CSE- 321
Software Engineering

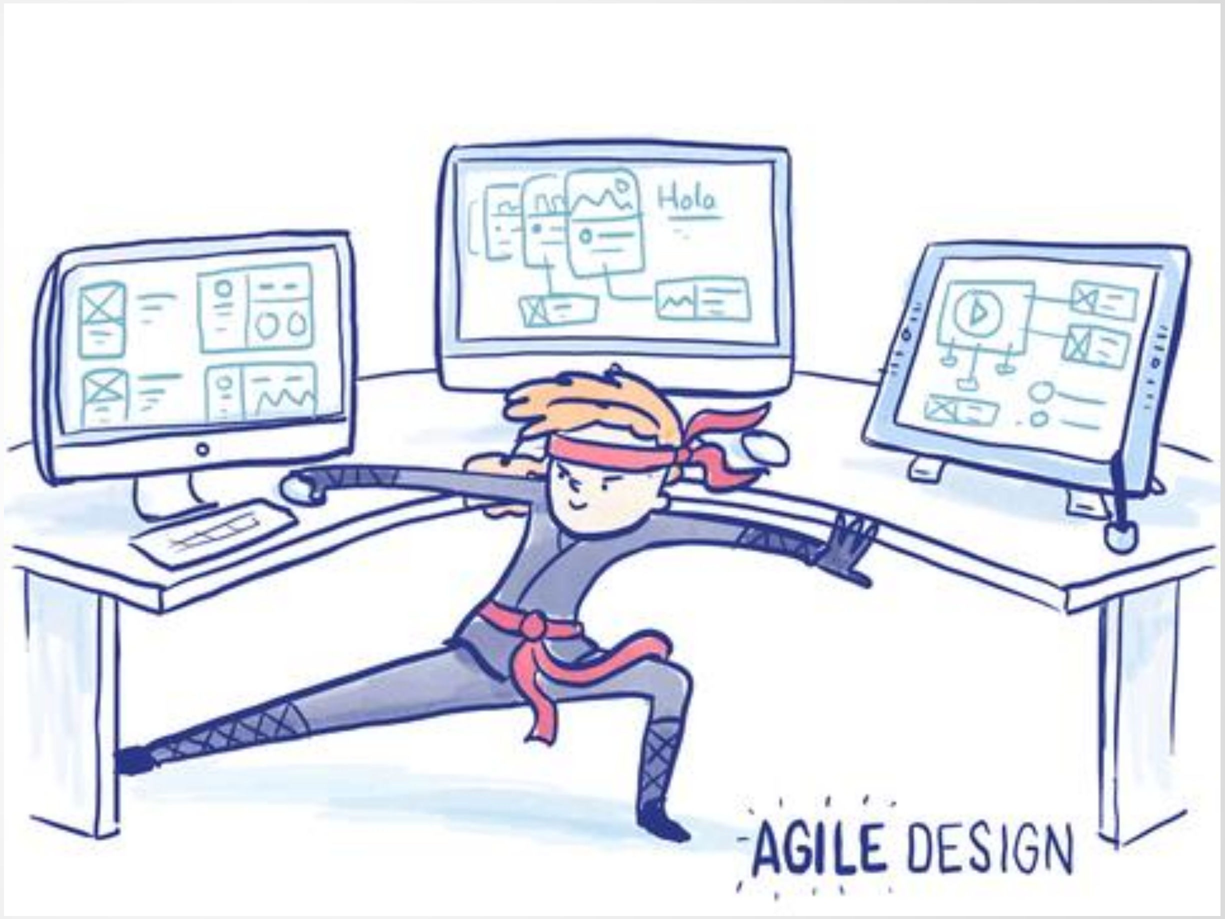
Lecture: 09

Software Processes
(part-03)

Fahad Ahmed
Lecturer, Dept. of CSE
E-mail: fahadahmed@uap-bd.edu

Lecture Outlines

- ❖ **Agile Model**



Rather than doing all of one thing at a time, is better to do a little of everything all the time. :)

eXtreme Programming (XP)

What is XP?

eXtreme Programming (XP)

XP is a

- lightweight,
- efficient,
- low-risk,
- flexible,
- predictable,
- scientific, and
- fun

way to develop software.

eXtreme Programming (XP)

eXtreme Programming (XP):

Extreme Programming is a subset of the Agile framework that helps your development team to produce a working software model in very short iterations.

Extreme programming (XP) is a Software Development Methodology which is intended to **improve software quality and responsiveness to changing customer requirements.**

- **Dynamically changing** software requirements
- Risks caused by fixed time projects using new technology
- Small, co-located extended development team
- The technology you are using allows for automated unit and functional tests

eXtreme Programming (XP)

eXtreme Programming (XP):

Extreme **Programming technique** is very helpful when there is constantly changing demands or requirements from the customers or when they are not sure about the functionality of the system.

