Software Engineering

HUANG Jie
School of Computer Science & Technology
Tongji University
2025

同济大学 TONGJI UNIVERSITY

Lesson 4
Agile Development

In Lesson 3, we have learned

- ✓ The waterfall process model.
- ✓ The incremental model.
- ✓ The prototyping & spiral model in the evolutionary models.
- ✓ Component-based development.
- ✓ Unified process model.

Lesson 4
Agile Development

In this lesson, we will discuss

- ✓ Agile methods in software development process.
- ✓ Agile principles.
- ✓ Extreme programming(XP).
- ✓ Scrum.
- ✓ Kanban.
- ✓ DevOps.

The 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."

Kent Beck et al

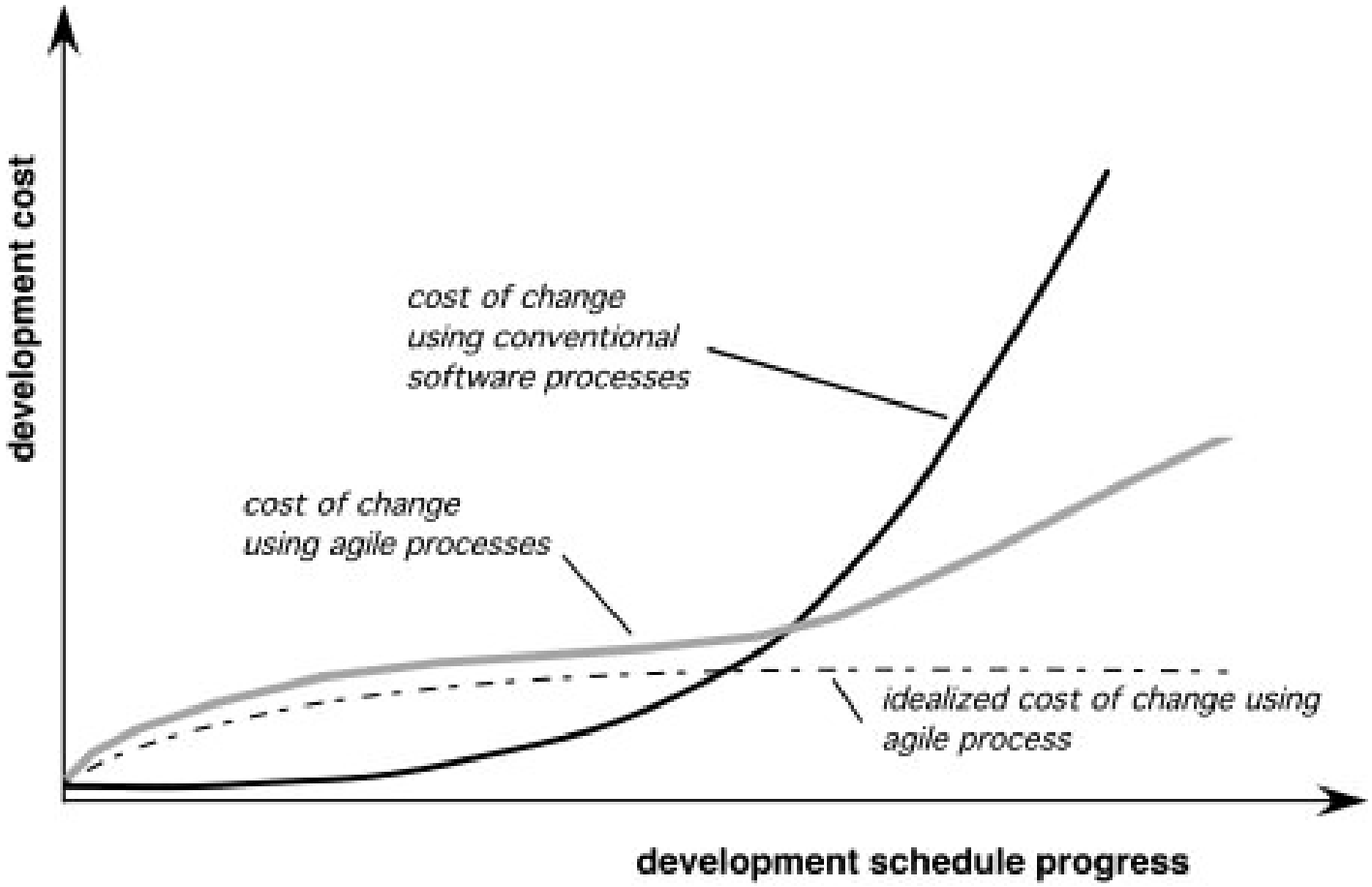
What is "**Agility**" ?

- Effective (rapid and adaptive) response to **change**.
- Effective **communication** among all stakeholders.
- Drawing the **customer onto the team**.
- Organizing a team so that it is in **control of the work** performed.
- Agile means *move quickly*.

Yielding …

- Rapid, Incremental delivery of software.

Agility and the Cost of Change



The Case for Agile Approach

- Huge number of statistics on FAILED projects.
- Huge amounts of WASTE.
- Systems are often NOT deployed to production.
- Users and software developers are polarized.
- Software is almost always delivered late.
- Research shows clearly that software development is NOT manufacturing.

An Agile Process

- Is driven by customer descriptions of what is required (scenarios).
- Recognizes that plans are short-lived.
- Develops software iteratively with a heavy emphasis on construction activities.
- Delivers multiple software increments.
- Adapts as changes occur.

Agility Principles

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.
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 conversation**.

