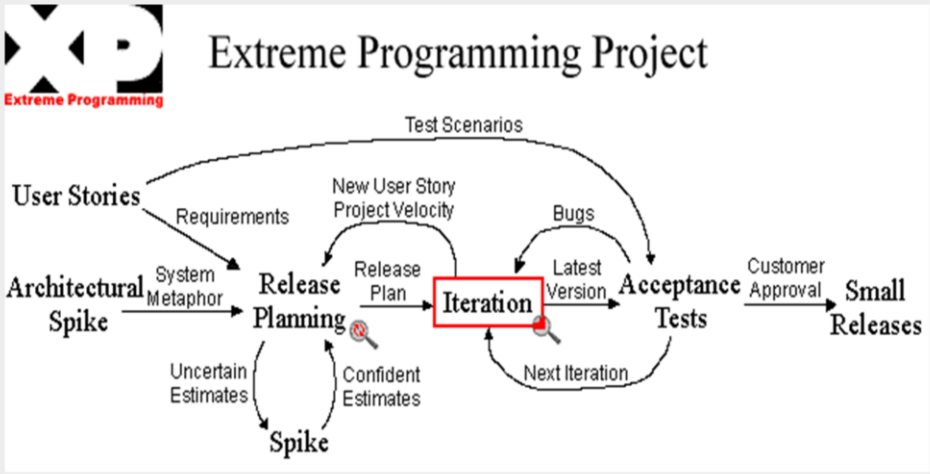

Introduction to Agile
Lesson 3: Agile Methods and Practices - Extreme Programming (XP),Lean Software Development & Kanban
Capgemini

Lesson Objectives

- Introduction to Extreme Programming
- The Rules of Extreme Programming
- Extreme Programming (XP) - Principles
- Extreme Programming (XP) – Key Terms
- Introduction to Lean Software Development
- Principles of Lean Software Development
- What is Kanban?

3.1: Agile Methods and Practices – Extreme Programming (XP)

Introduction to Extreme Programming

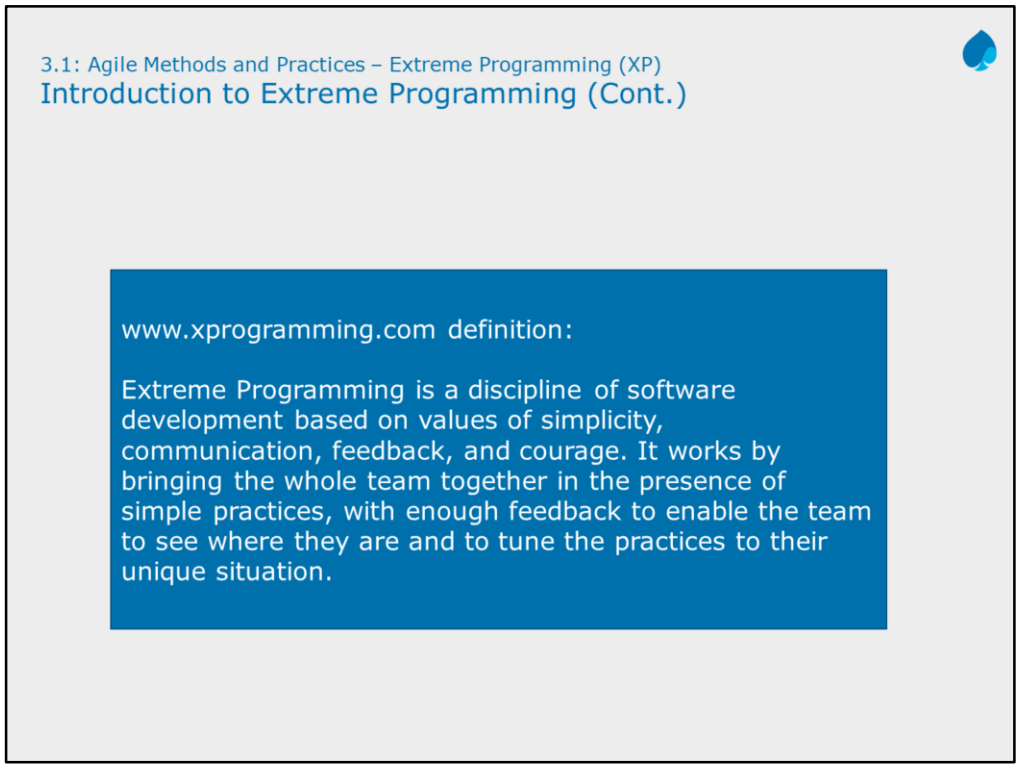


3.1: Agile Methods and Practices – Extreme Programming (XP)

