Agile Software Development

Introduction

(Unit 1)

Dr. Sreedhar Bhukya

Professor, Department of CSE, SNIST

Introduction - Rapid Software Development

- Rapid development and delivery is now often the most important requirement for software systems
  - Businesses operate in a fast changing requirement, and it is practically impossible to produce a set of stable software requirements
  - Software has to evolve quickly to reflect changing business needs
- Plan-driven development is essential for some types of system but does not meet these business needs
- Agile development methods emerged in the late 1990s
  - The aim was to radically reduce the delivery time for working software systems

What is Agile Development

- Program specification, design and implementation are inter-leaved
- The system is developed as a series of versions or increments with stakeholders involved in version specification and evaluation
- Frequent delivery of new versions for evaluation
- Extensive tool support (e.g. automated testing tools) used to support development.
- Minimal documentation - focus on working code

Why Agile?

- ✧ Agile development isn't a silver bullet
- ✧ Adopting agile development is not recommended to solely to increase productivity
- ✧ Its benefits - even the ability to release software more frequently - come from working differently, not from working faster
- ✧ When you consider using agile development, only one question matters
  - Will agile development help us be more successful?
  - When you can answer that question, you'll know whether agile development is right for you

Understanding Success

- The traditional idea of success is delivery on time, on budget, and according to specification
  - Successful
    - Completed on time, on budget, with all features and functions as originally specified
  - Challenged
    - Completed and operational but over budget, over the time estimate, (with) fewer features and functions than originally specified."
  - Impaired (Damaged)
    - Cancelled at some point during the development cycle.
- Despite their popularity, there's something wrong with these definitions

Understanding Success

- A project can be successful even if it never makes a dime (small money)
- It can be challenged even if it delivers millions of dollars in revenue
- Success means delivering value to the organization

Understanding Success

✧ All these three types of success are important

Understanding Success

- ✧ Without personal success,
  - You'll have trouble motivating yourself and employees
- ✧ Without technical success,
  - Your source code will eventually collapse under its own weight
- ✧ Without organizational success,
  - Your team may find that they're no longer wanted in the company

Personal Success

- Every one, from senior management to managers to programmers, will appreciate the agile methodology for achieving success

  - Executives and senior management
  - Users, stakeholders, domain experts, and product managers
  - Project and product managers
  - Developers
  - Testers