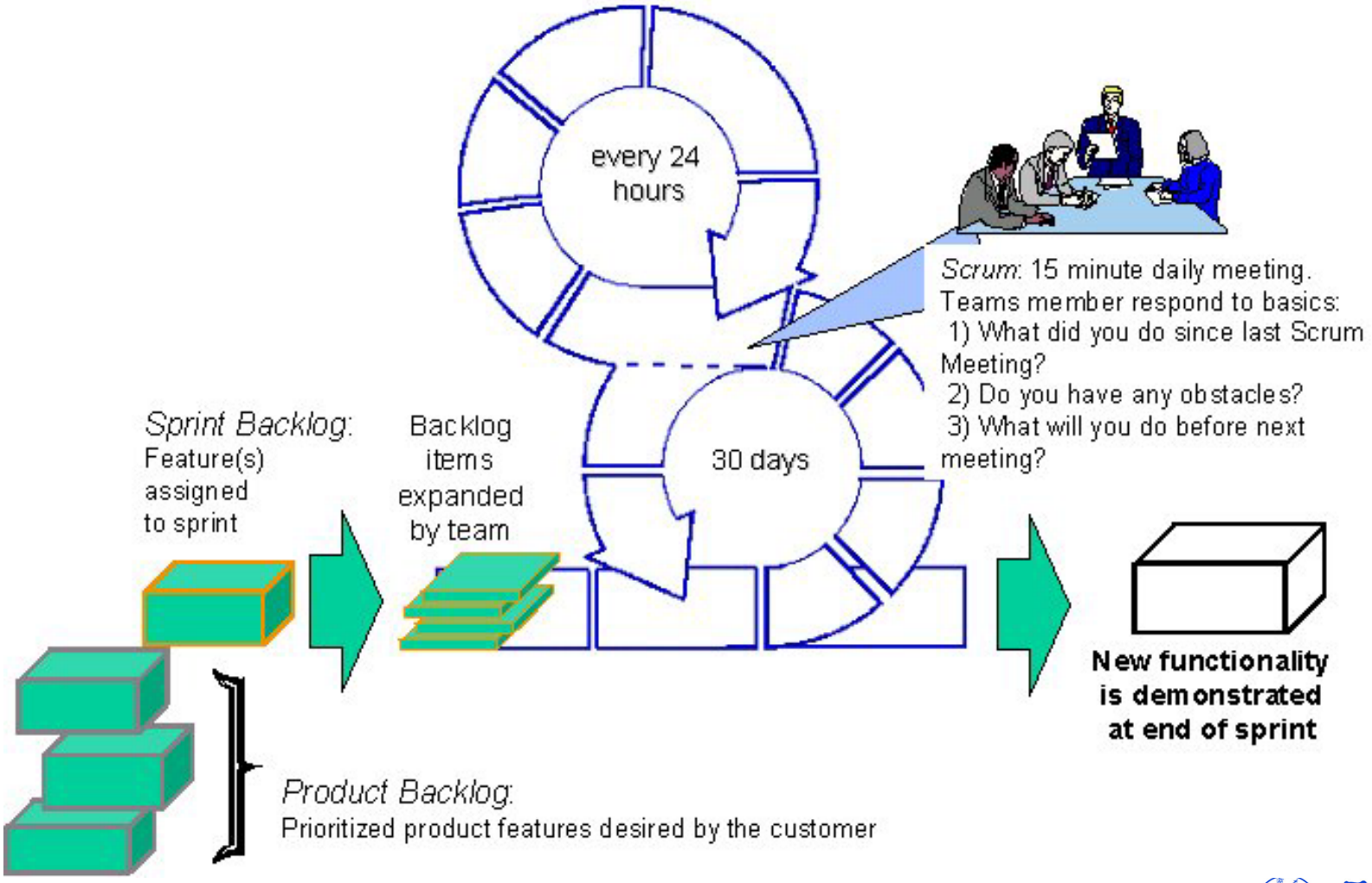
Agility Principles (Cont.)

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.
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.

Scrum

- Originally proposed by Schwaber and Beedle.
- Scrum — distinguishing features
  - Development work is partitioned into "**packets**".
  - **Testing and documentation are on-going** as the product is constructed.
  - Work occurs in "**sprints**" and is derived from a "**backlog**" of existing requirements.
  - **Meetings are very short** and sometimes conducted without chairs.
  - "**Demos**" are delivered to the customer with the time-box allocated.

Scrum



Scrum's THREE ROLES

- Three roles(actors) in Scrum

  Product Owner, Scrum Master, Team.

- Product Owner

  Own and prioritizes the Product Backlog.

- Scrum Master

  Facilitates the Scrum process.

  – NOT a traditional Project Manager !!

- Team

  Produces Increments of Shippable Product Functionality.

Scrum's THREE ROLES

- The Product Owner
  - Defines and Prioritizes Features
    - Owns the gathering of requirements.
  - Agrees to Iteration Ground Rules
    - Set length of calendar time for Sprint.
      - (2,3 or 4 weeks typical)
    - Does not interfere with Sprint(no scope creep).
    - Can pull the plug at any time (has the power).
    - Honors rules and the Scrum process during Sprints.

Scrum's THREE ROLES

- Scrum Master: A Boundary Manager
  - Supports the Team.
  - Facilitates the Daily Scrum meeting. Asks each developer:
    - What did you do yesterday?
    - What are you doing today?
    - What is in your way?
    - Listens and watches carefully during Scrum meeting, Pays careful attention to non-verbal cues.
  - Removes Impediments in Way of Team.
    - Secures resources (monitors, rooms, etc).
  - Communicates to Product Owner.

Scrum's THREE ROLES

- The Team
  - Participates in design
    - To gain understanding of problem/solution space.
  - Selects subset of prioritized Product Backlog for Sprint commitment
    - Estimates the effort.
    - Fills the timebox with work.
    - Commits to the work as a team.
  - Self-Organizes
    - Everyone commits to ALL TASKS necessary during the Sprint.
    - Determines the nature of self-organization.

Scrum's THREE ROLES

- The Team
  - Teams select work for each Sprint.
  - Teams self-organize.
  - Teams have a velocity.
- Team Velocity: How much work the team can average per iteration, FOR THAT TEAM.
  - Each time has a personality.
  - Each team is unique.
  - Teams 'velocity' becomes very predictable over time.