Introduction to Extreme Programming (Cont.)

- Extreme programming (XP) was created by Kent Beck where he applied a light weight methodology to deliver a financial system in 2 years which previously had been undelivered over a number of years with a team of 30 people

> Wikipedia definition:
>
> Extreme Programming (XP) is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type of agile software development, it advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints where new customer requirements can be adopted.


3.1: Agile Methods and Practices – Extreme Programming (XP)
Introduction to Extreme Programming (Cont.)
www.xprogramming.com definition:
Extreme Programming is a discipline of software development based on values of simplicity, communication, feedback, and courage. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation.

3.1: Agile Methods and Practices – Extreme Programming (XP)

The Rules of Extreme Programming

- Planning
  - User Stories User stories are written
  - Release planning creates the release schedule
  - Make frequent small releases
  - The project is divided into iterations
  - Iteration planning starts each iteration
- Managing
  - Give the team a dedicated open work space
  - Set a sustainable pace
  - A stand up meeting starts each day
  - The Project Velocity is measured
  - Move people around
  - Fix XP when it breaks

3.1: Agile Methods and Practices – Extreme Programming (XP)

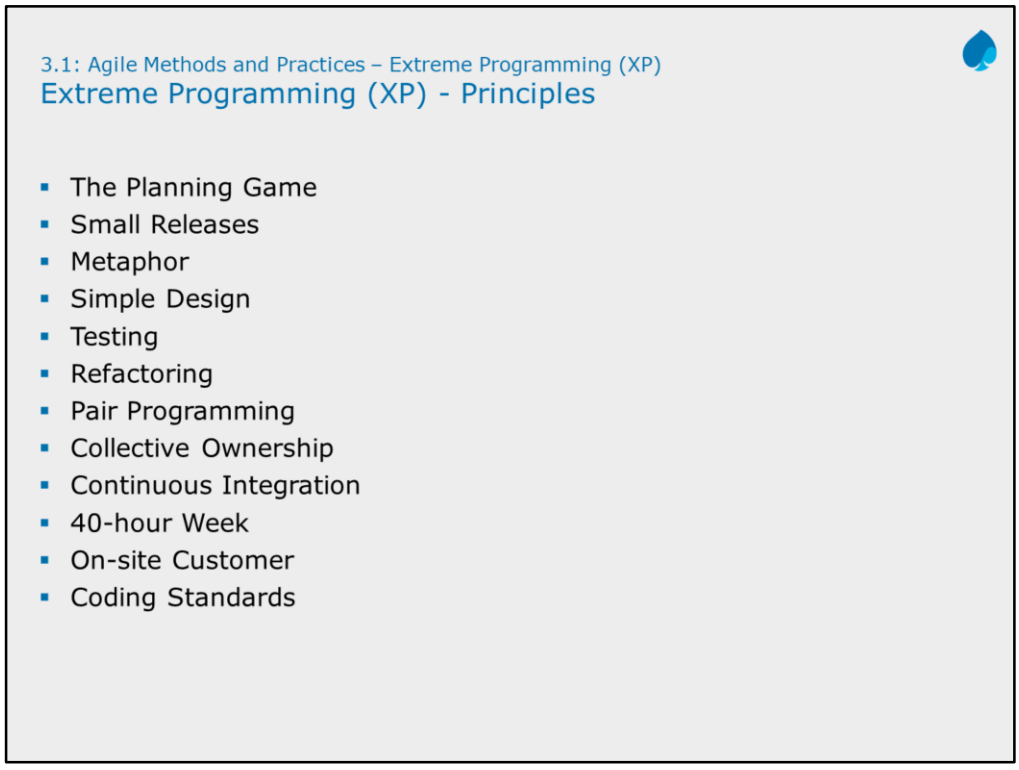
The Rules of Extreme Programming (Cont.)