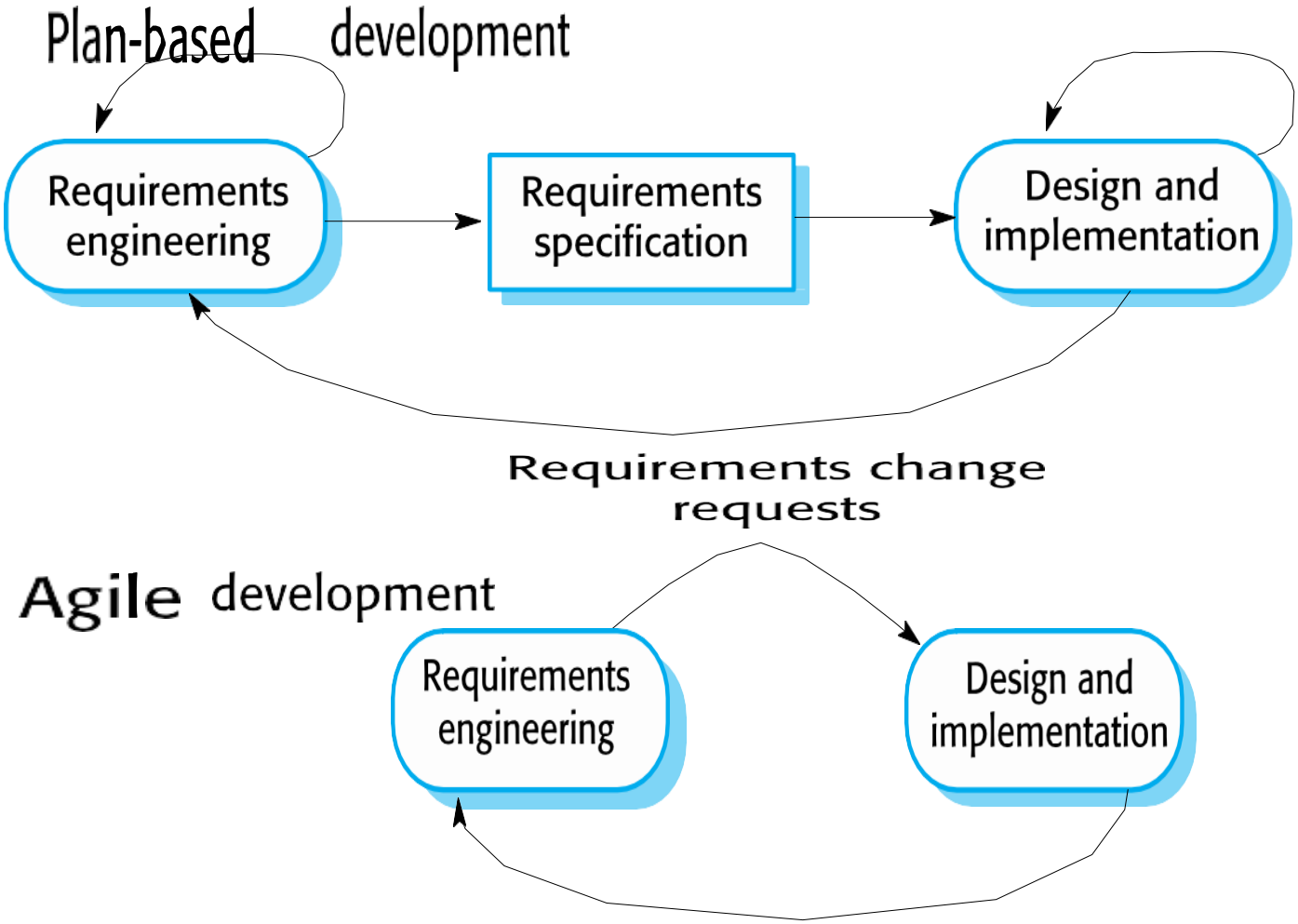
Technical Success

- In Extreme Programming (an agile method),
  - Programmers work together, which ensures that at least two people review every piece of code
  - Programmers continuously integrate their code, which enables the team to release the software whenever it makes business sense
  - The whole team focuses on finishing each feature completely before starting the next
  - Includes advanced technical practices: Test-driven development
  - Teams also create simple, ever-evolving designs that are easy to modify when plans change

Organizational Success

- ✧ Agile methods achieve organizational successes by focusing on delivering value and decreasing costs
- ✧ Agile teams increase value by
  - Including business experts and
  - Focusing development efforts on the core value that the project provides for the organization
- ✧ Agile teams decrease costs as well
  - They do this partly by technical excellence; the best agile projects generate only a few bugs per month
  - They also eliminate waste by cancelling bad projects early and replacing expensive development practices with simpler ones

