactoring).

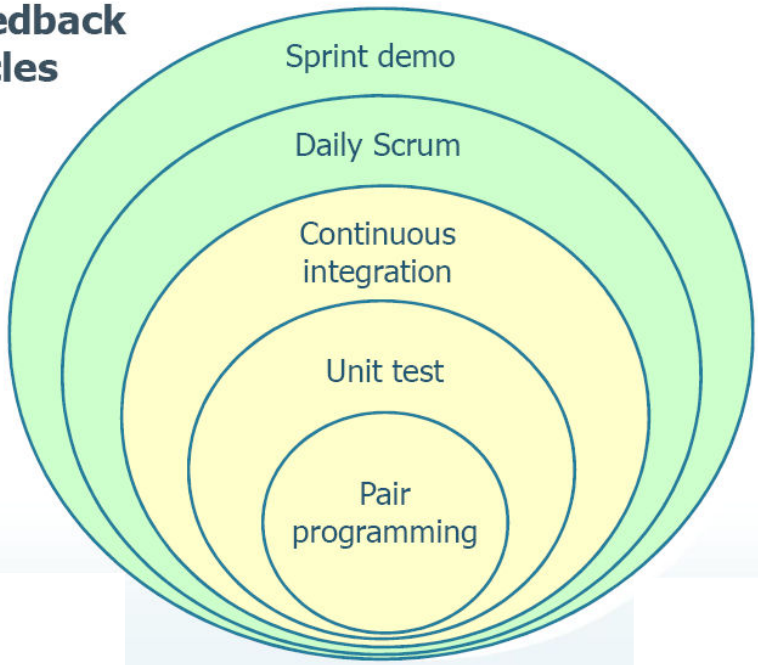
If simplicity is good, we'll always leave the system with the simplest design that supports its current functionality. (The simplest thing that could possibly work.

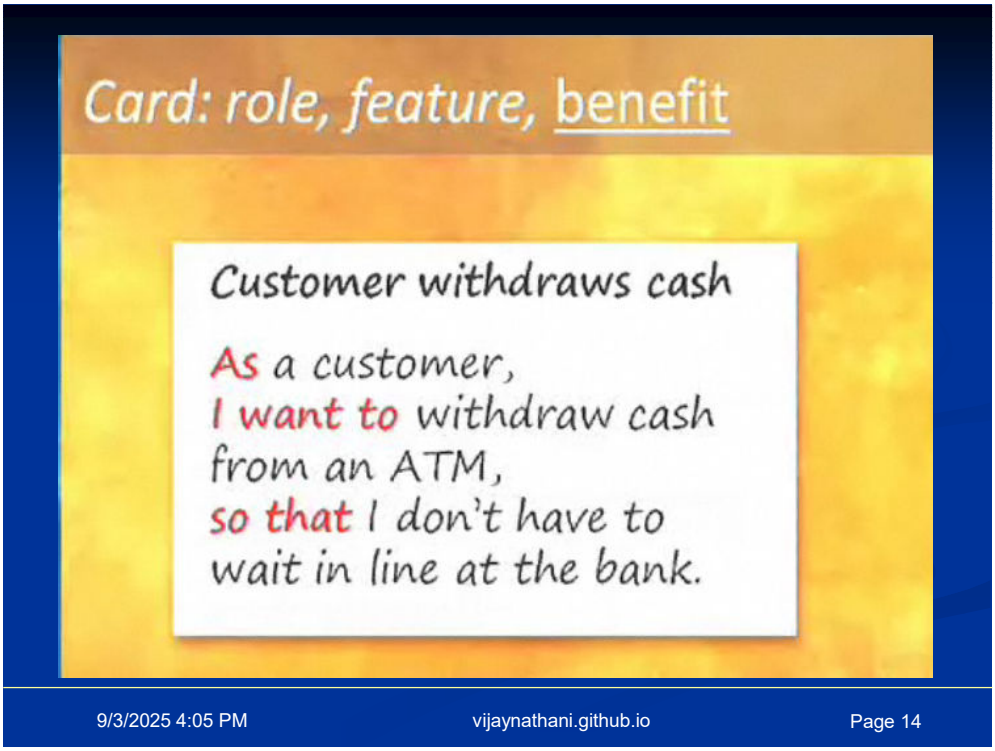
If architecture is important, everybody will work defining and refining the architecture all the time (Metaphor).

