Agile Principles

Dr. Elham Mahmoudzadeh

Isfahan University of Technology

mahmoudzadeh@iut.ac.ir

2023

What is Software Engineering?

IEEE Computer Society Definition:

- "Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software."

Software engineering(2)

- Doing the right thing
  - *Software that users want and need*
  - *Software that benefits society*
- Doing the thing right
  - *Following a good software process*
  - *Developing your programming skills*

Who is a software engineer?

- *Software engineers are the creative minds behind computers or programs.*

- A software engineer is the one who follows
  - *A systematic process that leads to understanding the requirements,*
  - *Working with teams and various professionals*
  - *Design and create the application software or components or modules*
  - *Fulfill the specific needs of the users successfully;*

Software Development Methodology (SDM)

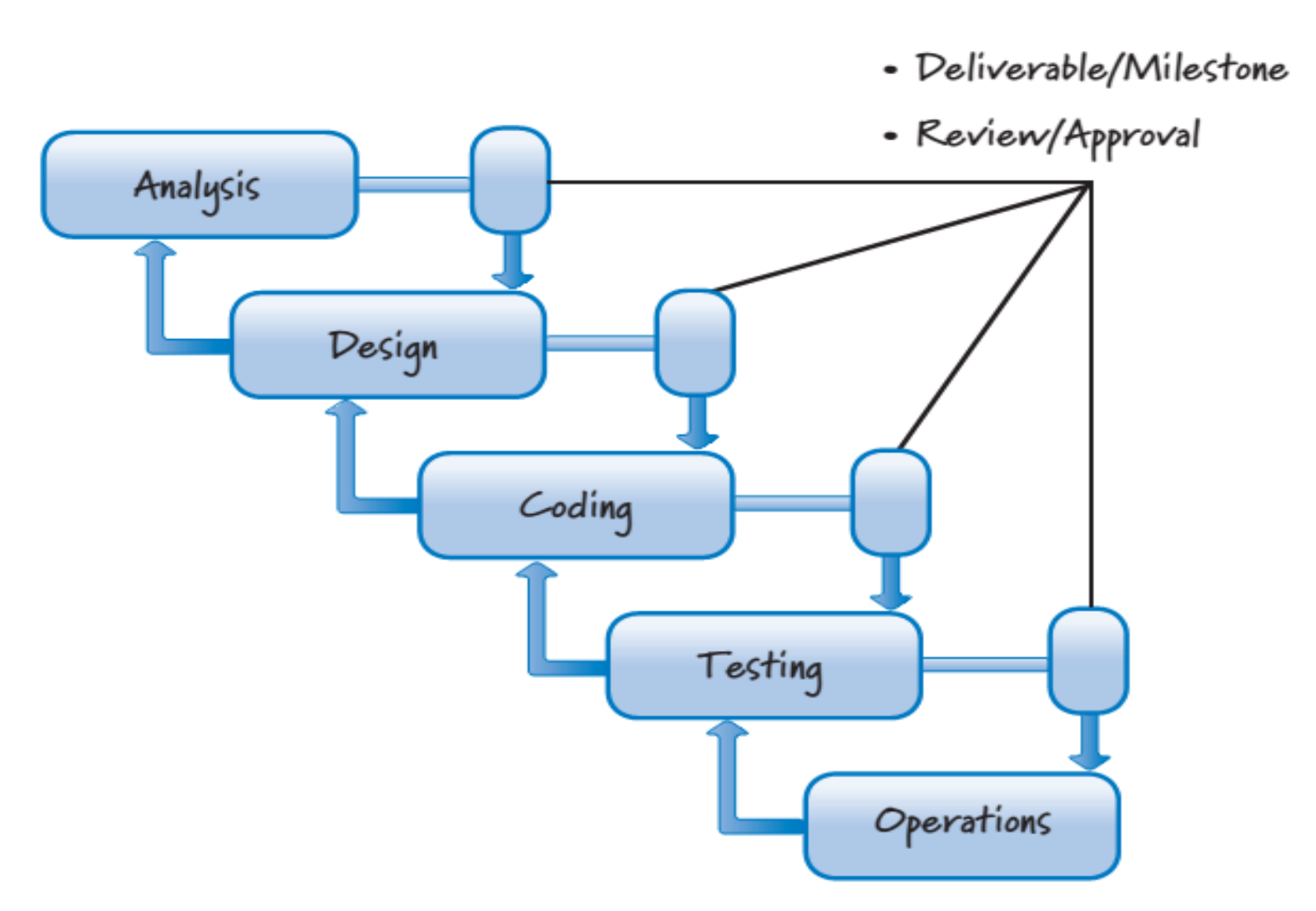
- A framework for applying software engineering practices with the specific aim of providing the necessary means for developing software-intensive systems.
- Have two parts.
  1. *A set of modeling conventions comprising a Modeling Language (syntax and semantics)*
  2. *A Process, which*
     - provides guidance as to the order of the activities,
     - specifies what artifacts should be developed using the Modeling Language,
     - directs the tasks of individual developers and the team as a whole,
     - offers criteria for monitoring and measuring a project's products and activities.