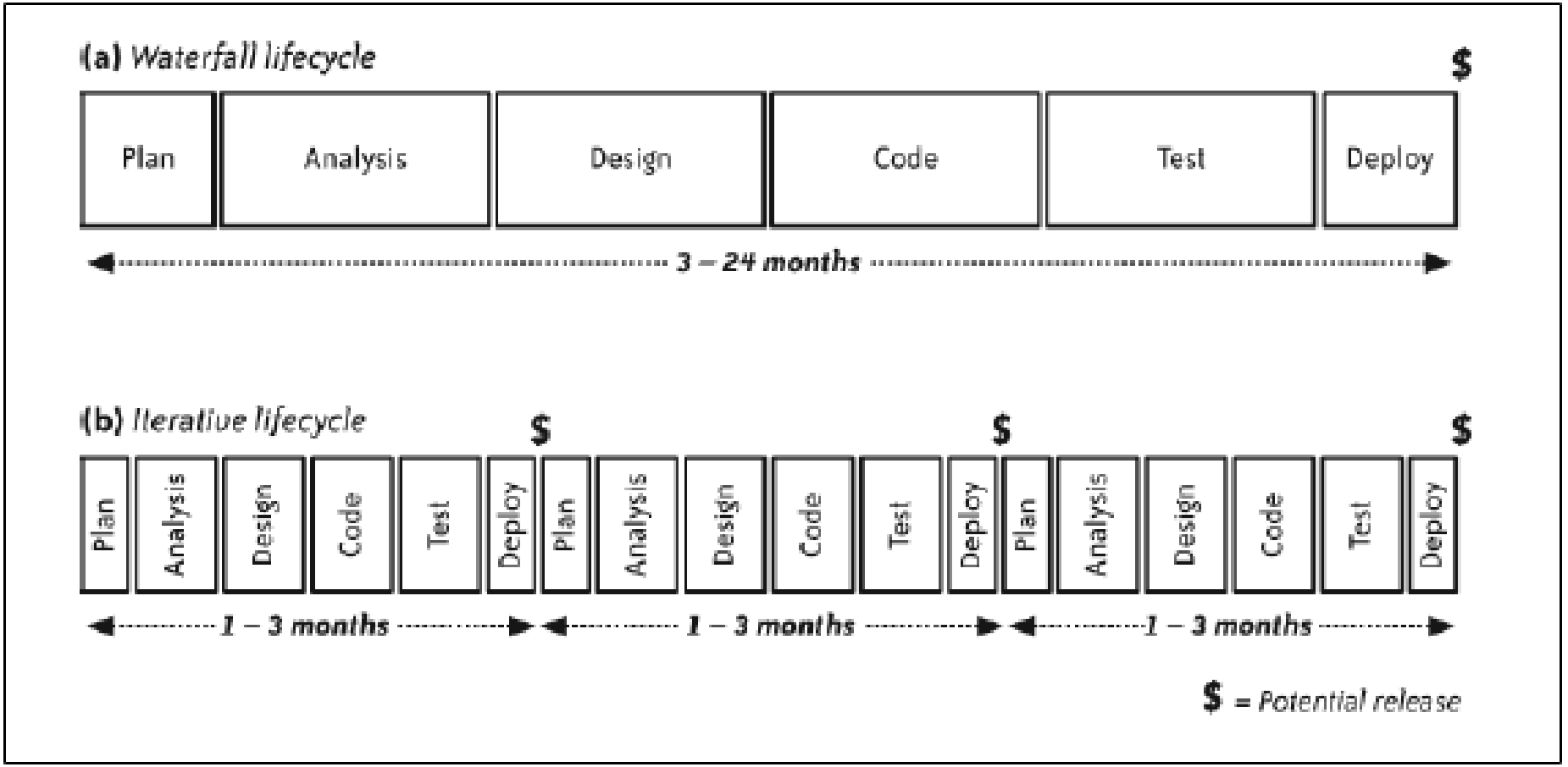
Plan-driven Vs Agile development



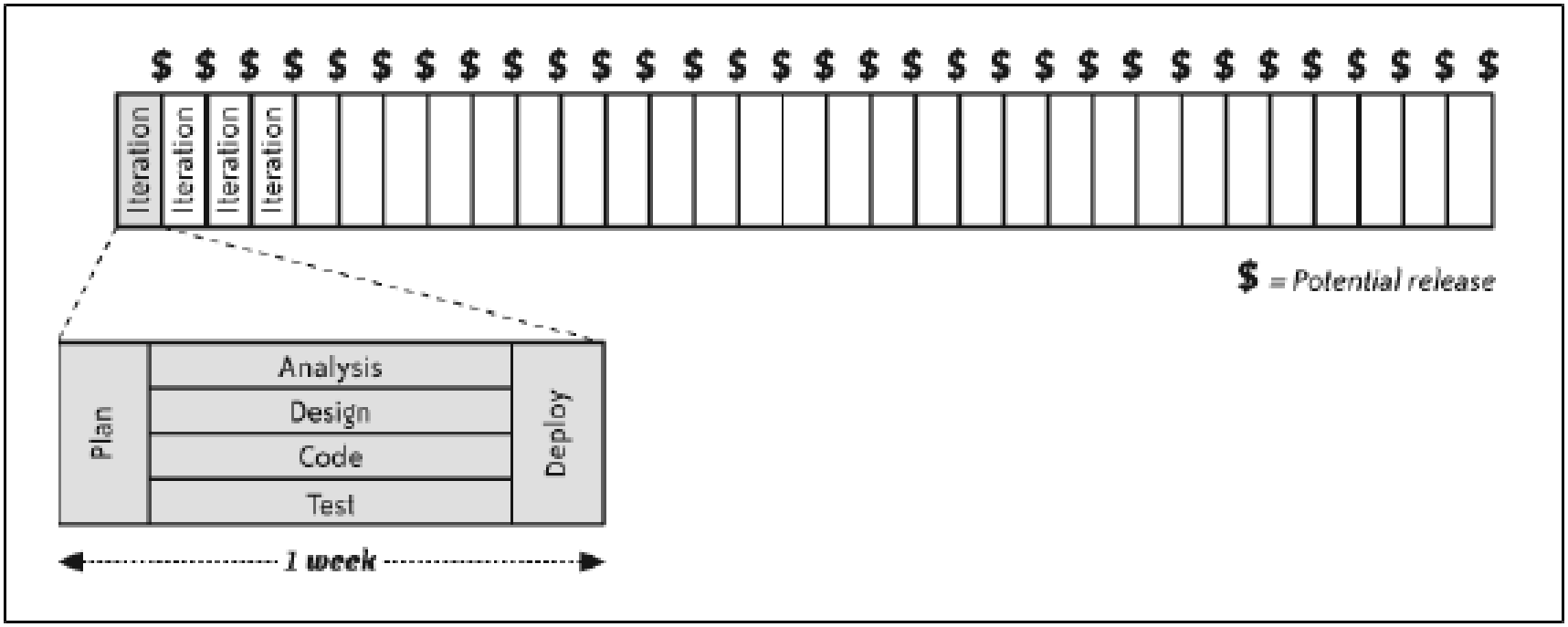
Plan-driven Vs Agile development – Contd.