It advocates frequent "releases" of the product in short development cycles, which inherently improves the productivity of the system and also introduces a checkpoint where any customer requirements can be easily implemented. The XP develops software keeping customer in the target.

eXtreme Programming (XP)

XP promises

- To reduce project risk,
- To improve responsiveness to business changes,
- To improve productivity throughout the life of a system,
- To add fun to building software in teams

all at the same time.

eXtreme Programming (XP)

Why is it called “eXtreme”?

eXtreme Programming takes the effective principles and practices to extreme levels.

- 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 times a day (continuous integration).

- If short iterations are good, we'll make the iterations really, really short—seconds and minutes and hours, not weeks and months and years (the Planning Game).

XP: Risk- The Basic Problem

- ❖Schedule slips
- ❖Project canceled
- ❖System goes sour
- ❖Defect rate
- ❖Business misunderstood
- ❖Business changes
- ❖False feature rich
- ❖Staff turnover

XP: Risk- The Basic Problem

How does XP address risks?

- **Schedule slips**
  - short Release cycles - a few months at most,
  - Iterations of customer requested features - one- to four-week.
  - Tasks - one- to three-day
  - implementing the highest priority features first
- **Project canceled**
  - XP asks the customer to choose the smallest release that makes the most business sense.
- **System goes sour**
  - comprehensive suite of tests, which are run and re-run after every change.
  - XP always keeps the system in prime condition.
- **Defect rate - XP** tests from the perspective of
  - programmers - writing tests function-by function
  - customers - writing tests program-feature-by-program-feature.

XP: Risk- The Basic Problem

How does XP address risks?

- **Business misunderstood**
  - customer to be an integral part of the team.
- **Business changes**
  - XP shortens the release cycle, so there is less change during the development of a single release.
- **False feature rich**
  - only the highest priority tasks are addressed.
- **Staff turnover**
  - programmers accept responsibility for estimating and completing their own work
  - human contact among the team

Practices of Extreme Programming

12 Practices

- The Planning Game
- Small Releases
- System Metaphor
- Simple Design
- Continuous Testing
- Refactoring
- Pair Programming
- Collective Code Ownership
- Continuous Integration
- 40-Hour Work Week
- On-Site Customer
- Coding Standards

Practices of Extreme Programming

The Planning Game:

In this practice, the development team and customers host two planning meetings.

In the release planning meeting, both the parties decide which features of the working software they plan to build. These items are then added to the backlog.

Practices of Extreme Programming

Small releases

In this programming practice, the XP team releases the first software version as soon as possible.

They then further develop the product by making small changes in every iteration.

Small releases are great for the XP team because it allows them to:

- Receive frequent feedback
- Detect bugs earlier
- Monitor how the product works easily

Metaphor

A system metaphor is what's used to ensure that your code is easy to understand.

For example:
A function name like **Open_loot_box()** is self explanatory.
Any developer can easily grasp that this piece of code allows users to open a loot box.

Practices of Extreme Programming

Simple Design

An XP team starts with a simple structure and then lets it evolve over every iteration. If there is any unnecessary complexity in the code, it is removed.

Test Driven Development (TDD)

Before the code is even written, the team creates an automated unit test that it needs to pass.

Only after this acceptance test is in place, does the developer write a minimal amount of code to pass the automated testing process.

Practices of Extreme Programming

Code Refactoring

refactoring is **cleaning** up your code.
It requires developers to continuously improve code by:

- ❖ Removing redundant code
- ❖ Editing out unnecessary functions

Practices of Extreme Programming

Pair Programming

In pair programming, two developers sit together and work on the same code on the same system.

One software developer writes the code, and the other reviews it, simultaneously.

.

Collective Ownership

- ❖ Since the XP team works together, they take ownership of the code.
- ❖ If something does go wrong, there's no finger-pointing.
- ❖ Everyone is **equally** responsible for the design of the software.
- ❖ That means **anyone** can edit the code or pitch new ideas to improve the work

Practices of Extreme Programming

Continuous Integration –
Every time a task is completed it should be integrated and tested, integration should take place many times a day

40-Hour Week –
Only work 40 hours a week, no overtime for two weeks in a row

On-Site Customer –
The customer sits together with the XP team to make sure the team is creating the product to their exact specifications.

If the customer is not available all the time, the role can be filled by experts like product managers, product owners, business analysts.

Practices of Extreme Programming

Continuous Integration –
Every time a task is completed it should be integrated and tested, integration should take place many times a day

40-Hour Week –
Only work 40 hours a week, no overtime for two weeks in a row

On-Site Customer –
The customer sits together with the XP team to make sure the team is creating the product to their exact specifications.

If the customer is not available all the time, the role can be filled by experts like product managers, product owners, business analysts.