Scrum's THREE CEREMONIES

- Sprint Planning
- Daily Scrum
- Sprint Review (retrospective追溯)

Scrum's THREE CEREMONIES

- Ceremony #1: Sprint Planning Meeting
  - Product Owner reviews
    - Vision, Roadmap, Release Plan.
  - Team reviews
    - Estimates for each item on Backlog that is a candidate for the Sprint.
  - Team pulls the work
    - From the Product Backlog onto the Sprint Backlog.

Scrum's THREE CEREMONIES

- Ceremony #2: The Daily Scrum
  - By and for the Team.
  - Other may attend and NOT speak.
  - Team members speak, others listen.
  - Team stays on task with the 3 questions, divergences(分歧) are addressed offline outside of this meeting.
  - Visibility, clear understanding on a day-by-day basis.
- Product owners know the score on a daily basis
  - Can pull the plug at ANY time.

Scrum's THREE CEREMONIES

- Ceremony #3: Sprint Review Meeting
  - Part 1: Product Demo
    - Led by Product Owner.
  - Part 2: Sprint Retrospective
    - Led by Scrum Master.
    - What worked ?
    - What didn't ?
    - What adjustments can we make now ?

Scrum's THREE ARTIFACTS

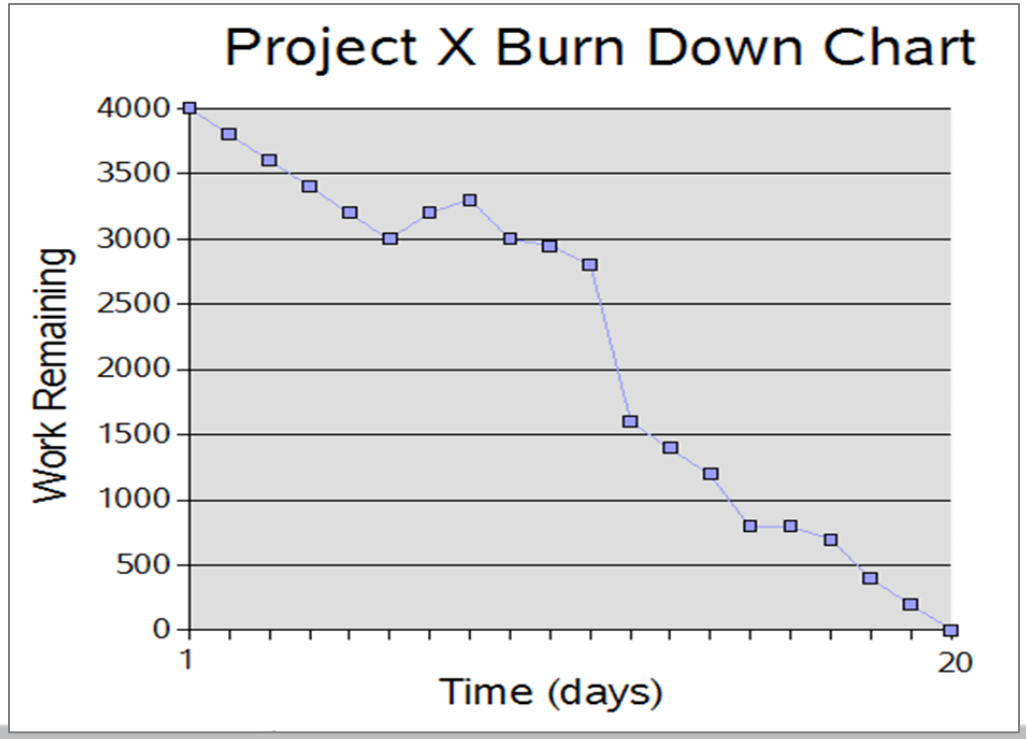
- Artifact #1: Product Backlog
  - A list of features, prioritized by business value.
  - Each feature has an associated estimate, provided by the ACTUAL team who will do the work.
  - Backlog items come in from diverse sources, including the Team.

Scrum's THREE ARTIFACTS

- Artifact #2: Sprint Backlog
  - Topmost subset of the Product Backlog, loaded onto the Sprint's "timebox".
  - Usually has more detail attached, including planned hours and primary person responsible to do the work during the Sprint.
  - Is the list of work the Team is addressing during the current Sprint.

Scrum's THREE ARTIFACTS

- Artifact #3: Burn down Chart
  - Provides visibility into the Sprint.
  - Illustrates progress by the team.
  - Time on the Horizontal, Work on the Vertical.
- Sample BurnDown Chart



Scrum' THREE BEST PRACTICES

- Best Practice #1: User Stories
  - Plain-English requirements, written on common 3 X 5 index cards.
  - Form:

    As [a type of user] I want to [perform a specific action] such that [result].
  - Example:

    "As a web user, I want to make a reservation, such that I may secure my lodging"
  - Stories that are big are called EPICS
  - Acceptance criteria goes on card back

Scrum' THREE BEST PRACTICES

- Best Practice #2: Planning Poker
  - A way for the team to do estimates.
  - Each participant has cards numbered 1,2,3,5,8,13,21.
  - Values represent 'story points' of effort.
  - Players discuss feature, then throw down a card together.
  - Differences are noted and discussed, then process repeats till a concensus(协议) estimate is formed.

Scrum' THREE BEST PRACTICES

- Best Practice #3: Use of the Scrum Board
  - Scrum Board is a rows-and-columns depictions of work-in-progress.
  - Items of work are rows, work status labels are columns.
  - Work is addressed from top to bottom.
  - Work migrates from left to right on the board.

A piece of Video about Scrum

- Introduction to Scrum (about 7 Minutes).
  - You can see and download this video from Canvas.

- **Question** ?
  - Do you remember some keywords about SCRUM ?