- Plan-driven development
  - A plan-driven approach to software engineering is based around separate development stages with the outputs to be produced at each of these stages planned in advance
  - Not necessarily waterfall model – plan-driven, incremental development is possible
  - Iteration occurs within activities
- Agile development
  - Specification, design, implementation and testing are inter-leaved and the outputs from the development process are decided through a process of negotiation during the software development process.

Plan-driven Vs Agile development – Contd.



Plan-driven Vs Agile development – Contd.



Agile methods

- Dissatisfaction with the overheads involved in software design methods of the 1980s and 1990s led to the creation of agile methods.
- Agile development is a philosophy; It's a way of thinking about software development
- Agile methods are processes that support the agile philosophy.
  - Examples include Extreme Programming and Scrum.

Agile methods – Contd.

- Agile methods consist of individual elements called practices
  - Agile methods combine them in unique ways, accentuating those parts that support the agile philosophy, discarding the rest, and mixing in a few new ideas

Agile methods – Contd.

- Agile methods:
  - Focus on the code rather than the design
  - Are based on an iterative approach to software development
  - Are intended to deliver working software quickly and evolve this quickly to meet changing requirements
- The aim of agile methods is
  - To reduce overheads in the software process (e.g. by limiting documentation) and
  - To be able to respond quickly to changing requirements without excessive rework

Agile Manifesto

- The description of the agile philosophy is the Agile Manifesto, a collection of 4 values and 12 principles
- We are uncovering better ways of developing software by doing it and helping others do it.
- Through this work we have come to value:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan
- That is, while there is value in the items on the right, we value the items on the left more.

Agile Principles

Principles	Description
Customer Satisfaction	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
Changing Requirements	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
Deliver Frequently	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale
Customer Collaboration	Business people and developers must work together daily throughout the project
Trust	Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done
Sitting Together	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation

Agile Principles

Principles	Description
Working Software	Working software is the primary measure of progress
Sustainable Development	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
Technical Excellence	Continuous attention to technical excellence and good design enhances agility
Simplicity	Simplicity - the art of maximizing the amount of work not done - is essential
Self-Organizing Teams	The best architectures, requirements, and designs come from self-organizing teams
Adaptive Planning	At regular intervals, the team reflects on how to become more effective, then tunes and adjust its behavior accordingly

The Road to Mastery

- Mastering the art of agile development requires real-world experience using a specific, well-defined agile method
- Extreme Programming (XP) suits this purpose very well. It has several advantages:
  - XP is most complete when compared to other agile methods
  - XP places a strong emphasis on technical practices in addition to the more common teamwork and structural practices.
  - XP has undergone intense scrutiny
  - Well documented in the literature
  - Its capabilities and limitations are very well understood

The Road to Mastery – Contd.

- To master the art of agile development, follow these steps:
  - Decide why you want to use agile development. Will it make your team and organization more successful? How?
  - Adopt as many of XP's practices as you can
  - Follow the XP practices rigorously and consistently; If a practice doesn't work, try to follow the proposed approach more closely
    - Teams new to XP often under apply its practices
    - Expect to take 2 or 3 months to start feeling comfortable with the practices; another 2 to 6 months for them to become second nature
  - As you become confident that you are practicing XP correctly, give it several months
    - Start experimenting - Each time you make a change, observe what happens and make further improvements.