Practices of Extreme Programming

Coding Standard

Always remember:

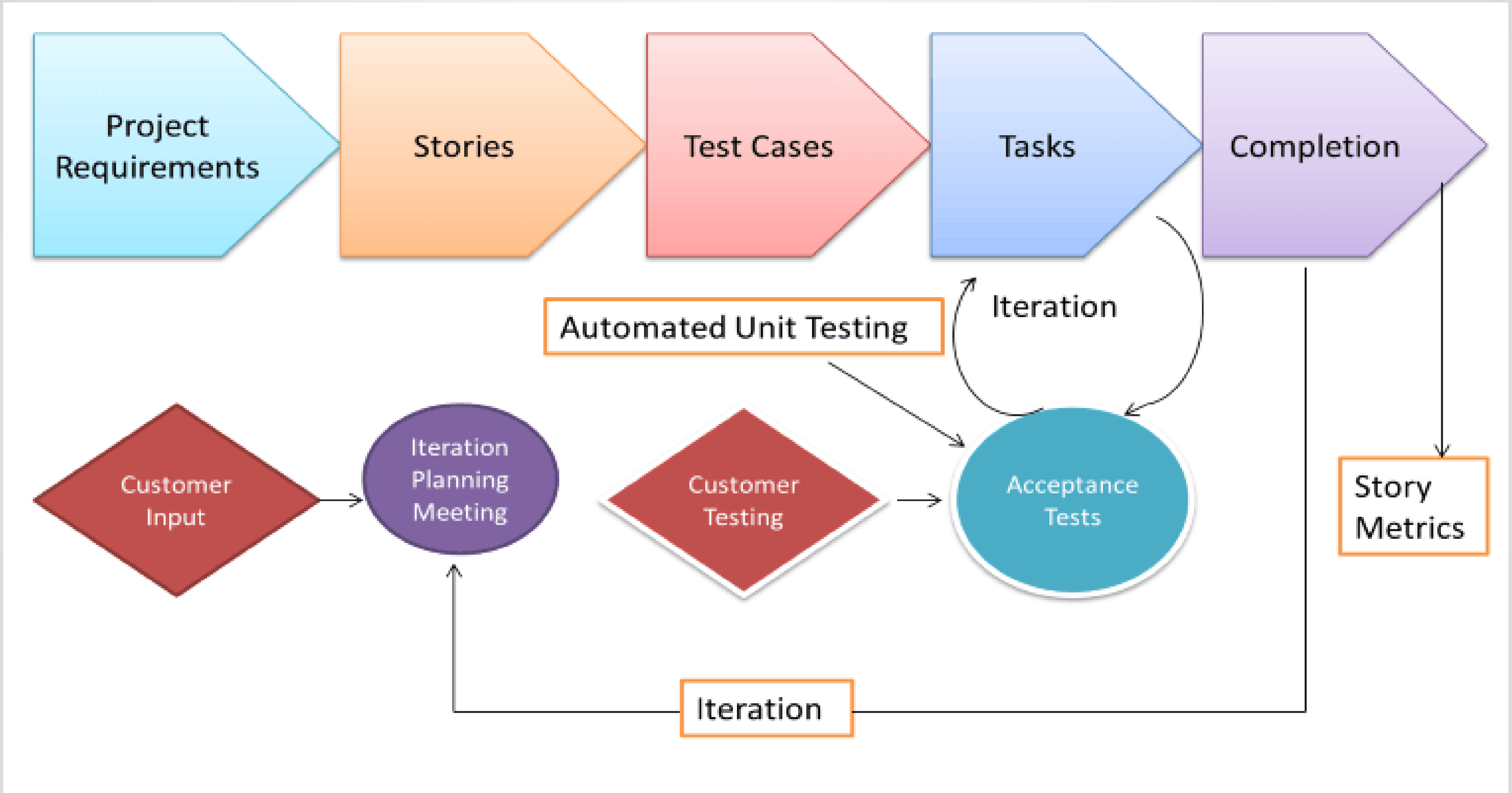
The whole team should adhere to the **same** standards.

Practices of Extreme Programming

Where Will XP not work?

- Anytime you are using a technology with an inherently exponential cost curve.
- An environment where it takes a long time to gain feedback
- Any place where the physical environment is not suited for communication
- XP will not work well with a large amount of people

eXtreme Programming (XP) : Working phases



Extreme Programming (XP)

eXtreme Programming (XP)

There are 6 phases available in Agile XP method, and those are explained as follows:

Planning

- Identification of stakeholders and sponsors
- Infrastructure Requirements
- Security related information and gathering
- Service Level Agreements and its conditions

Analysis

- Capturing of Stories in Parking lot
- Prioritize stories in Parking lot
- Scrubbing of stories for estimation
- Define Iteration SPAN(Time)
- Resource planning for both Development and QA teams

Design

- Break down of tasks
- Test Scenario preparation for each task
- Regression Automation Framework

eXtreme Programming (XP)

Execution

- Coding
- Unit Testing
- Execution of Manual test scenarios
- Defect Report generation
- Conversion of Manual to Automation regression test cases
- Mid Iteration review
- End of Iteration review

Wrapping

- Small Releases
- Regression Testing
- Demos and reviews
- Develop new stories based on the need
- Process Improvements based on end of iteration review comments

Closure

- Pilot Launch
- Training
- Production Launch
- SLA Guarantee assurance
- Review SOA strategy
- Production Support

eXtreme Programming (XP)

User Stories

- A short description of the behavior of the system from the point of view of the Customer
- Use the Customer's terminology without technical jargon
- One for each major feature in the system
- Must be written by the users
- Are used to create time estimates for release planning
- Replace a large Requirements Document

User stories have three crucial aspects:

Card

Enough information to identify the story

Conversation

Customer and Programmers discuss the story to elaborate on the details

Verbal when possible, but documented when required

Confirmation

Acceptance tests to confirm that the story has been properly implemented

Differences between Scrum framework and XP

Sprint length

A Scrum team works on iterations or sprints that last from 2-4 weeks. Whereas, an XP iteration cuts it down to 1-2 weeks.