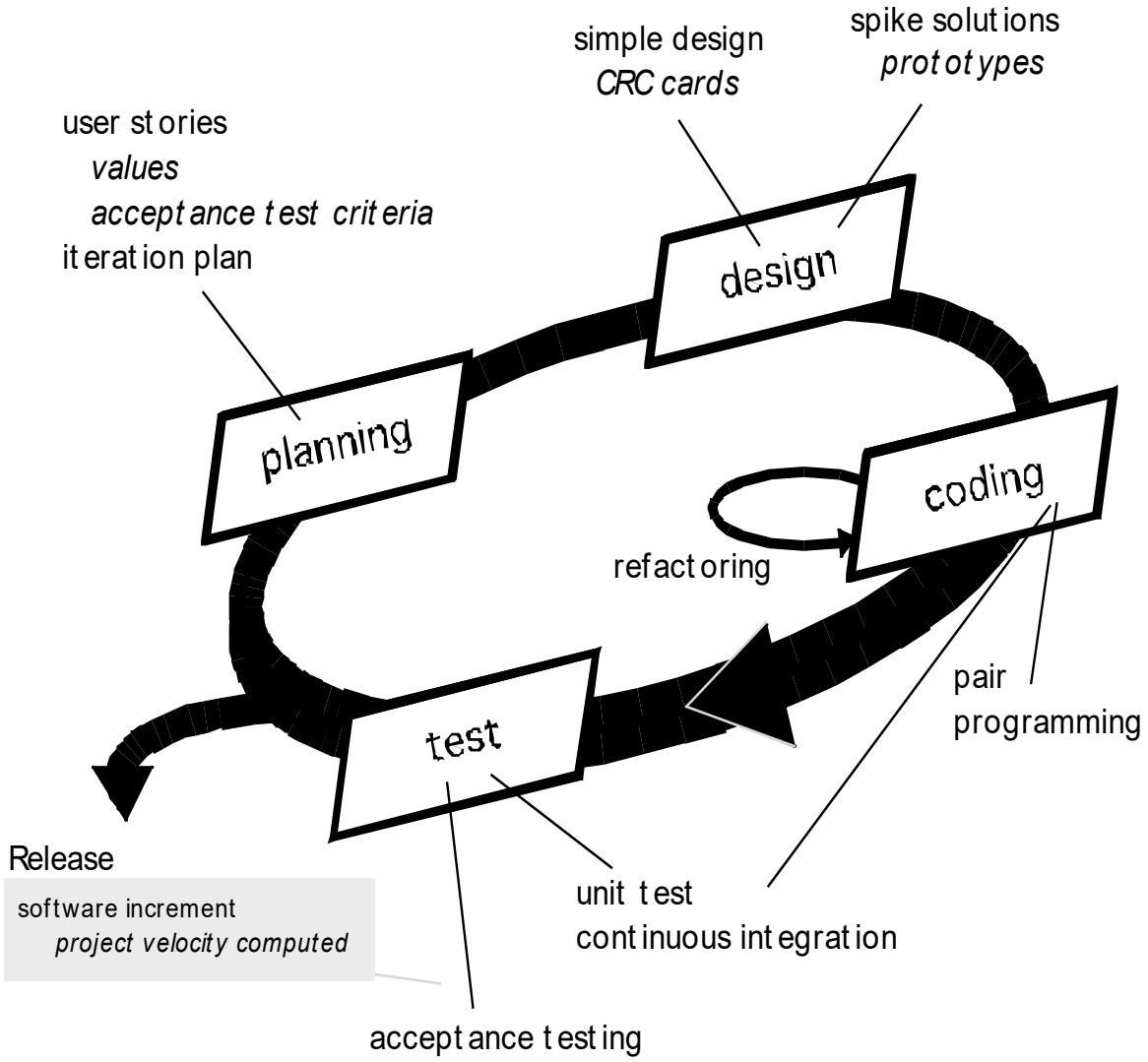
Extreme Programming (XP)

- The most widely used XP programming process, originally proposed by Kent Beck.
- XP Planning
  - Begins with the creation of "**user stories**".
  - Agile team assesses each story and assigns a **cost**.
  - Stories are grouped to for a **deliverable increment**.
  - A **commitment** is made on delivery date.
  - After the first increment, "**project velocity**(速度)" is used to help define subsequent delivery dates for other increments.

Extreme Programming (XP)