- Coding
  - The customer is always available
  - Code must be written to agreed standards
  - Code the unit test first
  - All production code is pair programmed
  - Only one pair integrates code at a time
  - Set up a dedicated integration computer
- Designing
  - Simplicity
  - Choose a system metaphor
  - Use CRC cards for design sessions
  - Create spike solutions to reduce risk
  - No functionality is added early
  - Refactor whenever and wherever possible

3.1: Agile Methods and Practices – Extreme Programming (XP)

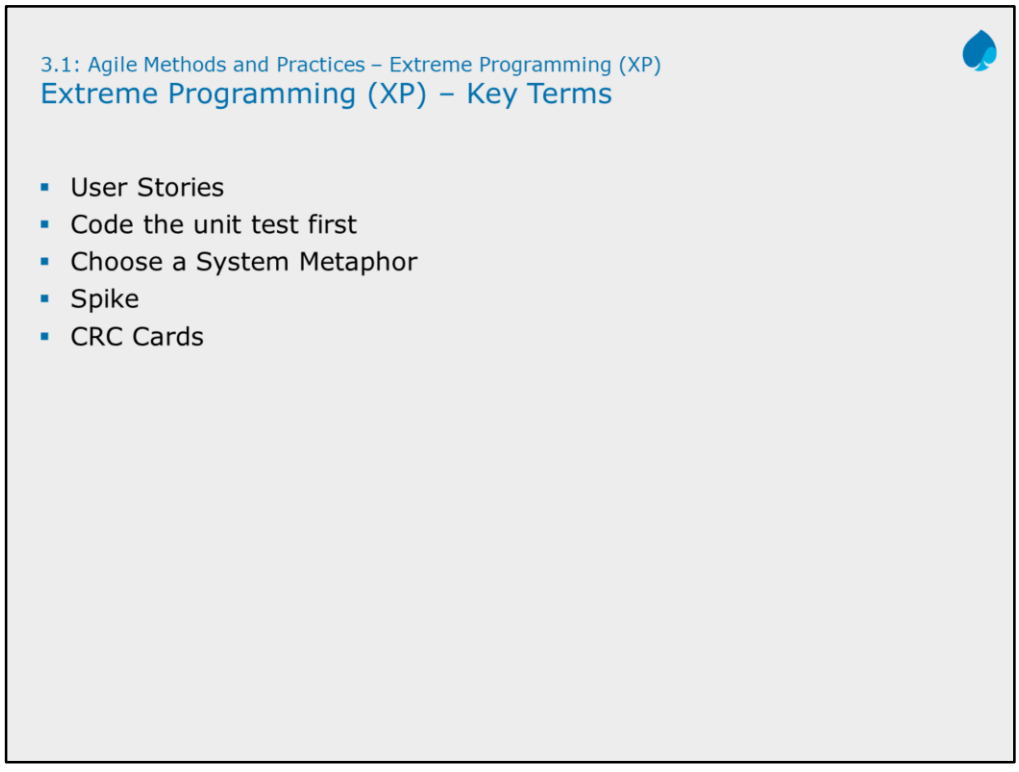
The Rules of Extreme Programming (Cont.)

- Testing
  - All code must have unit tests
  - All code must pass all unit tests before it can be released
  - When a bug is found tests are created.
  - Acceptance tests are run often and the score is published



Extreme Programming (XP) – Principles:

1. The Planning Game - Scope next release with business priorities, technical, estimates Update the plan based on reality.
2. Small Releases - Put simply system into production quickly Release new version in short cycle.
3. Metaphor - Guide development with simple shared story of how the whole system works.
4. Simple Design - Design as simple as possible at any given moment.
5. Testing - Continually write and run unit tests.
6. Refactoring - Restructure system without changing its behavior to remove duplication, improve communication, add flexibility and simplify.
7. Pair Programming - Two programmers, one machine, four eyes are better than two.
8. Collective Ownership - Anyone can change code anywhere in the system at any time.
9. Continuous Integration - Integrate and build the system many times a day, every time a talk is completed.
10. 40-hour Week - Never work overtime a second week in a row.
11. On-site Customer - Real, live user on the team, available full-time to answer questions.
12. Coding Standards - All code written accordance with rules emphasizing communication through the code.



