Agile Software Development

Understanding XP

(Unit 2)

Dr. Sreedhar Bhukya

Professor, Department of CSE, SNIST

Extreme Programming - An Introduction

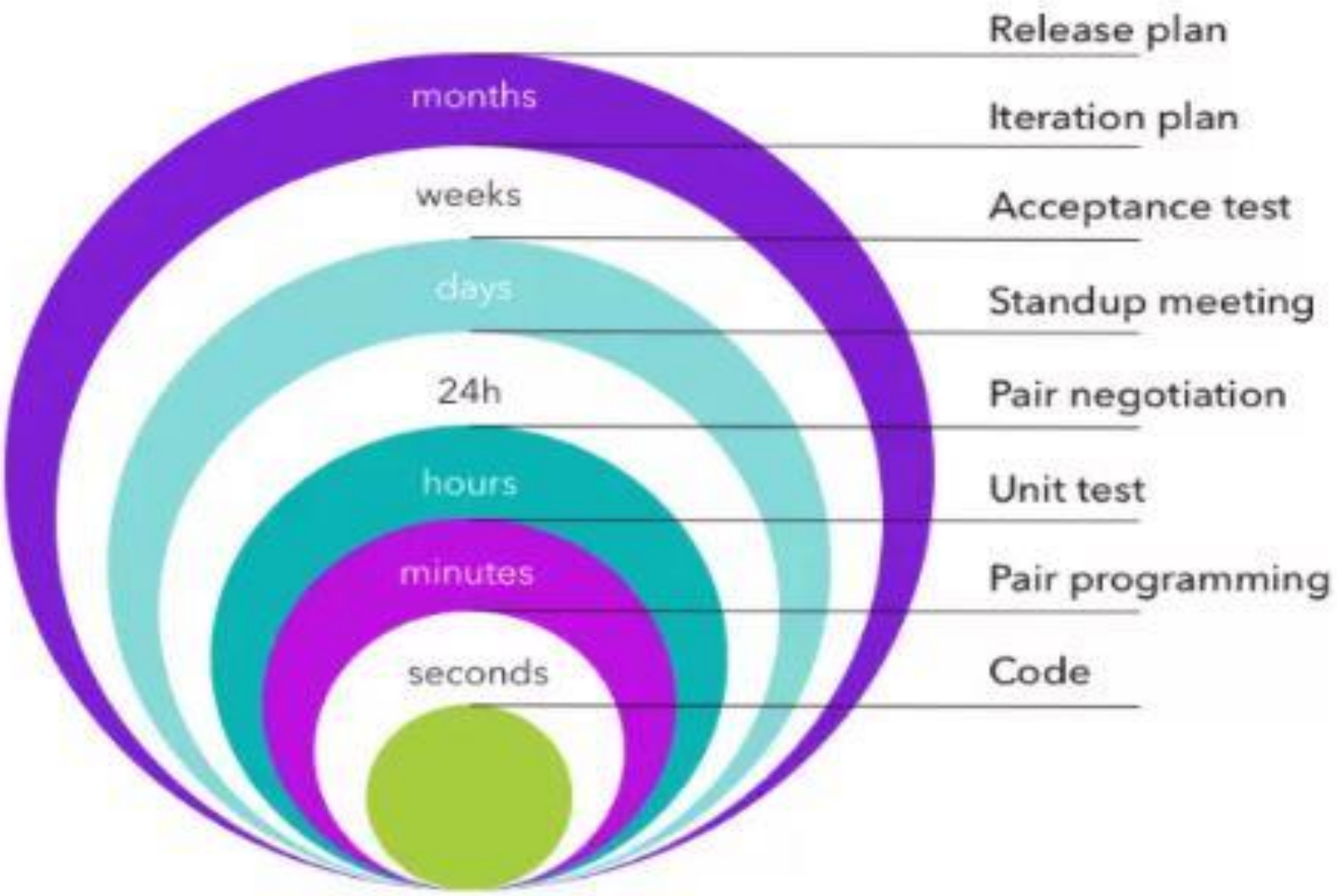
Extreme Programming (XP): is an Agile software development methodology that focuses on delivering high-quality software through frequent and continuous feedback, collaboration, and adaptation.