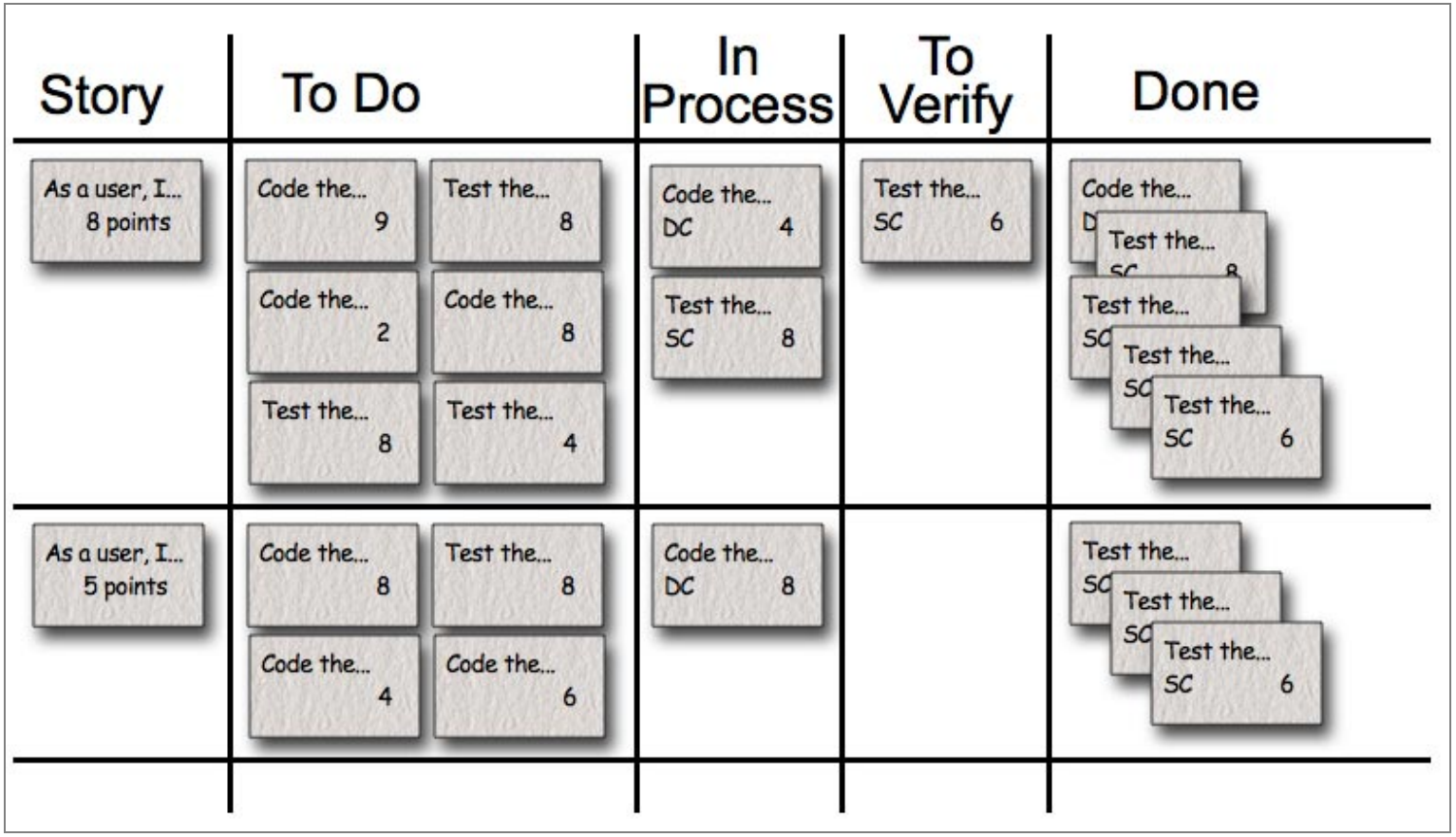
- XP Design
  - Follows the **KIS** principle.
  - Encourage the use of **CRC** cards (see Chapter 8).
  - For difficult design problems, suggests the creation of "**spike solutions**"—a design prototype.
  - Encourages "**refactoring**"— an iterative refinement of the internal program design.
- XP Coding
  - Recommends the **construction of a unit test** for a store *before* coding commences.
  - Encourages **pair programming**.
- XP Testing
  - All **unit tests are executed daily**.
  - **"Acceptance tests"** are defined by the customer and executed to assess customer visible functionality.

Extreme Programming (XP)