Agile vs. Traditional development process

- The goal of comparing agile principles with traditional development principles is not to make the case that **plan-driven**, **sequential development** is bad and that Scrum is good.
- Both are tools in the professional developer's toolkit; there is no such thing as a bad tool, rather just inappropriate times to use that tool.

Scrum and traditional, plan-driven, sequential development are appropriate to use on different classes of problems.

Plan-driven process (Waterfall)



Plan-driven process (I)

- Plan for and anticipate up front all of the features a user might want in the end product, and to determine how best to build those features.
- The idea here is that the better the planning, the better the understanding, and therefore the better the execution.
- Also called sequential processes because practitioners perform, in sequence, a complete requirements analysis followed by a complete design followed in turn by coding/building and then testing.

Plan-driven process (II)

- Works well if you are applying it to problems that are well defined, predictable, and unlikely to undergo any significant change.
- The problem is that most product development efforts are anything but predictable, especially at the beginning.
- So, while a plan-driven process gives the impression of an orderly, accountable, and measurable approach, that impression can lead to a false sense of security.

After all, developing a product rarely goes as planned.

Plan-driven process (III)

- Understand it, design it, code it, test it, and deploy it, all according to a well-defined, prescribed plan.
- There is a belief that it should work. If applying a plan-driven approach doesn't work, the prevailing attitude is that we must have done something wrong.
- Sure that if they just do it better, their results will improve. The problem, however, is not with the execution.
- It's that plan-driven approaches are based on a set of beliefs that do not match the uncertainty inherent in most product development efforts.

Traditional Pros.

- It is supremely logical.
- Think before you build.
- Write it all down.
- Follow a plan.
- Keep everything as organized as possible.

What's Wrong With Traditional Software Development?

Humans are involved.

- Creativity is inhibited.
- Written documents have their limitations.
- Bad timing.
- No crystal balls.
- Too much work and no fun.
- Sub-optimized results.

Manifesto for Agile Software Development

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.

Principles Behind the Agile Manifesto(I)

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.

Principles Behind the Agile Manifesto(II)

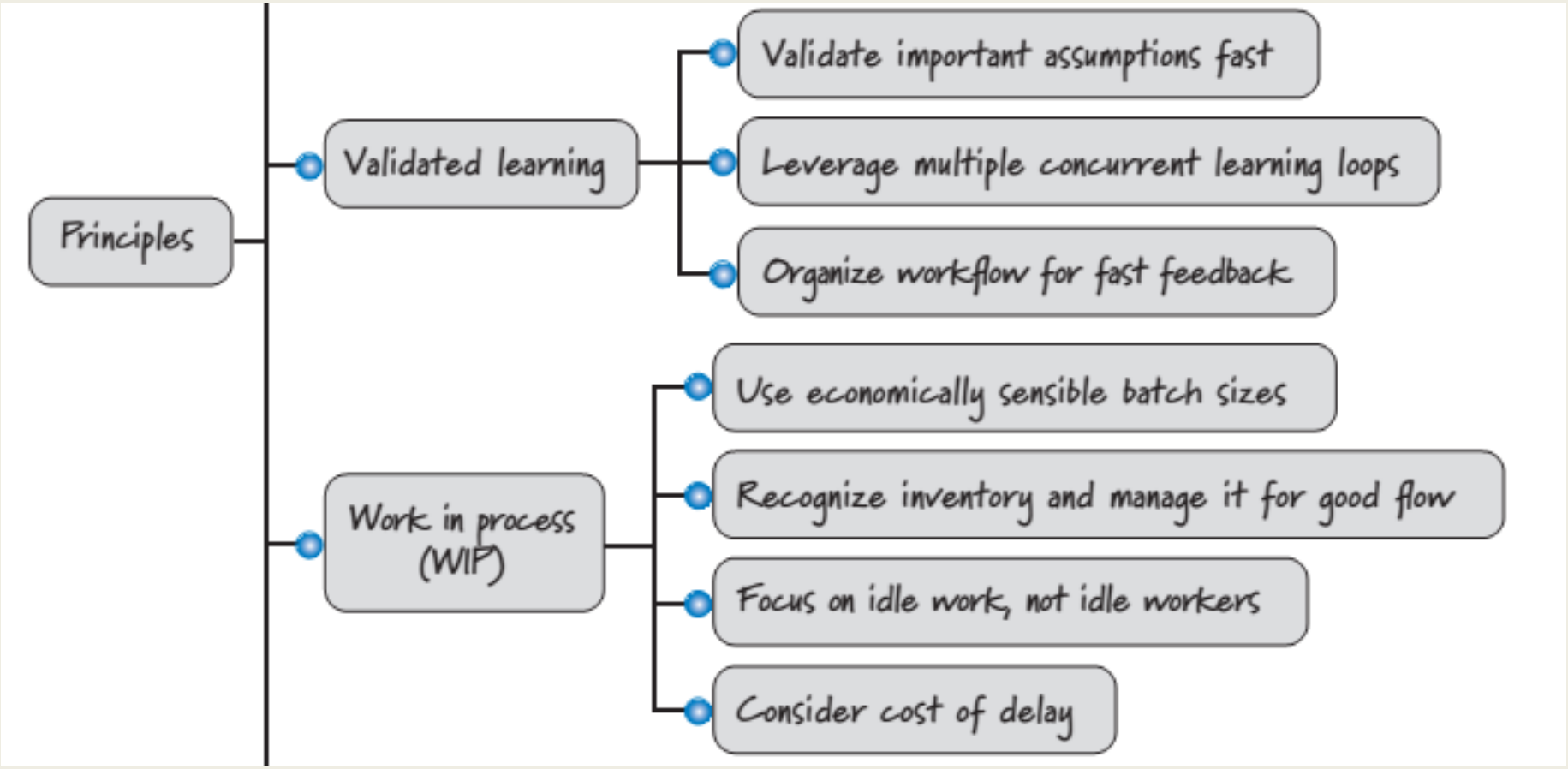
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Principles Behind the Agile Manifesto(III)