When Applicable

The general characteristics where XP is appropriate were described by Don Wells on:

- 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

XP Feedback Loops



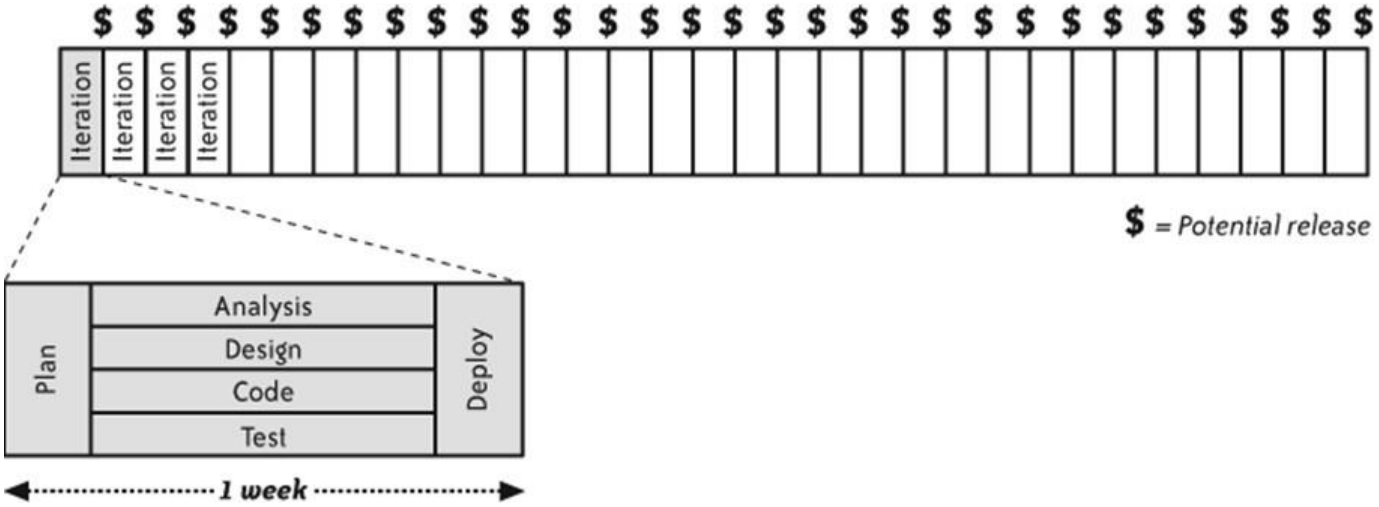
Extreme Programming - An Introduction

- One of the most astonishing (amazing) premises of Extreme Programming (XP):
  - You can eliminate requirements, design, and testing phases as well as the formal documents that go with them
- Its quite different from the way we typically learn to develop software: many people dismiss it as a fantasy
- The traditional approach is: “we need more requirements, design, and testing, not less!”
  - That’s true - software projects do need more requirements, design, and testing
  - That’s the reason why XP teams work on these activities every day. Yes, every day.

Introduction to XP – Contd.

- ✧ XP emphasizes (give special importance) face-to-face collaboration
  - This is so effective in eliminating communication delays and misunderstandings that the team no longer needs distinct phases.
  - This allows them to work on all activities every day - with simultaneous phases

Introduction to XP – Contd.



- Using simultaneous phases, an XP team produces deployable software every week
- In each iteration, the team analyzes, designs, codes, tests, and deploys a subset of features