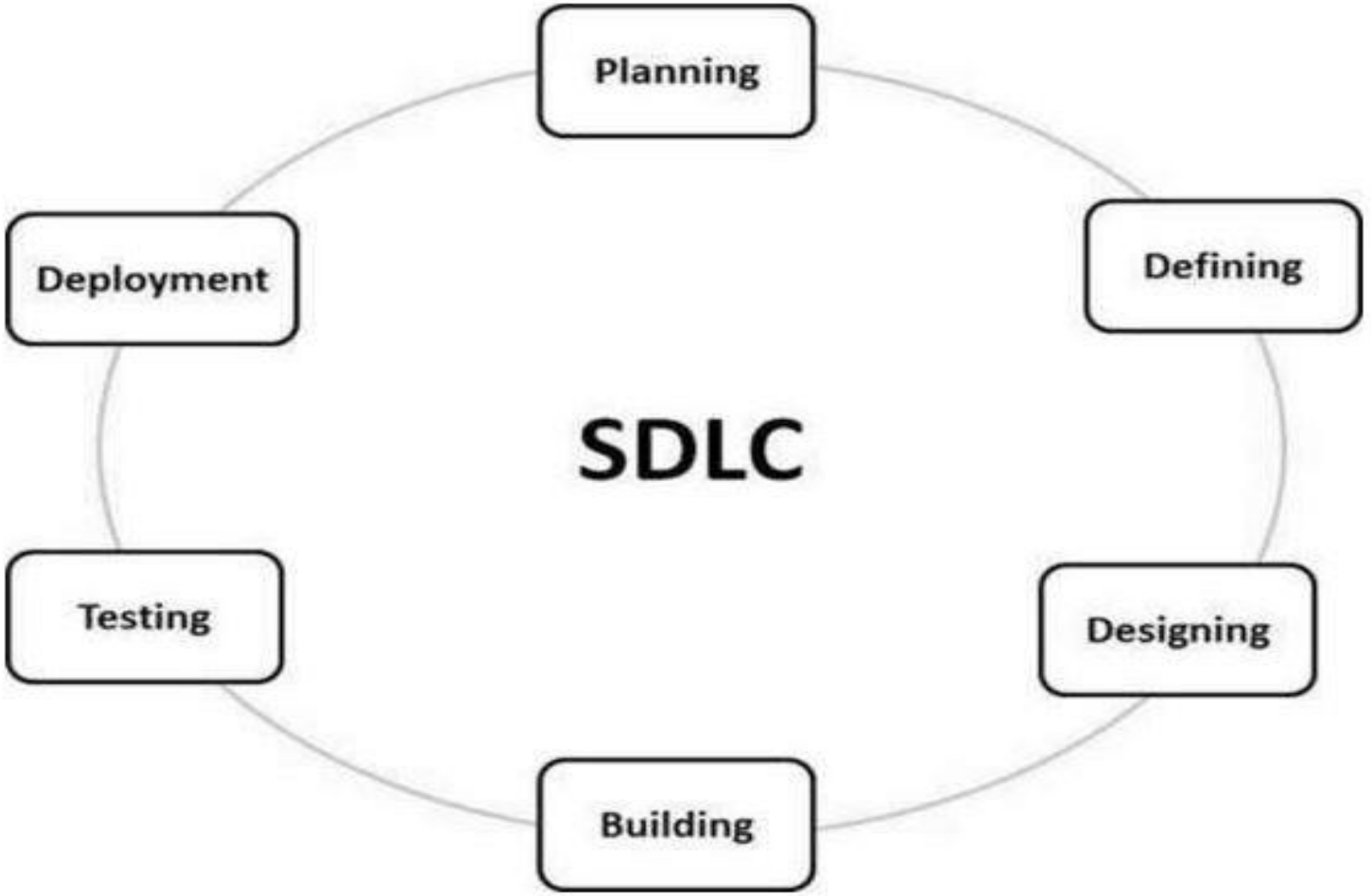
Find a Mentor

- As you adapt XP to your situation, you're likely to run into problems and challenges
- For these situations, you need a mentor: an outside expert who has mastered the art of agile development
- The hardest part of finding a mentor is finding someone with enough experience in agile development