Extreme Programming (XP) – Key Terms

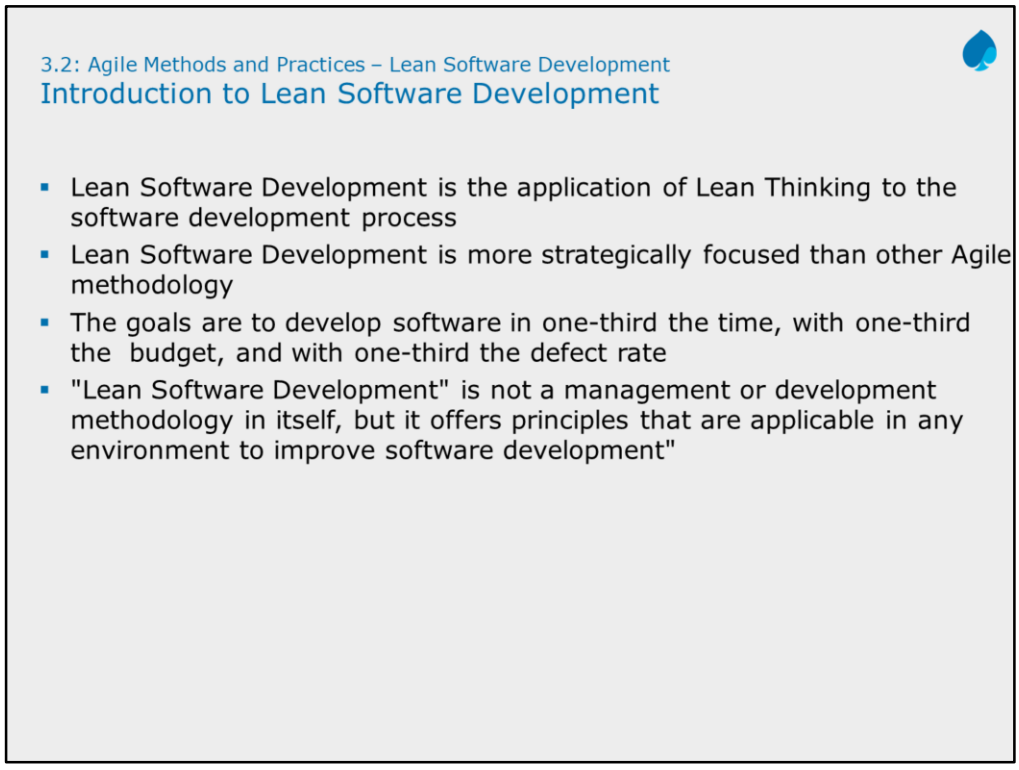
User Stories - User stories serve the same purpose as use cases but are not the same. They are used to create time estimates for the release planning meeting. They are also used instead of a large requirements document. User Stories are written by the customers as things that the system needs to do for them. They are similar to usage scenarios, except that they are not limited to describing a user interface. They are in the format of about three sentences of text written by the customer in the customers terminology without techno-syntax.

Code the unit test first –Test Driven Development (TDD) is a development technique that switches the coding approach from "code & test" to "test & code". By designing a test first you flush out design and testability issues earlier. Once you have your test, you write just enough code to make the test pass. You continue to iteratively and incrementally add new tests and new code to make the test pass.

Choose a System Metaphor - This is a story that everyone can tell to explain how the system works through the use of common and understandable terminology.