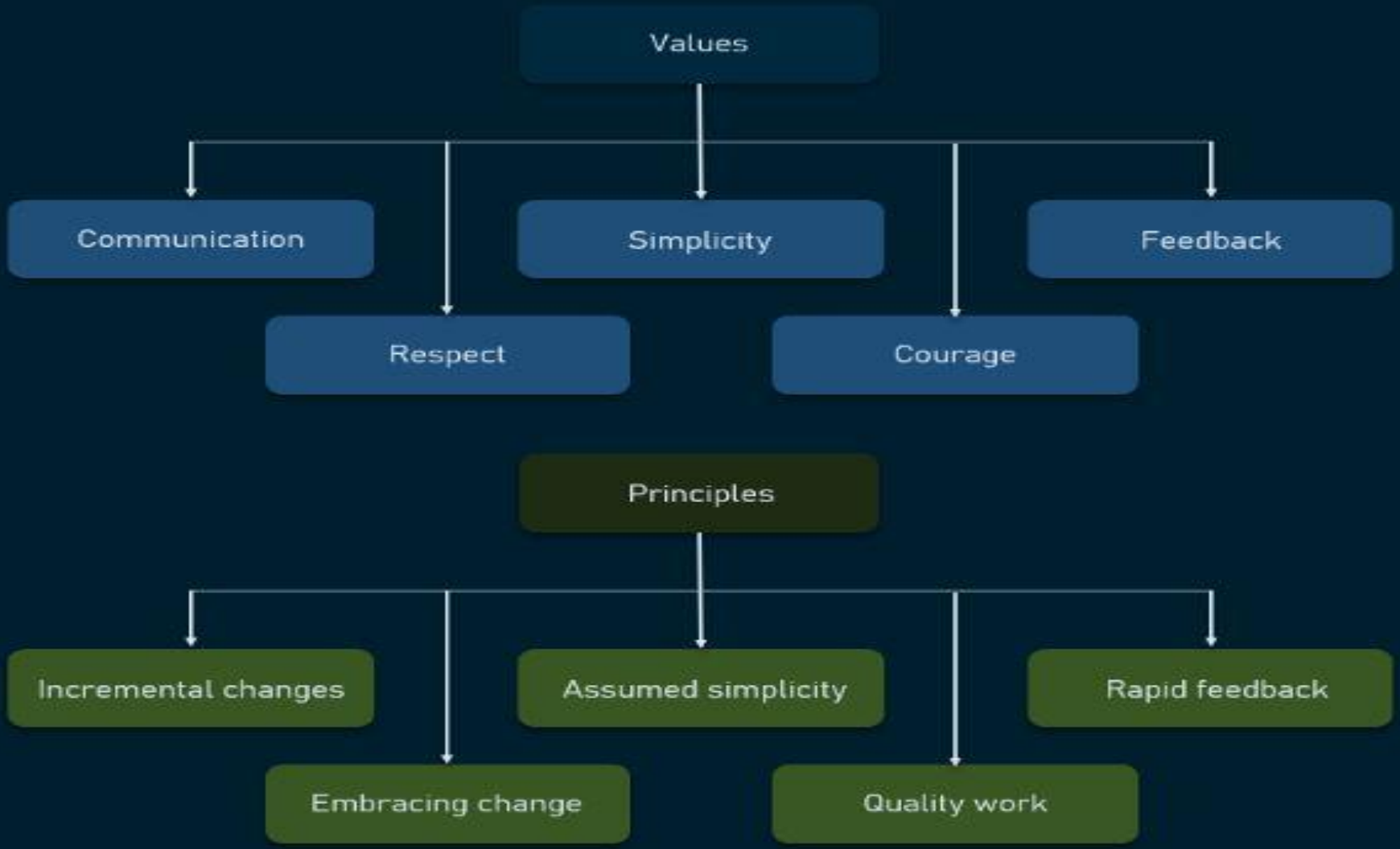
Introduction to XP – Contd.

- ✧ Analysis, design, coding, testing, and even deployment occur with rapid frequency - simultaneously
- ✧ XP does it by working in iterations:
  - Every week, the team does a bit of release planning, a bit of design, a bit of coding, a bit of testing, and so forth
  - They work on stories: very small features, or parts of features, that have customer value
  - Every week, the team commits to delivering four to ten stories
  - Throughout the week, they work on all phases of development for each story.
  - At the end of the week, they deploy their software for internal review

EXTREME PROGRAMMING PRACTICES

Group	Practices
Feedback	✓ Test-Driven Development ✓ The Planning Game ✓ On-site Customer ✓ Pair Programming
Continual Process	✓ Continuous Integration ✓ Code Refactoring ✓ Small Releases
Code understanding	✓ Simple Design ✓ Collective Code Ownership ✓ System Metaphor ✓ Coding Standards
Work conditions	✓ 40-Hour Week