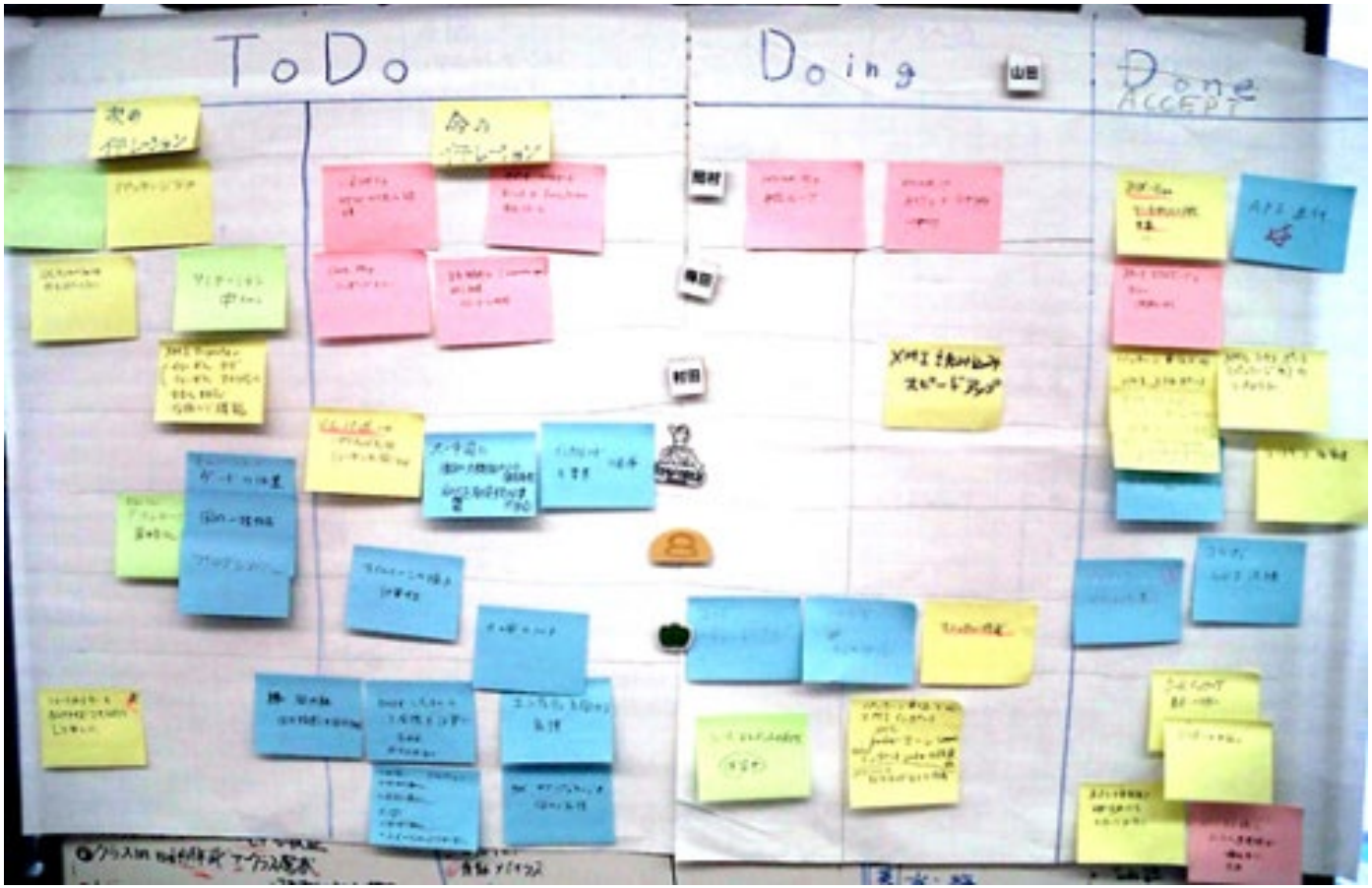
Kanban method

- Kanban Board

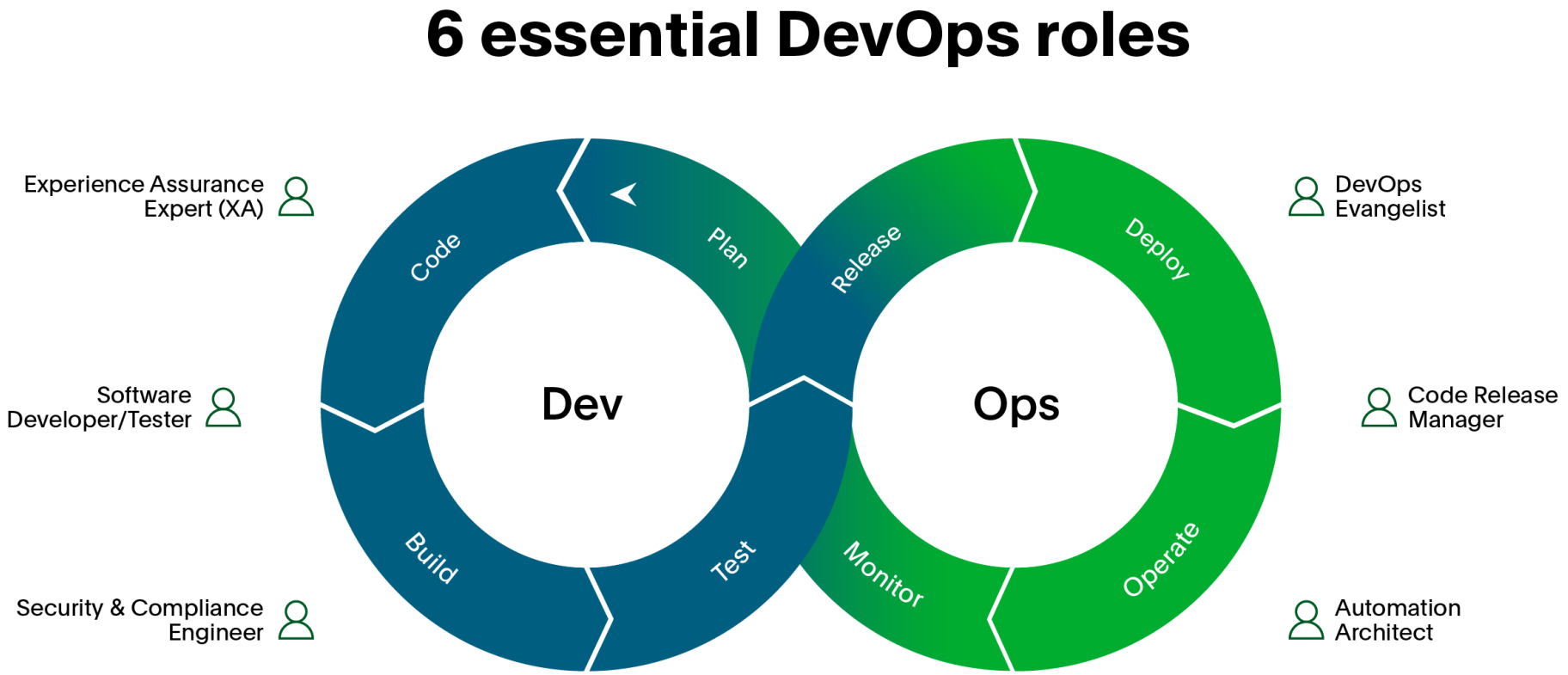


Kanban method

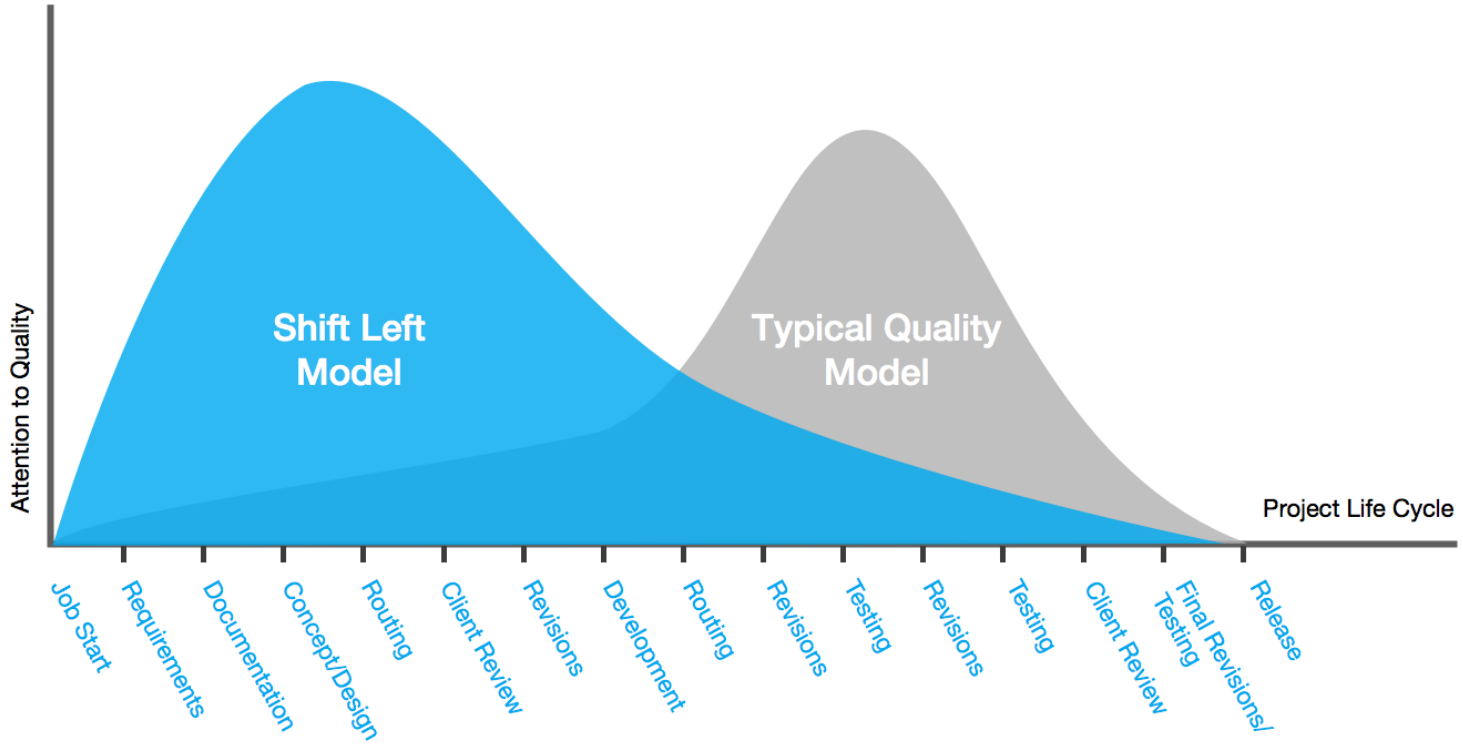
- Sample: Kanban Board



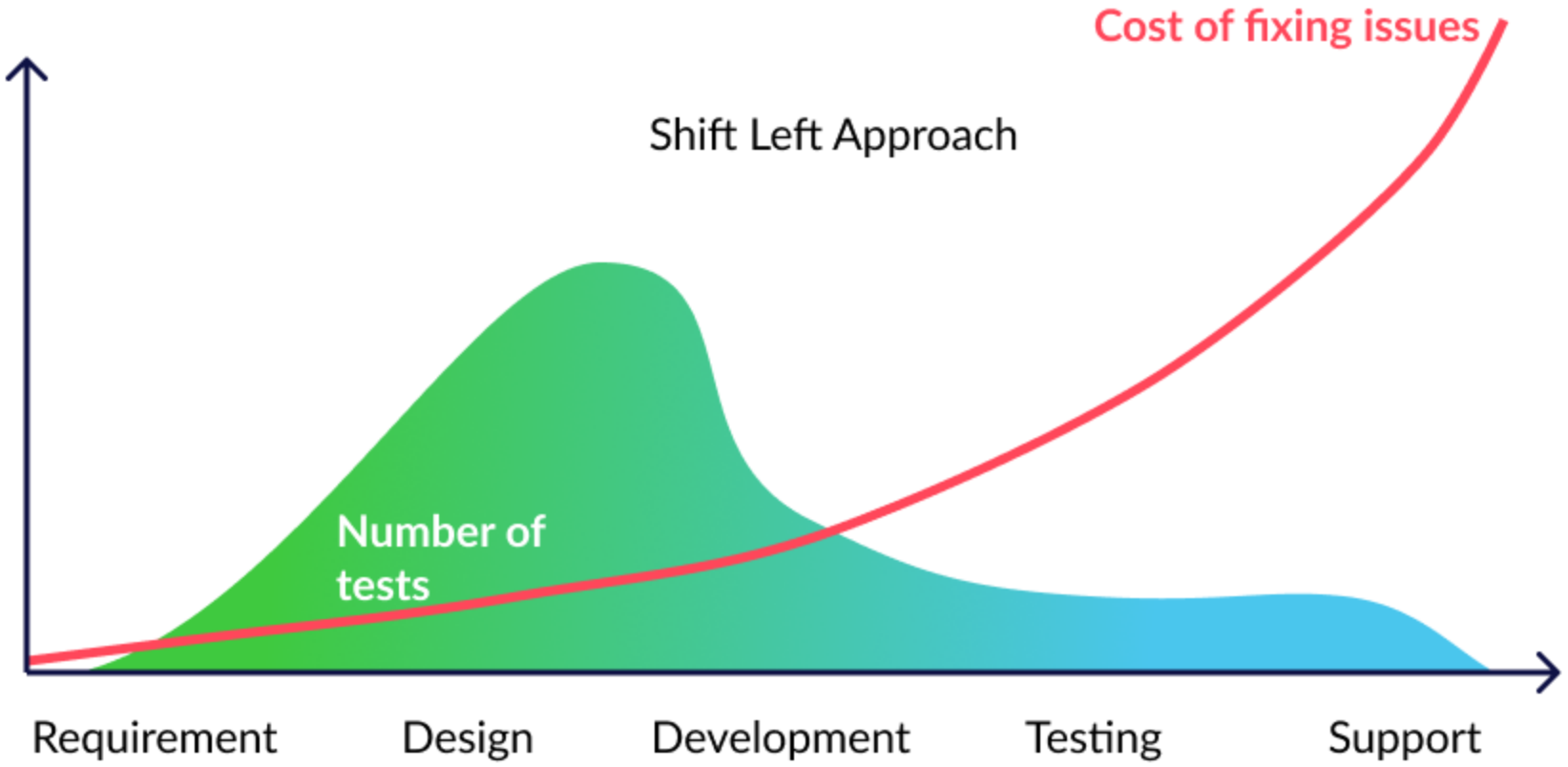
DevOps



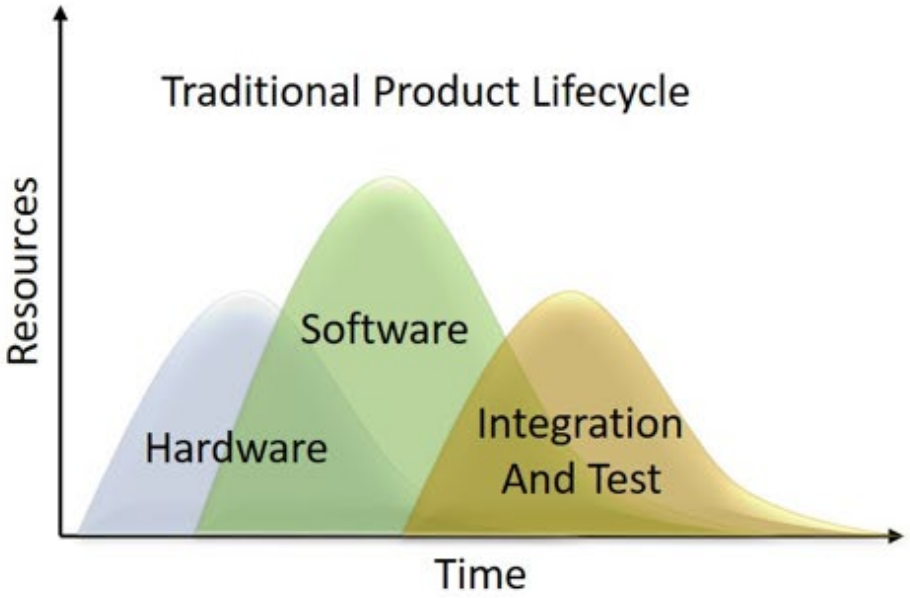
DevOps

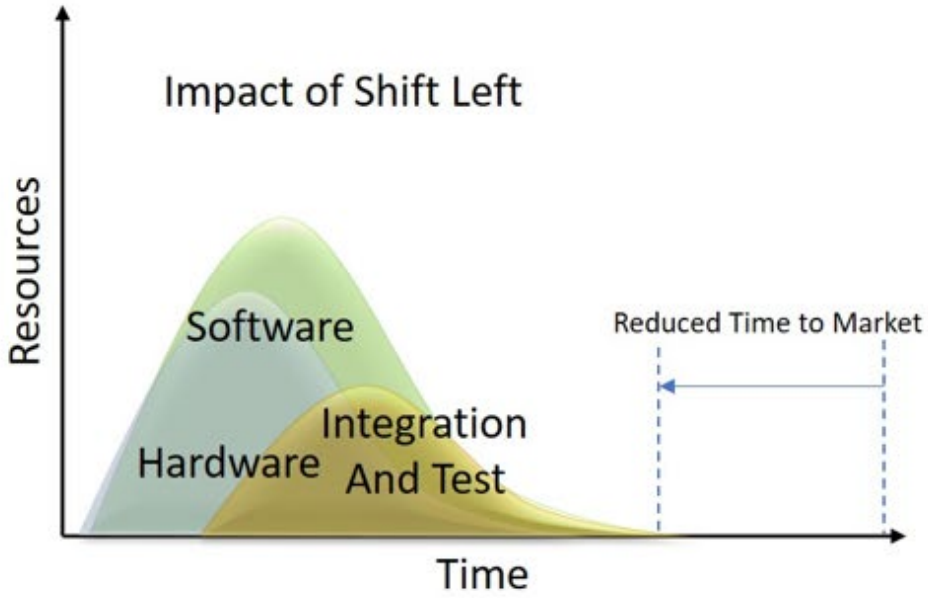


DevOps



DevOps





Agile Process

- Each AUP iteration addresses these activities
  - Modeling.
  - Implementation.
  - Testing.
  - Deployment.
  - Configuration and project management.
  - Environment management.

Conclusion

- Differences for developers
  - Agile: knowledgeable, collocated, collaborative.
  - Heavyweight: plan-driven, adequate skills, access to external knowledge.
- Differences for customers
  - Agile: dedicated, knowledgeable, collocated, collaborative, representative, empowered.
  - Heavyweight: access to knowledgeable, collaborative, representative, empowered customers.
- Differences for primary objective
  - Agile: rapid value.
  - Heavyweight: high assurance.

Conclusion

- Differences for requirements
  - Agile: largely emergent, rapid change.
  - Heavyweight: knowable early, largely stable.
- Differences for architecture
  - Agile: designed for current requirements.
  - Heavyweight: designed for current and foreseeable requirements.
- Differences for size