If integration testing is important, then we'll integrate and test several time a day (continuous integration).

If feedback is good, we'll get feedback quickly -- seconds and minutes and hours, not weeks and months and years (the Planning Game).

Feedback cycles





User Story Example

“I would have written a shorter letter, but I didn’t have time” – Mark Twain.

The main purpose of a story card is to act as a reminder, to discuss the feature.

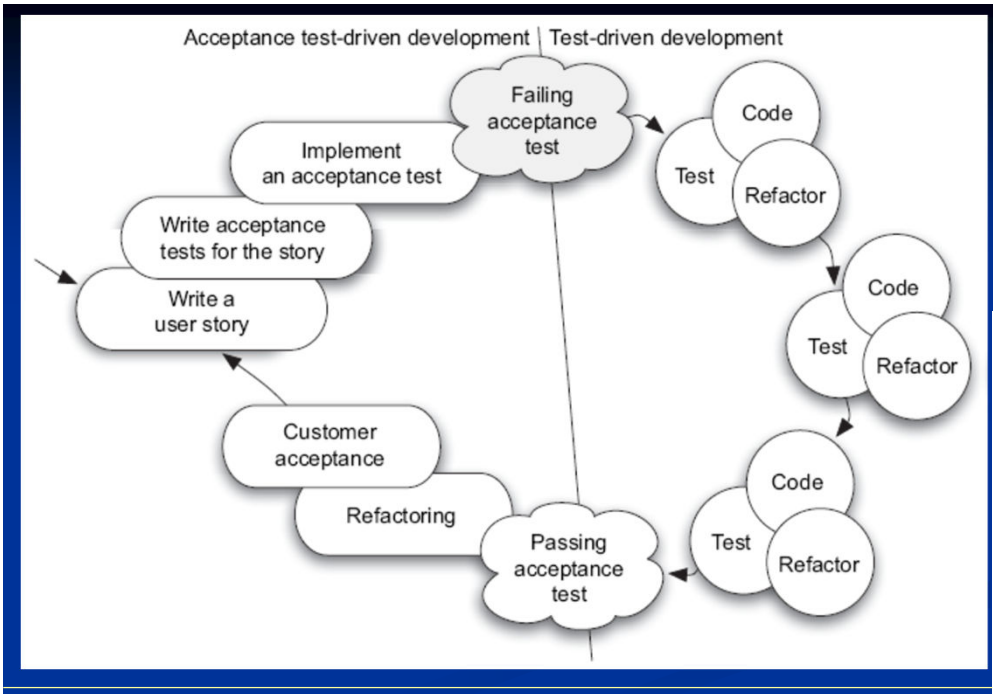
In XP, the estimate is given in ideal hours. These are written at the corner of the card.

User Story Characteristics

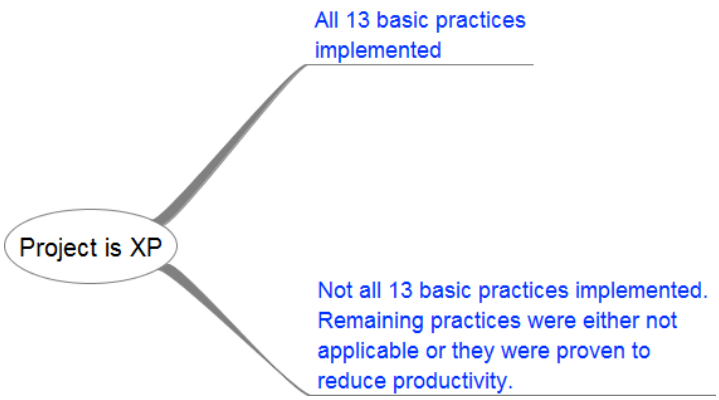
- I – Independent
- N – Negotiable
- V – Valuable
- E – Estimatable
- S – Small
- T – Testable

A User story should be closed i.e. People must get a feeling of accomplishment, when it is done.