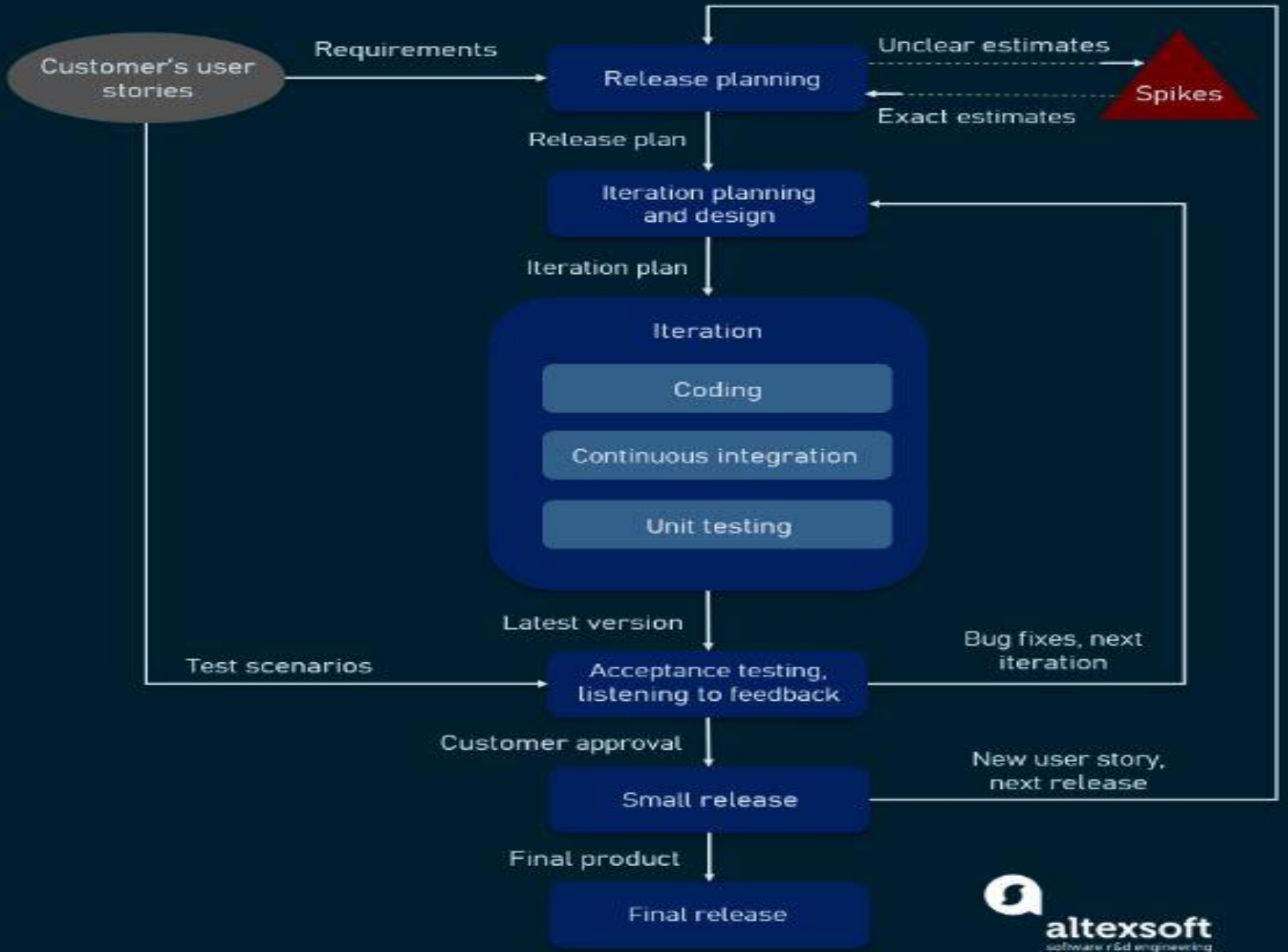
EXTREME PROGRAMMING VALUES AND PRINCIPLES



The XP Lifecycle

- ✧ Planning
- ✧ Analysis
- ✧ Design and coding
- ✧ Testing
- ✧ Deployment

EXTREME PROGRAMMING LIFECYCLE



XP Lifecycle: Planning

- Every XP team includes several business experts - the On-site Customers - who are responsible for making business decisions
- On-site customers point the project in the right direction by
  - Clarifying the project vision,
  - Creating stories,
  - Constructing a Release Plan, and
  - Managing Risks
- Programmers provide estimates and suggestions
  - These are blended with customer priorities in a process called the Planning Game

XP Lifecycle: Planning – Contd.