Flexibility

According to the Scrum framework, a Scrum team **doesn’t allow changes** in the sprint. Once the sprint backlog has been decided and the sprint starts, **nothing new** can be added to the backlog.
XP teams can change the sprint backlog items during an iteration, as long work hasn’t begun on it.

Differences between Scrum framework and XP

Sequences

Scrum is like an open-world game, where there are many missions and stories, but you’re free to choose the order in which you can play them.

In Scrum, the product owner prioritizes the product backlog items (missions), but it’s up to the Scrum master and the Scrum team to choose the sequence

XP is like **Super Mario**.

The levels are already set, and you **have to complete them in order.**

The work is already prioritized, and the team has to complete tasks in a sequence for maximum speed and efficiency.

The Best Way To Manage Your XP Team

1. Craft an epic iteration strategy with Gantt View

An XP team needs to do all their programming on a tight schedule.

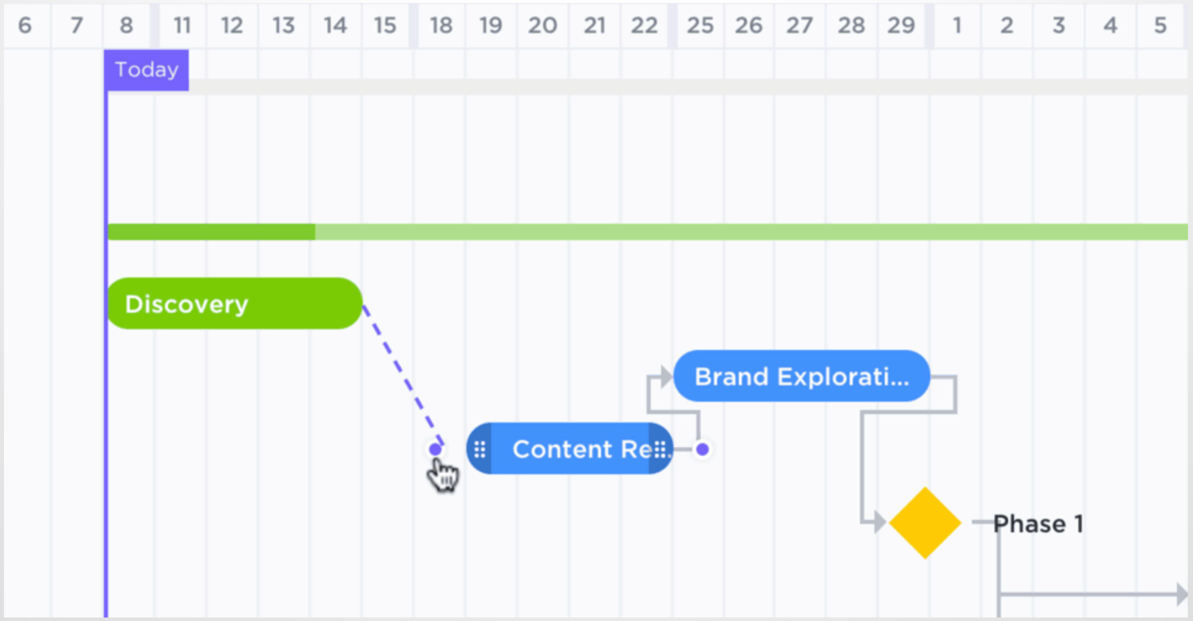
Otherwise, the project would remain unfinished like the game Half-Life 3, whose development process has been stagnant for 10+ years!

The Best Way To Manage Your XP Team

Gantt View

Gantt charts visualize the tasks that need to be completed in sequence, kind of like a project timeline! Each task has a start and end date, so your team's development time won't go overboard.

- Automatically **reschedule your dependencies** whenever you make a change
- Visualize **progress percentage** of project based on completed tasks against total tasks
- Determine the **critical path**, a chain of tasks that is critical to your project's completion
- Create **dependent tasks** by drawing a line between two tasks like so:

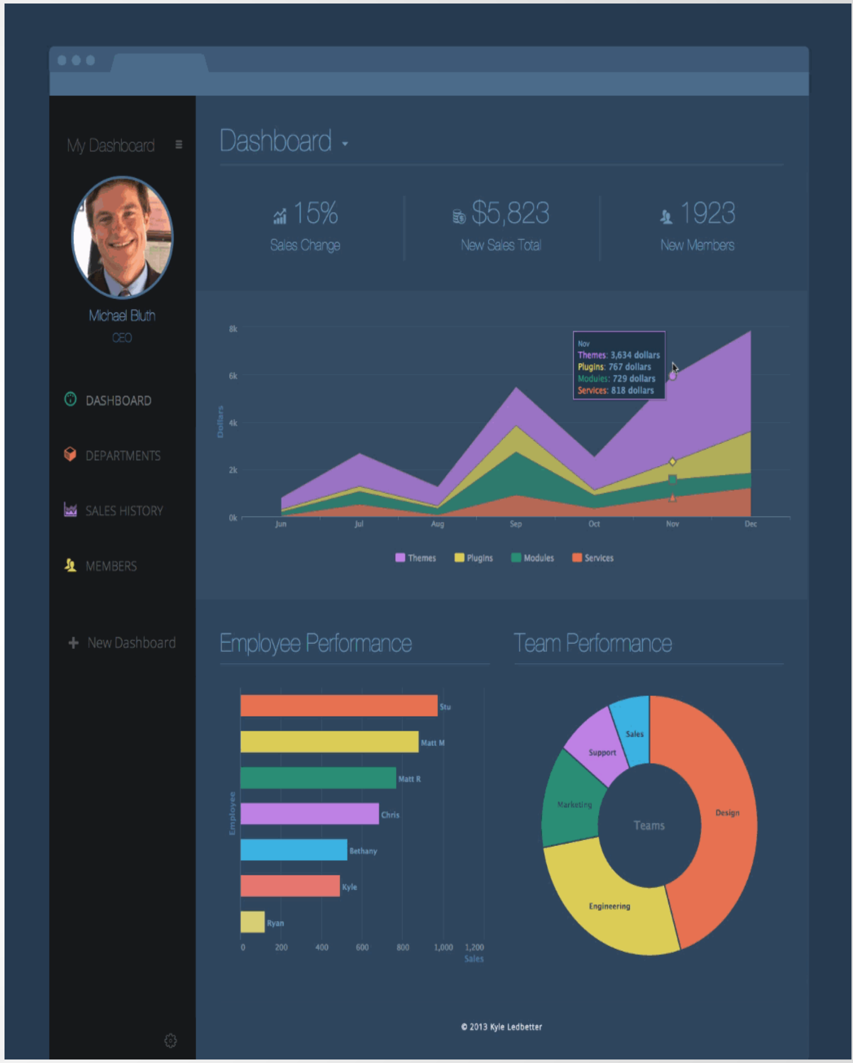


The Best Way To Manage Your XP Team

2. Power up your sprint performance with Dashboards

Gantt charts visualize the Your XP team needs to be highly motivated at all times.

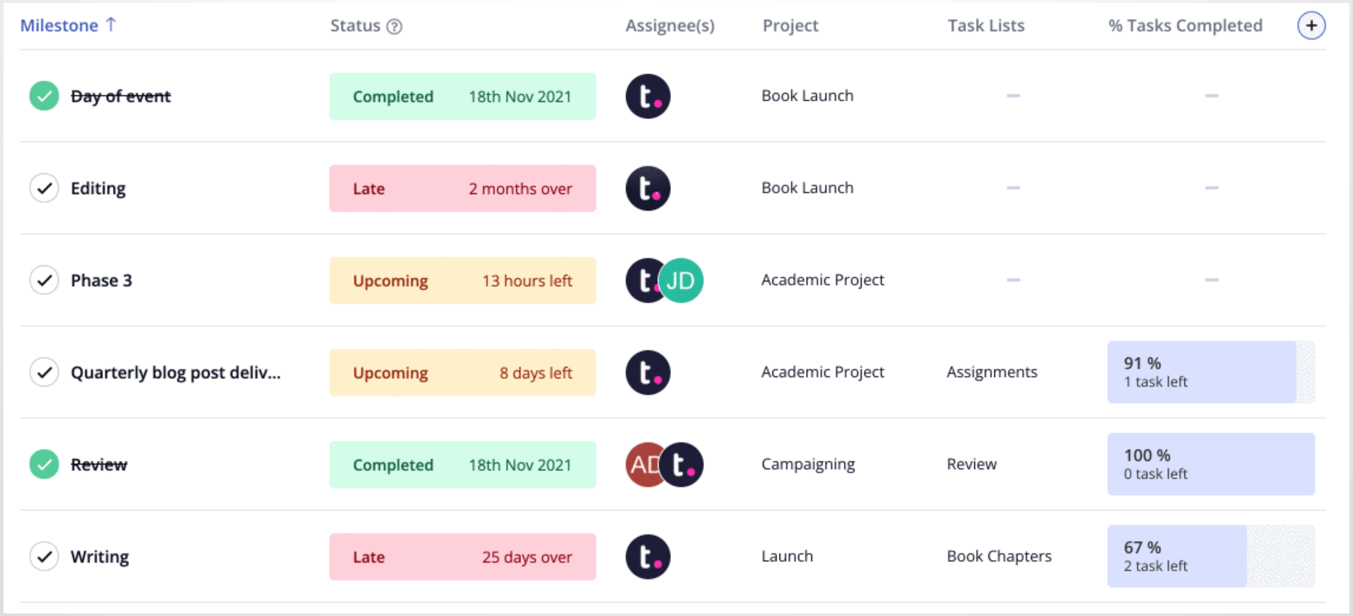
- ❖ **Velocity Charts:** helps the team figure out how many sprint backlog items they can handle in a single sprint
- ❖ **Burn-up Charts:** shows the amount of work remaining in the project
- ❖ **Burndown Charts:** determines whether the team will be able to finish their tasks before the deadline
- ❖ **Cumulative Flow Diagrams:** reveals hidden bottlenecks in the development process