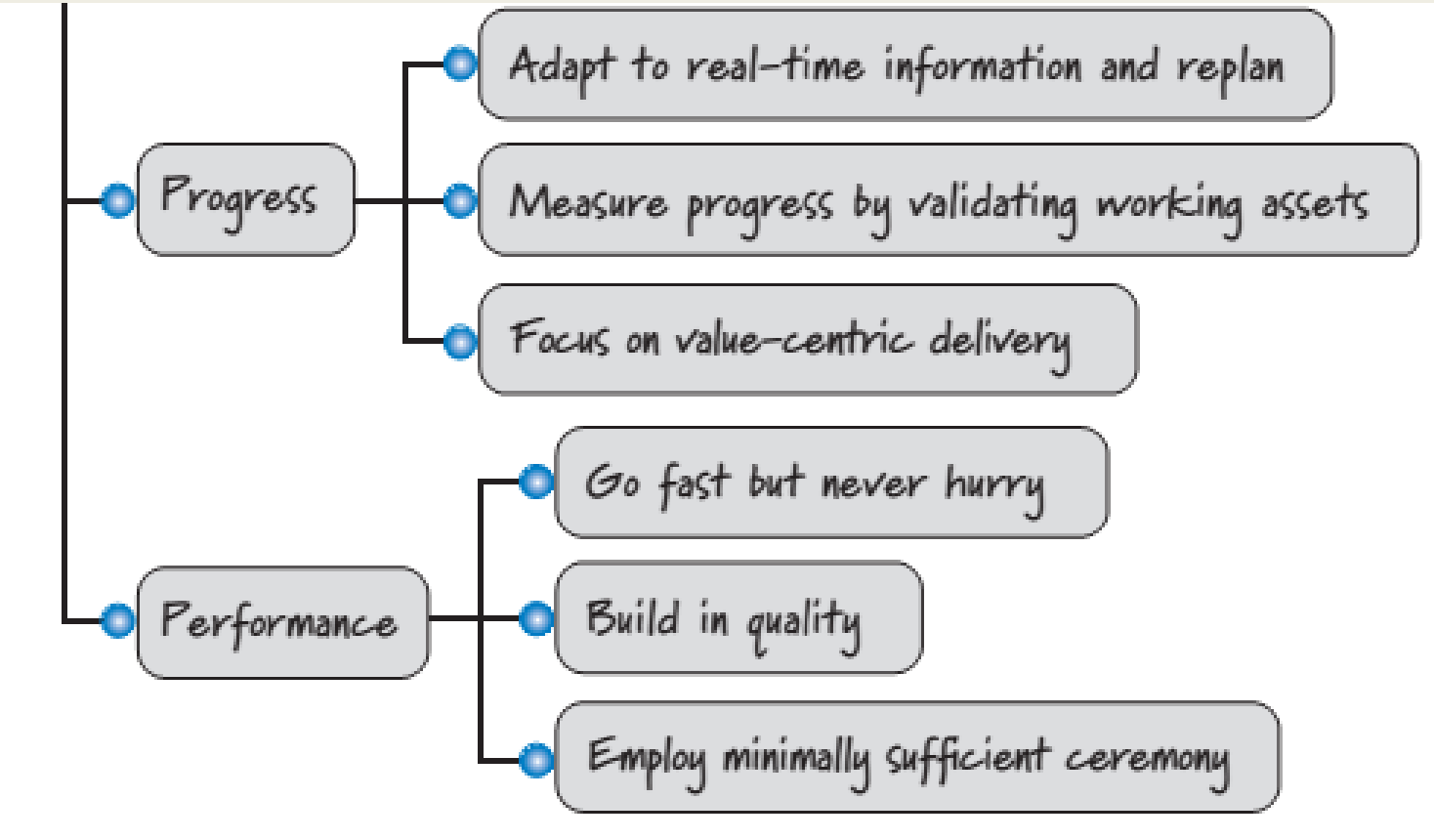
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity–the art of maximizing the amount of work not done–is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Categorization of principles (Up)

Categorization of principles (Middle)



Categorization of principles (Bottom)



References

1- K. S. Rubin, “Essential Scrum, A Practical guide to the most popular agile process,” 2013.

2- J. Sutherland, “Scrum handbook,” 2010.