Find a Mentor

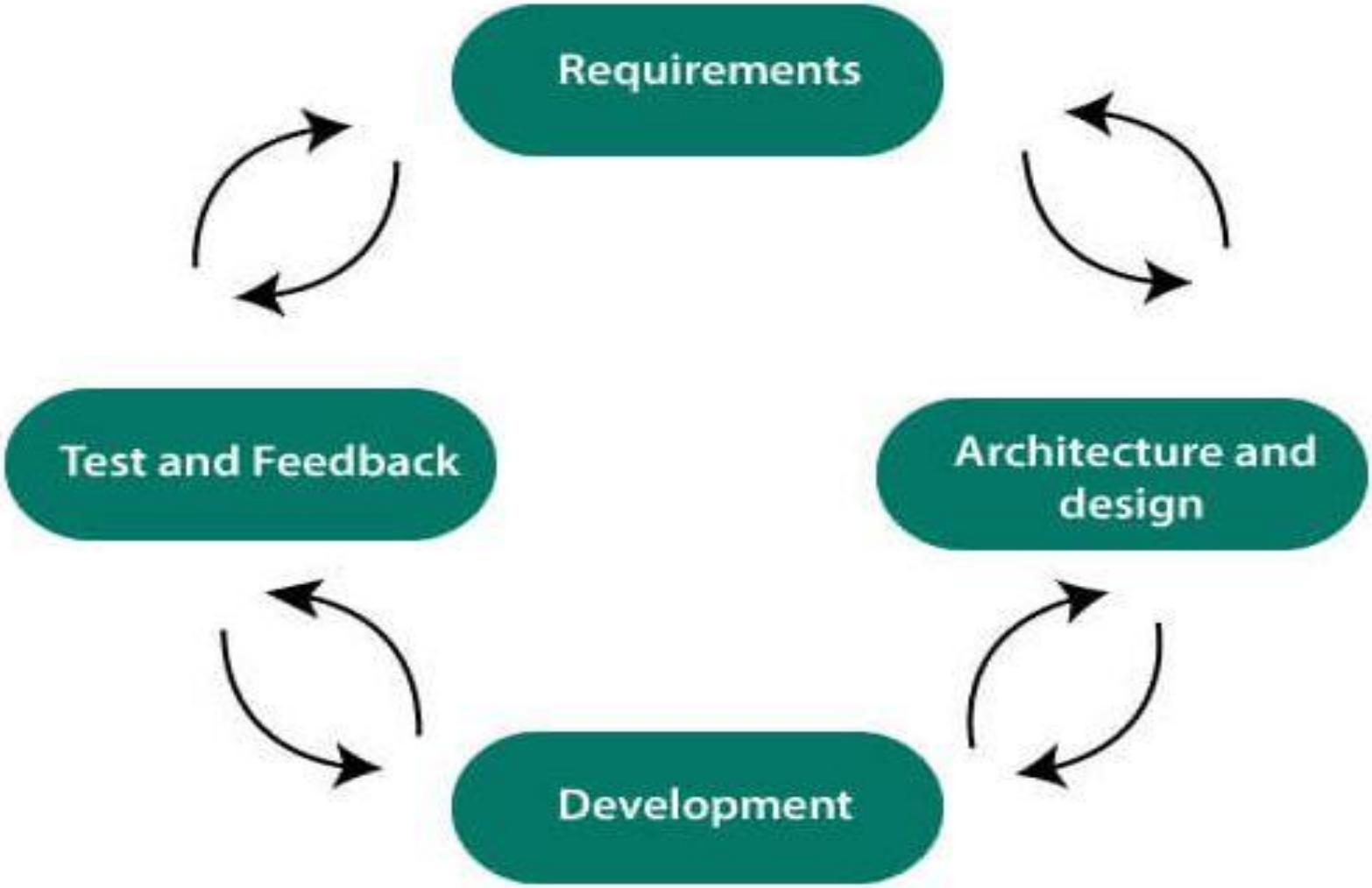
- Sources to try for finding a mentor include:
  - Other groups practicing XP in your organization
  - Other companies practicing XP in your area
  - A local XP/Agile user group
  - XP/Agile consultants
  - The XP mailing list: extremeprogramming@yahoogroups.com
- Asking a mentor for help means that the correct solution depends on the details of your situation
  - A mentor can help us troubleshoot the problem and offer situation-specific advice.

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares.



A typical Software Development Life Cycle consists of the following stages –

Agile Software Development Life Cycle (SDLC) is the combination of both iterative and incremental process models. It focuses on process adaptability and customer satisfaction by rapid delivery of working software product.



Advantages of Agile SDLC

- Project is divided into short and transparent iterations.
- It has a flexible change process.
- It minimizes the risk of software development.
- Quick release of the first product version.
- The correctness of functional requirement is implemented into the development process.
- Customer can see the result and understand whether he/she is satisfied with it or not.


Enhances collaboration between teams that don't usually work together
54%
Increases the level of software quality in organizations
52%
Results in increased customer satisfaction
49%
Shortens time to market
43%
Reduces cost of development
42%


T
E
A
M

Thank you