Every User story is 0% done or 100% done.

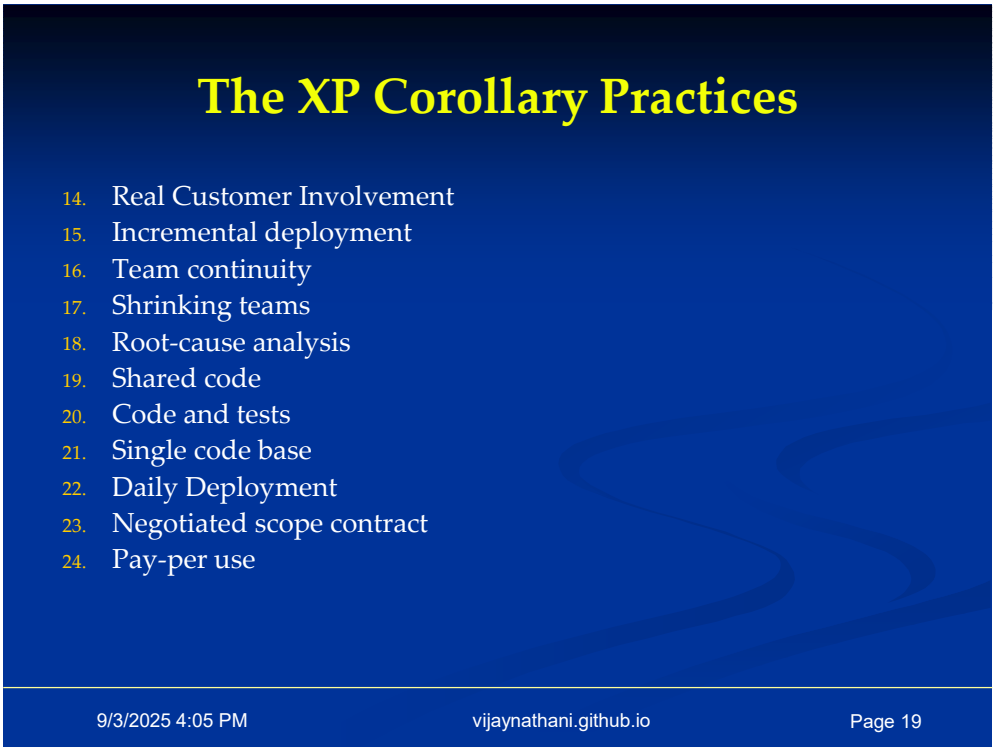

Acceptance test-driven development
Test-driven development
Failing acceptance test
Implement an acceptance test
Write acceptance tests for the story
Write a user story
Customer acceptance
Refactoring
Passing acceptance test
Code
Test
Refactor
Code
Test
Refactor
Code
Test
Refactor

Is my Project XP?



The XP Basic Practices

1. Sit together
2. Whole Team
3. Informative Workspace
4. Energized work
5. Pair Programming
6. Stories
7. Weekly cycle
8. Quarterly cycle
9. Slack
10. Ten-minute build
11. Continuous Integration
12. Test-first programming
13. Incremental Design



Flicr does 10+ deployments everyday. (Documented in an agile talk by James Shore)

Stackoverflow deploys everyday (http://itc.conversationsnetwork.org/audio/download/ITC.SO-Episode70-2009.10.13.mp3)

Amazon deploys everyday.

IMVU.com does continuous deployment e.g. deployment 50 times a day. For every checkin, if all tests pass, deploy.