- On-site customers and Programmers
  - Together as a team, strives to create small, frequent releases that maximize value
- The planning effort is most intense during the first few weeks of the project
- During the remainder of the project, customers continue to review and improve the vision
- In addition to the overall release plan, the team creates a detailed plan for the upcoming week at the beginning of each iteration

XP Lifecycle: Analysis

- Rather than using an upfront analysis phase to define requirements, on-site customers sit with the team full-time.
- On-site customers:
  - May or may not be real customers depending on the type of project, but they are the people best qualified to determine what the software should do.
  - Are responsible for figuring out the requirements for the software.
  - Use their own knowledge as customers combined with traditional requirements gathering techniques

XP Lifecycle: Analysis – Contd.

- When programmers need information, they simply ask
- Customers are responsible for organizing the programmers' work so they are ready when they ask for information
- Customers figure out:
  - The general requirements for a story before the programmers estimate it
  - The detailed requirements for the story before the programmers implement it

XP Lifecycle: Analysis – Contd.

- Some requirements are tricky or difficult to understand
- Customers formalize these requirements, with the assistance of testers, by creating customer tests: detailed, automatically checked examples
- Customers and testers create the customer tests for a story around the same time that programmers implement the story

XP Lifecycle: Design and Coding

- XP uses incremental design and architecture to continuously create and improve the design in small steps.
- This work is driven by test-driven development (TDD), an activity that inextricably weaves together testing, coding, design, and architecture
- To support this process, programmers work in pairs
  - Increases the amount of brainpower brought to bear on each task
  - Ensures that one person in each pair always has time to think about larger design issues

XP Lifecycle: Design and Coding - Contd.

- Programmers
  - Are responsible for managing their development environment
  - Integrate their code every few hours
  - Ensure that every integration is technically capable of deployment
- To support this effort, programmers also maintain coding standards and share ownership of the code
- The team shares a joint aesthetic for the code
  - Everyone is expected to fix problems in the code regardless of who wrote it

XP Lifecycle: Testing

- XP includes a sophisticated suite of testing practices
- Each member of the team - programmers, customers, and testers - makes his own contribution to software quality.
- Well-functioning XP teams produce only a handful of bugs per month in completed work
- Programmers provide the first line of defense with Test-Driven Development (TDD)
- TDD produces automated unit and integration tests

XP Lifecycle: Testing – Contd.

- Customer tests help ensure that the programmers' intent matches customers' expectations
- Testers help the team understand whether their efforts are in fact producing high quality code.
  - Testers use exploratory testing to look for surprises and gaps in the software
- Testers also explore the software's non-functional characteristics, such as performance and stability

XP Lifecycle: Testing – Contd.

- The team doesn't perform any manual regression testing
- TDD and customer testing leads to a sophisticated suite of automated regression tests
- When bugs are found, programmers create automated tests to show that the bugs have been resolved

XP Lifecycle: Deployment

- XP teams keep their software ready to deploy at the end of any iteration
- They deploy the software to internal stakeholders every week in preparation for the weekly iteration demo
- Deployment to real customers is scheduled according to business needs

XP Lifecycle: Deployment – contd.

- As long as the team is active, it maintains the software it has released
- Depending on the organization, the team may also support its own software
- In other cases, a separate support team may take over
- Similarly, when the project ends, the team may hand off maintenance duties to another team
  - In this case, the team creates documentation and provides training as necessary during its last few weeks

The XP Team

- Team software development is different than working solo on a project
  - Information is spread out among many members of the team
  - Different people know:
    - How to design and program the software: Programmers, Designers, and Architects
    - Why the software is important: Product Manager
    - The rules the software should follow: Domain Experts
    - How the software should behave: Interaction Designers
    - How the user interface should look: Graphic Designers
    - Where defects are likely to hide: Testers
    - How to interact with the rest of the company: Project Manager
    - Where to improve work habits: Coach

The XP Team – contd.

