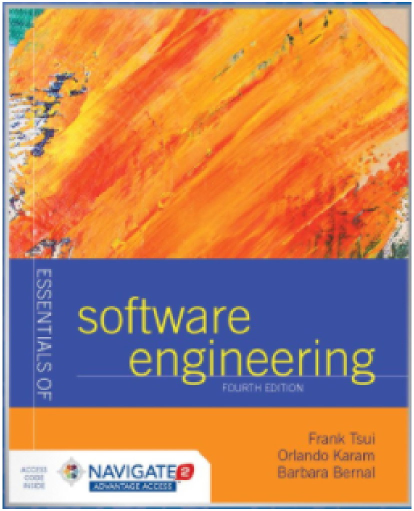
New and Emerging Process Methodologies

CSC 354, Software Engineering I

Chapter 5: New and Emerging Process Methodologies



Problems with “Traditional” Processes

- Focused on and oriented towards “large projects” and *lengthy development time* – years
- Inability to cope with *changes in requirements* and technology fast enough
- Assumes requirements are *completely understood* at the beginning of the project

Problems with “Traditional” Processes (continued)

- Relying on non-sustainable *heroic* and lengthy development *effort* by developers
- *Complex* set of activities
- *Duplication* of effort, especially in documentation

The Agile Manifesto

"We are uncovering better ways of developing software by doing it and helping others to it. That is, while there is value in the items on the"right," we value the items on the "left" more."

"left"	"right"
Individuals and interactions	Processes and Tools
Working software	Comprehensive documentation
Customer collaboration	Contract Negotiation
Responding to change	Following a plan

More Recent Processes: Agile Methodologies

- Family of software development methodologies:
  - *Short* releases and multiple iterations
  - *Incremental* design/development
  - *User involvement* – especially for in-house
  - *Minimal* documentation
  - Assume *changes*

Some Agile Methodologies

- Extreme Programming (XP): most popular
- Crystal Clear/Orange: one size does not fit all
- Scrum: currently popular; not really part of Agile (partially Agile)
- Rational Unified Process (RUP): heavy process
- Kanban: visual cards for tasks; minimize work-in-progress

Extreme Programming Core Values

- Communication: between team and with customers
- Simplicity: in design and code
- Feedback: at many levels, team and customer
- Courage: to make and implement difficult decisions

Extreme Programming Fundamental Principles

- *Rapid feedback* through unit testing, integration, and short releases
- *Simplicity*: happy path first
- *Incremental change*: small changes add up
- *Embrace change*: preserve options
- *Quality work*: do the best work all the time

Extreme Programming Lesser/Other Principles

- Ongoing learning
- Playing to win
- Open, honest communication
- Concrete experiments
- Traveling light
- Working with people's instincts
- Accepting responsibility
- Honest measurement
- Local adaptation
- Small initial investment

Extreme Programming (XP)

- XP takes an “extreme” approach to iterative development
  - New versions may be built several times per day
  - Increments are delivered to customers every two weeks
  - All tests must be run for every build and the build is only accepted if tests are successful

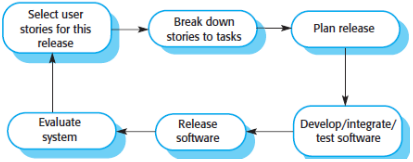
XP's 12 Key Practices

1. Planning Game: small units of requirements
2. Onsite Customer: immediate and better feedback
3. Metaphor: use one set of metaphors for design/architecture
4. Simple Design: just enough to cover what is needed
5. Coding Standard: facilitates better communication
6. Collective Code Ownership: peer pressure to improve code

XP's 12 Key Practices (continued)

7. Pair Programming: feedback and shared improvements
8. Refactoring: continuous examination for duplicative design/code
9. Continuous Functional and Unit Testings: 100% completion
10. Small/Short Releases
11. Continuous Integration
12. 40 Hour Work: high morale and energy level

Extreme Programming Release Cycle


Select user stories for this release
Break down stories to tasks
Plan release
Develop/integrate/ test software
Release software
Evaluate system

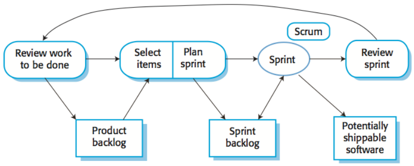
Scrum Development Process

- Introduced by Takeuchi and Nonaka in 1986 modeled after the way a rugby game is played
- Ken Schwaber and Mike Beedle published a book, “Agile Software Development with Scrum,” in 2001
- It is an incremental and iterative approach
  - Develops small sprints, or increments (of features) in short cycles of about 2-3 weeks

Scrum Development Process

- There are three main roles
  - Product Owner who talks to and decides with users about the content of each sprint
  - Scrum Master who runs the sprints
  - Scrum Team of about 7-8 members who develop the sprint

Scrum Sprint Cycle



Scrum

- Answer these three questions:
  1. What did you get done last time?
  2. What are you going to do this time?
  3. Any blockers?
- Note “time” can be arbitrary, usually a day or week

Scrum Benefits

- The product is broken into a set of manageable and understandable chunks
- Unstable requirements do not hold up progress
- The whole team have visibility into everything and consequently team communication is improved
- Customers see on-time delivery of increments and gain feedback on how the product works
- Trust between customers and developers is established and a positive culture is created in which everyone expects the project to succeed