The Best Way To Manage Your XP Team

3. Workplace Reports

- **Task Completed Report:** displays the number of tasks completed by each team member
- **Worked On Report:** displays the number of tasks each team member has been a part of
  - **Who's Behind Report:** reveals team members with the most number of unfinished tasks
  - **Time Tracked Report:** measures time spent on tasks by each team member
- **Workspace Points Report:** motivates team members to finish more tasks by rewarding them with points



Milestone ↑	Status		Assignee(s)	Project	Task Lists	% Tasks Completed
✓ ~~Day of event~~	Completed	18th Nov 2021	t.	Book Launch	–	–
✓ Editing	Late	2 months over	t.	Book Launch	–	–
✓ Phase 3	Upcoming	13 hours left	t. JD	Academic Project	–	–
✓ Quarterly blog post deliv...	Upcoming	8 days left	t.	Academic Project	Assignments	91 % 1 task left
✓ ~~Review~~	Completed	18th Nov 2021	AD t.	Campaigning	Review	100 % 0 task left
✓ Writing	Late	25 days over	t.	Launch	Book Chapters	67 % 2 task left

The Best Way To Manage Your XP Team

4. Speed up communication with Comments

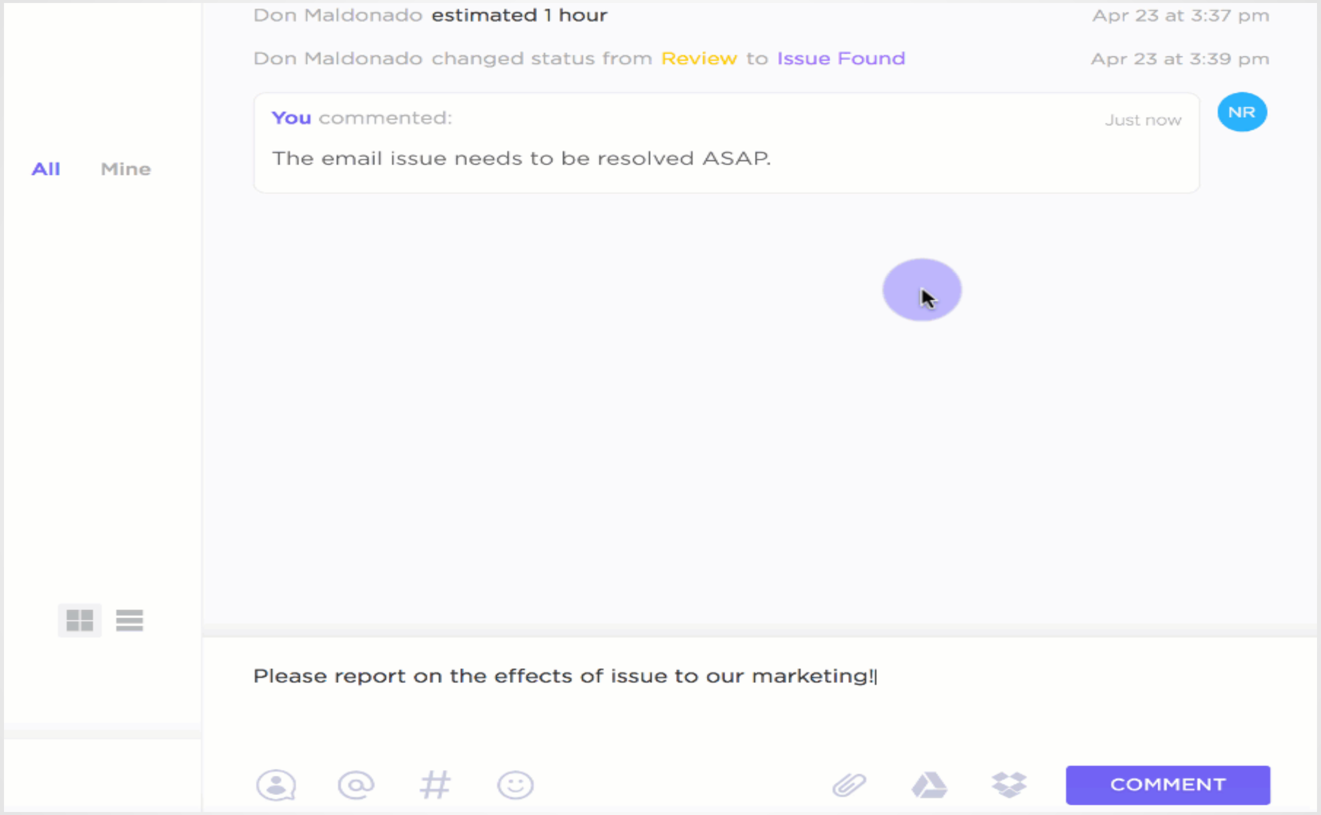
Besides notifying your team members about something, you can also use comments to do a whole lot more!