Spike - Create spike solutions to figure out answers to tough technical or design problems. A spike solution is a very simple program to explore potential solutions. Build the spike to only addresses the problem under examination and ignore all other concerns. Most spikes are not good enough to keep, so expect to throw it away. The goal is reducing the risk of a technical problem or increase the reliability of a user story's estimate. When a technical difficulty threatens to hold up the system's development put a pair of developers on the problem for a week or two and reduce the potential risk.

CRC Cards –Class, Responsibility and Collaboration cards are a design technique that uses cards to represent objects. You use a workshop based approach to then walk through how the objects interact with each other i.e. the messages that they send to other objects.

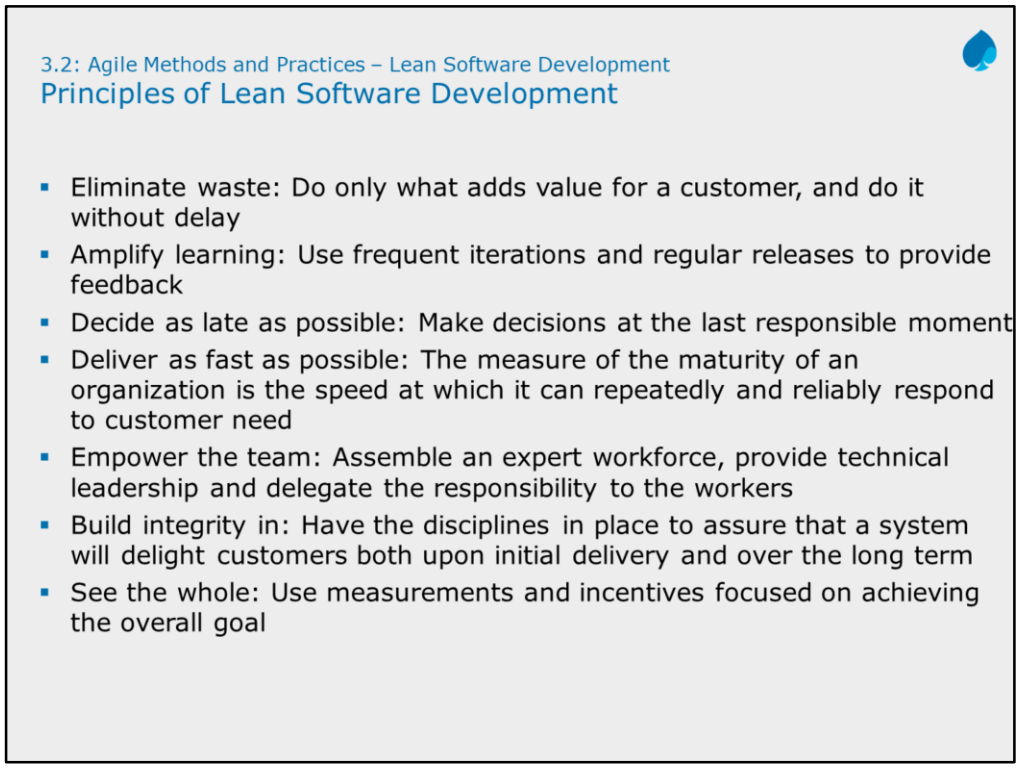


What is Lean Software Development?

The concepts of a lean-enterprise originate from Toyota in Japan after the Second World War. Lean software development is a management philosophy that focuses on throughput. Lean software development doesn't focus on particular components of the value-stream like code-construction or QA, but on whether the components of the chain are working as efficiently as possible so as to generate as much value as possible to the customer.

Lean software development practices were created as a result of recognizing that enterprises were reacting to complex software development and resource challenges by adding:

- **More measurements**
- **More controls**
- **More checks and balances**
- **More process rigor**



Principles of Lean Software Development

Eliminate waste - In software development, waste is anything that does not improve the quality of code, reduces the amount of time and effort it takes to produce code, or does not deliver business value to the customer. In other words, any activity that does not “pay for itself” in reduced effort elsewhere in the system.

Amplify Learning - For programmers to develop a system that delivers business value, they will have to learn about many things. Some are technical, such as the advantages and disadvantages to various approaches to do remote communications in .NET (i.e., remoting, COM+,web services, etc.). Others are requirements related, such as understanding what the business user really needs versus what the developer thinks the user needs.