- XP creates cross-functional teams composed of diverse people who can fulfill all the team's roles
- XP teams
  - Sit together in an open workspace
  - At the beginning of each iteration, the team meets for a series of activities:
    - an iteration demo,
    - a retrospective, and
    - iteration planning
  - These typically take two to four hours in total
  - Also meet for daily stand-up meetings, which usually take five to ten minutes each

The XP Team – contd.

- XP Teams
  - Other than the scheduled activities, everyone on the team plans his own work
  - That doesn't mean everybody works independently; they just aren't on an explicit schedule.
  - Team members work out the details of each meeting when they need to.
  - Sometimes it's as informal as somebody standing up and announcing across the shared workspace that he would like to discuss an issue
- This self-organization is a hallmark of agile teams

The XP Team – contd.

- On-site Customers
  - Product Manager
  - Domain Experts
  - Interaction Designers
  - Business Analysts
- Programmers
  - Designers and architects
  - Technical specialists
- Testers

The XP Team – contd.

- Coaches
  - Programmer-coach
  - Project Manager
- The above list is not comprehensive
- An XP team should include exactly the expertise necessary to complete the project successfully
  - Examples of other team Members
    - Technical Writer
    - ISO 9001 Analyst.

XP Concepts

- XP has its own vocabulary. Some of the most common are:
  - Refactoring
  - Technical Debt
  - Timeboxing
  - The Last Responsible Moment
  - Stories
  - Iterations
  - Velocity
  - Theory of Constraints
  - Mindfulness

Adopting XP

- Before adopting XP, we need to decide whether it's appropriate for our situation
- Often, people's default reaction to hearing about XP is to say,
  - *"Well, of course that works for other teams, but it couldn't possibly work for us."*
- XP's applicability is based on organizations and people, not types of projects

Is XP Right for Us?

- We can adopt XP in many different conditions, although the practices that is used shall vary depending on the situation
- We shall discuss some chosen practices to give the greatest chance of success
  - That leads to some prerequisites and recommendations about your team's environment

Prerequisites for Adopting XP

✧ Prerequisite #1: Management Support

- It is very difficult to use XP in the face of opposition from the management. Active support is best.
- If we want the management to support our adoption of XP, they need to believe in its benefits.
- If management refuses to support, then XP probably may not be the appropriate approach for your team

Prerequisites for Adopting XP

✧ Prerequisite #2: Team Agreement

- The team's agreement to use XP is equally important
- If team members don't want to use XP, it's not likely to work
- XP assumes good faith on the part of team members - there's no way to force the process on somebody who's resisting it.

Prerequisites for Adopting XP – contd.

- ✧ Prerequisite #3:
  - A Co-located Team
- ✧ Prerequisite #4:
  - On-Site Customers
- ✧ Prerequisite #5:
  - The Right Team Size
- ✧ Prerequisite #6:
  - Use All the Practices

Recommendations for Adopting XP

- ✧ Recommendation #1:
  - A Brand-New Codebase
- ✧ Recommendation #2:
  - Strong Design Skills
- ✧ Recommendation #3:
  - A Language That's Easy to Refactor
- ✧ Recommendation #4:
  - An Experienced Programmer-Coach
- ✧ Recommendation #5:
  - A Friendly and Cohesive Team

Advantages of XP:

Slipped schedules – Timely delivery is ensured through slipping timetables and doable development cycles.

Misunderstanding the business and/or domain – Constant contact and explanations are ensured by including the client on the team.

Canceled projects – Focusing on ongoing customer engagement guarantees open communication with the consumer and prompt problem-solving.

Staff turnover – Teamwork that is focused on cooperation provides excitement and goodwill. Team spirit is fostered by multidisciplinary cohesion.

Costs incurred in changes – Extensive and continuing testing ensures that the modifications do not impair the functioning of the system. A functioning system always guarantees that there is enough time to accommodate changes without impairing ongoing operations.

Business changes – Changes are accepted at any moment since they are seen to be inevitable.

Production and post-delivery defects: Emphasis is on – the unit tests to find and repair bugs as soon as possible.