- Sharing documents and files
- Sharing links
- Tagging users
- Mentioning other tasks
- Creating comment threads

The Best Way To Manage Your XP Team

4. Speed up communication with Comments

Your customers can even leave effective feedback with Assigned Comments.



Don Maldonado estimated 1 hour Apr 23 at 3:37 pm

Don Maldonado changed status from Review to Issue Found Apr 23 at 3:39 pm

You commented: Just now

The email issue needs to be resolved ASAP.

Please report on the effects of issue to our marketing!

The Best Way To Manage Your XP Team

Flexible Views: organize your tasks into a checklist, Kanban board, or a calendar format

Automations: save tons of time by automating work processes

Profiles: see what your team members are working on in real-time

Native Time Tracking: manage your time efficiently for efficient remote project management

Custom Statues: create your own statuses to suit your team’s workflow

Agile Model : Key terms

Spike: It generally refers to a too large and complex user story in software development that cannot be estimated until the development team runs a timeboxed investigation. These stories can be used for various activities like research, design, exploration, prototyping, etc. Spikes are usually created to resolve some technical issues and design problems in the project.

Zero Sprint: It generally refers to the first step or pre-preparation step that comes just before the first sprint. It includes all activities such as setting a development environment, preparing backlog, etc.

A **velocity** is basically a measurement unit that measures or calculates how much work an agile development team can successfully complete in a single sprint and how much time will be required to finish a project.

Agile Model

What are Burn-up and Burn-down charts in Agile?

Burn-up Chart: It is a type of chart that is used to display or represent the amount of work that has been completed and the total amount of work for a sprint or iteration.

Burn-down Chart: It is a type of chart that is used to display or represent the amount of work that is remaining to be completed in the project. These charts are very simple and easy to understand.

No More Today

When to use Agile and Non-Agile models

Project Attributes	Agile Model	Non-Agile Model
Requirement of the Project	Requirements in Agile model can **change as per the customer requirement**. Sometimes requirements are not very clear.	In Non-Agile models the requirements are very clear before entering into the development phases. Any change in the requirement is not easily accepted during the development phases.
Size of the Project	The Project **size is small** in Agile model hence small team is required.	But in Non-Agile models the Project size is usually big hence big team is required.
Design of the Project	In Agile model the **architecture is made as per the current requirements** but is designed to be flexible.	In Non-Agile models the architecture is made as per the current requirements as well as for future requirements.
Type of Customers	Agile methodology is followed by the collaborated, dedicated collated and knowledgeable customers.	In Non-Agile models the customers are of Contract provisions.
Developers required	In Agile model the developers should be knowledgeable, analytically strong, collated and collaborative.	In Non-Agile models the developers should be more Plan Oriented.

Agile Model

Advantage :

- Frequent Delivery
- Face-to-Face Communication with clients.
- Efficient design and fulfils the business requirement.
- Anytime changes are acceptable.
- It reduces total development time.

Disadvantages :

- Highly dependent on **clear customer requirements**
- **Quite Difficult** to predict time and effort **for larger projects**
- **Not suitable** for **complex projects**
- Lacks documentation efficiency
- Increased maintainability risks

Difference between Agile and Waterfall Model

Key Difference

1. Waterfall is a Liner Sequential Life Cycle Model whereas Agile is a continuous iteration of development and testing in the software development process.