Decide as late as possible - The idea here is to wait until what the authors term “the last responsible moment” to make a decision. This is the moment at which, if the team does not make a decision, the decision will be made for them (doing nothing is a choice). The benefits of this are avoiding or delaying the costs of change, which obviously cannot be incurred if you have not limited your options yet.

Deliver as fast as possible - This is the foundation of iterative development. Requirements change as a percentage of the original requirements increases non-linearly as the amount of time increases. Typical 9-12 month projects generate roughly a 25 percent change in requirements. However, the amount of requirements change over a month averages only 1-2 percent. And it is much easier to get users to accept waiting until next month rather than next year.

3.2: Agile Methods and Practices – Kanban

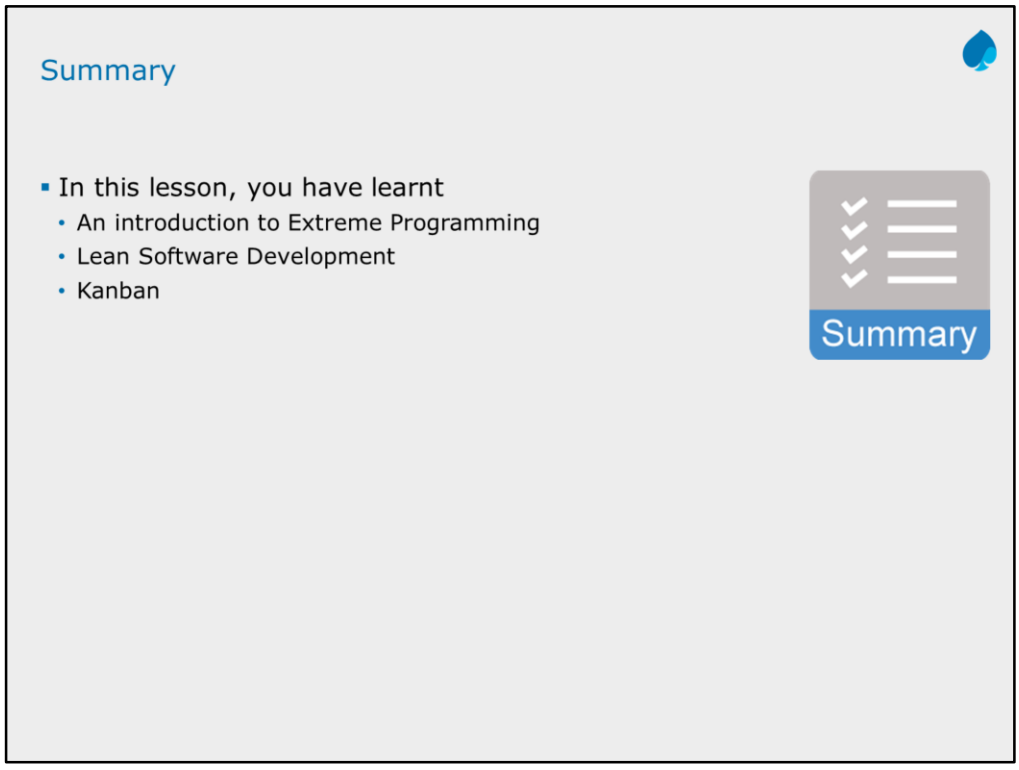
What is Kanban?

- The word Kan means "visual" in Japanese and the word "ban" means "card". So Kanban refers to "visual cards"

3.2: Agile Methods and Practices – Kanban

What is Kanban? (Cont.)

- Kanban is way for teams and organizations to visualize their work, identify and eliminate bottlenecks and achieve dramatic operational improvements in terms of throughput and quality
- Kanban is a method to gradually improve whatever you do – whether software development, IT/ Ops, Staffing, Recruitment, Marketing and Sales
- in fact, almost any business function can benefit from applying Kanban to bring about significant benefits such as reduced lead time, increased throughput and much higher quality of products or services delivered



Add the notes here.