  - Agile: smaller teams and products.
  - Heavyweight: larger teams and products.

Summary

- During the development of any type of software, adherence to a suitable process model has become universally accepted by software development organizations. Adoption of a suitable life cycle model is now accepted as a primary necessity for successful completion of projects.
- We discussed only the central ideas behind some important process models. Good software development organizations carefully and elaborately document the precise process model they follow and typically include the following in the document:
  - Identification of the different phases.
  - Identification of the different activities in each phase and the order in which they are carried out.
  - The phase entry and exit criteria for different phases.
  - The methodology followed to carry out the different activities.

Summary

- Adherence to a software life cycle model encourages the team members to perform various development activities in a systematic and disciplined manner. It also makes management of software development projects easier.
- The principle of detecting errors as close to their point of introduction as possible is known as phase containment of errors. Phase containment minimizes the cost to fix errors.
- The classical waterfall model can be considered as the basic model and all other life cycle models are embellishments of this model. Iterative waterfall model has been the most widely used life cycle model so far, though agile models have gained prominence lately.
- Even though an organization may follow whichever life cycle model that may appear appropriate to a project, the final document should reflect as if the software was developed using the classical waterfall model. This makes it easier for the maintainers to understand the software documents.

Assignment

Different software process models have their own advantages and disadvantages. Therefore, an appropriate process model should be chosen for the real requirements of project.

Design a table and make comparisons between these software process models, point out the advantages and disadvantages of each model.

Note 1: At least five different models need to be compared in your assignment.

Note 2: A specific requirement proposed by teacher in the classroom should be considered in your assignment.

The answer of the assignment should be mailed to Canvas.

Deadline of this assignment：23:59 Oct. 25th 2025

Assignment

Preview

《Software Engineering》9th Edition by R.S.Pressman

Chapter 4 Recommended software process

《Software Engineering》8th Edition by R.S.Pressman

Chapter 6 Human aspect of Software Engineering

huangjie@tongji.edu.cn