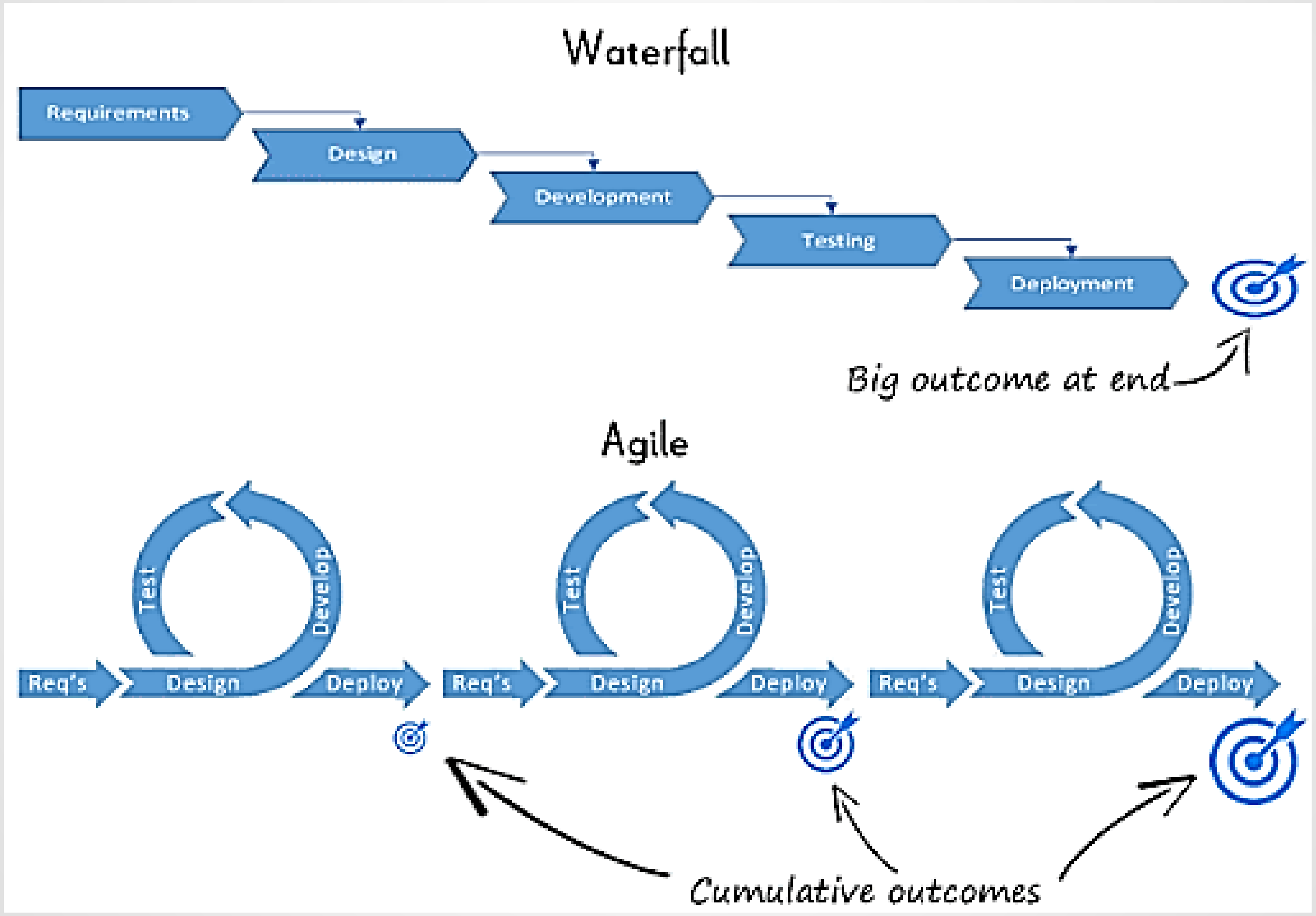
2. Agile methodology is known for its flexibility whereas Waterfall is a structured software development methodology.

3. Agile follows an incremental approach whereas the Waterfall methodology is a sequential design process.

4. Agile performs testing concurrently with software development whereas in Waterfall methodology testing comes after the “Build” phase.

5. Agile allows changes in project development requirement whereas Waterfall has no scope of changing the requirements once the project development starts.

Difference between Agile and Waterfall Model



Comparison of Various SDLC Models

Properties of Model	Water-Fall Model	Incremental Model	Spiral Model	Rad Model
Planning in early stage	Yes	Yes	Yes	No
Returning to an earlier phase	No	Yes	Yes	Yes
Handle Large-Project	Not Appropriate	Not Appropriate	Appropriate	Not Appropriate
Detailed Documentation	Necessary	Yes but not much	Yes	Limited
Cost	Low	Low	Expensive	Low
Requirement Specifications	Beginning	Beginning	Beginning	Time boxed release
Flexibility to change	Difficult	Easy	Easy	Easy
User Involvement	Only at beginning	Intermediate	High	Only at the beginning
Maintenance	Least	Promotes Maintainability	Typical	Easily Maintained
Testing	After completion of coding phase	After every iteration	At the end of the engineering phase	After completion of coding
Overlapping Phases	No	Yes	No	Yes
Re-usability	Least possible	To some extent	To some extent	Yes
Working software availability	At the end of the life-cycle	At the end of every iteration	At the end of every iteration	At the end of the life cycle
Objective	High Assurance	Rapid Development	High Assurance	Rapid development
Team size	Large Team	Not Large Team	Large Team	Small Team
Customer control over administrator	Very Low	Yes	Yes	Yes

Question:

1. How do I develop an Agile methodology mind-set while implementing a project?
2. Is Agile the best methodology for Mobile app development? Is Agile methodology appropriate for all projects?
3. Will agile methodology work for data warehousing projects?
4. Is it possible to run scrum without a scrum master? If yes, in what scenarios?
5. Does Scrum work for single person projects?
6. What kind of projects can benefit from XP? What projects are “too big” and therefore outside the scope of XP?
7. Should Your Agile Team Use XP Methodology?
8. What are different project management tools that are mostly used in Agile?
9. What is the next generation software development methodology after Agile?

https://www.quora.com/

Thanks to All