cuiatd.edu.pk

subexpert.com

COMSATS University Islamabad
Abbottabad Campus

Computer Science Department

CSE304 Object Oriented Software Engineering

`

Lecture 3 – Agile Principles and Practices

Mukhtiar Zamin,
MS (Computer Science)
Iowa, United States of America
mukhtiar@cuiatd.edu.pk

ACKNOWLEDFMENT

- Lectures are prepared from the following books:
  - Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development
    By: Craig Larman
    - https://www.amazon.com/Applying-UML-Patterns-Introduction-Object-Oriented-ebook/dp/B000OZ0N8U
  - Design Patterns: Elements of Reusable Object-Oriented Software
    - Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides
  - In addition there are other examples and code snippets from other sources which are mentioned on respective slides

Lecture 3

- Risk-Driven and Client-Driven Iterative Planning
- Agile Methods and Attitudes
- The Agile Manifesto
- Agile Principles
- Agile Modeling
- Agile UP
- Other Critical UP Practices

Risk-Driven and Client-Driven Iterative Planning

- The UP (and most new methods) encourage a combination of risk-driven and client-driven iterative planning.
  - Identify and drive down the highest risks. Early iterations focus on building, testing, and stabilizing the core architecture
  - Build visible features that the client cares most about

Agile Methods and Attitudes

- Agile Method
  - Apply timeboxed iterative and evolutionary development
  - Employ adaptive planning
  - Promote incremental delivery, and
  - Include other values and practices that encourage agility—rapid and flexible response to change.

- Not possible to exactly define agile methods, as specific practices vary widely.

- Short time-boxed iterations with evolutionary refinement of plans, requirements, and design is a basic practice the methods share.

The Agile Manifesto

- Individuals and interactions
  - over processes and tools
- Working software
  - over comprehensive documentation
- Customer collaboration
  - over contract negotiation
- Responding to change
  - over following a plan

Agile Principles

1. **Satisfy the customer** through early and continuous delivery of valuable software.

2. **Welcome changing requirements**, even late in development.

3. **Deliver working software frequently** (shorter time scale)

4. Business people and developers **must work together**

5. Build projects around **motivated individuals**.

6. Use **face-to-face conversation** to convey information to and within a development team

7. **Working software** is the primary measure of **progress**.

The Agile Principles

8. Agile process promote **sustainable development**

9. The sponsors, developers, and users should be able to maintain **a constant pace** indefinitely

10. **Continuous attention to technical excellence** and good design enhances agility

11. Simplicity—the art of **maximizing the amount of work not done**—is essential

12. The best architectures, requirements, and designs emerge from **self-organizing teams**

13. At regular intervals, the team reflects on **how to become more effective**, then tunes and adjusts its behavior accordingly.

Agile Alliance (www.agilealliance.com)

Agile Modeling

- The purpose of modeling (sketching UML, …) is primarily to understand, not to document.
- It implies a number of practices and values, including:

  - Adopting an agile method does not mean avoiding any modeling. Many agile methods, such as Feature-Driven Development, Dynamic systems development method (DSDM), and Scrum, normally include significant modeling sessions.
  - The purpose of modeling and models is primarily to support understanding and communication, not documentation
  - Don't model or apply the UML to all or most of the software design.
  - Use the simplest tool possible
  - Don't model alone, model in pairs (or triads)
  - Create models in parallel.
  - Know that all models will be inaccurate, and the final code or design different—sometimes dramatically different—than the model. Only tested code demonstrates the true design
  - Use "good enough" simple notation
  - Developers themselves should do the OO design modeling

Agile UP

- **Agile** (foaled 1902 in Kentucky) was an American Thoroughbred racehorse that was the winner of the 1905 Kentucky Derby
- UP was not meant by its creators to be heavy or un-agile, it was meant to be adopted and applied in the spirit of adaptability and lightness—an agile UP
- Prefer a small set of UP activities and artifacts
- Since the UP is iterative and evolutionary, requirements and designs are not completed before implementation. They adaptively emerge through a series of iterations, based on feedback.
- Apply the UML with agile modeling practices

- There isn't a detailed plan for the entire project. There is a high-level plan (called the Phase Plan) that estimates the project end date and other major milestones

Other Critical UP Practices

Some additional best practices and key concepts in the UP:

- Tackle high-risk and high-value issues in early iterations
- Continuously engage users for evaluation, feedback, and requirements
- Build a cohesive, core architecture in early iterations
- Continuously verify quality; test early, often, and realistically
- Apply use cases where appropriate
- Do some visual modeling (with the UML)
- Carefully manage requirements
- Practice change request and configuration management