SoberIT
Software Business and Engineering Institute

Testing in Agile Methodologies and Extreme Programming

Maaret Pyhäjärvi
Software Business and Engineering Institute
(SoberIT)

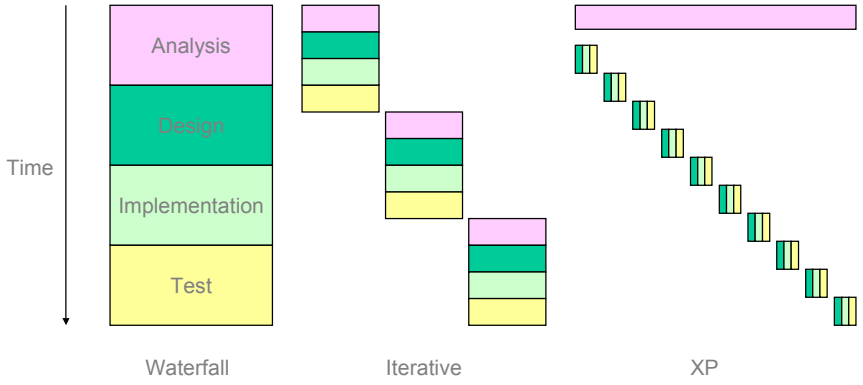
SoberIT
Software Business and Engineering Institute

Agile Methodology

- Impossible to name one best and correct way to develop software
- Light-but-sufficient rules of project behavior anad the use of human- and communication-oriented rules
- Examples:
  - eXtreme Programming (XP) - Beck
  - Adaptive Software Development – Highsmith
  - Scrum – Schwaber, Sutherland
  - Crystal family - Cockburn

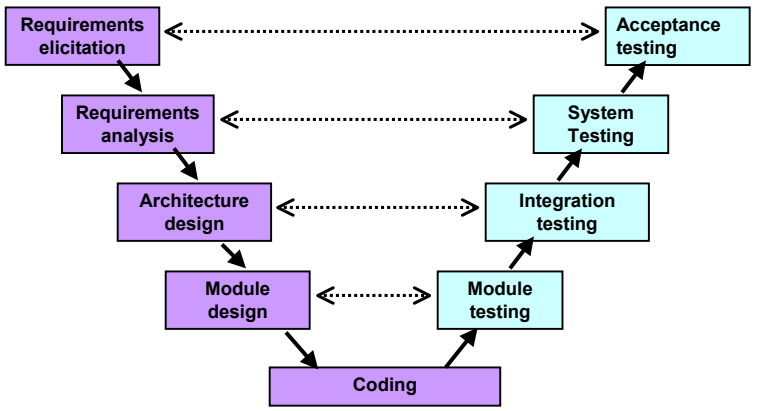
SoberIT
Software Business and Engineering Institute

From Waterfall to XP



SoberIT
Software Business and Engineering Institute

V-model



Specifications → *Planning* → *Testing*

Iterative and Incremental Development

- **Iterative** refers to a scheduling and staging strategy that allows rework of pieces of the system
- **Incremental** refers to a scheduling and staging strategy in whuch pieces of the system are developed at different rates and integrated as they are developed.
- ”Estimating testing is different” -> regression becomes part of everything in an iterative and incremental environment

SoberIT
Software Business and Engineering Institute

eXtreme Programming (XP)

- Agile software development methodology
- Small teams
  - 2-10 person
- Vague or rapidly changing requirements
- High discipline practices
- Common sense principles and practices taken to extreme levels
  - testing, reviews, simplicity, ...
- Kent Beck
  - C3 project 1996

SoberIT
Software Business and Engineering Institute

XP vs. Other Methodologies

- Early, concrete and continuing feedback from short cycles
- Incremental planning and flexible scheduling
- Automated tests
- Oral communication, tests, and code to communicate system structure and intent
  - tests and source code are the technical documentation
- Evolutionary design process lasts as long as the system lasts
- Close collaboration of programmers
- Practices work with both the short-term instincts of programmers and long-term interest of the project

SoberIT
Software Business and Engineering Institute

What Does XP Recommend We Test?

- Unit tests
- Acceptance tests
- Performance / load tests
- By Hand testing

Rules of XP Testing

- All code must have unit tests
- All code must pass unit tests before it can be released
- When a bug is found, tests are created
- Acceptance tests are run often and the score is published
- All tests should be automated
- Test performance when needed
- *If it doesn't have a test it doesn't exist*

SoberIT
Software Business and Engineering Institute

Four Values of XP (1/2)

- Short-term individual goals vs. long-term social goals
  - common values required
- Communication
  - punisment from bad news kills communication
  - practices that keep people communicating
- Simplicity
  - What is the most simplest thing that could possibly work?
  - pay a little more later to change the simple solution if needed vs. make a complicated solution that may not be needed at all

Four Values of XP (2/2)

- Feedback
  - Concrete feedback about the state of the system
  - Scale of minutes and days
    - unit tests
    - writing and estimating stories
    - progress
  - Scale of weeks and months
    - functional tests
    - schedule review
    - system into production as soon as it makes sense
      - development is a temporary state
- Courage
  - changing the system
  - throwing code away
  - pair programming

SoberIT
Software Business and Engineering Institute

Basic principles

- Derived from the four values
- Fundamental principles
  - rapid feedback
    - critical for learning
  - assume simplicity
    - treat every problem as if it can be solved with ridiculous simplicity
  - incremental change
    - series of smallest changes that make a difference
  - embracing change
    - best strategy preserves most of the options while actually solving your most pressing problem
  - quality work
    - everybody likes doing a good job
- Secondary principles
  - concrete experiments
  - accepted responsibility
  - travel light
  - local adaptation
  - ...

Practices - overview

- Simple practices that support each other
- Not unique or original
- Have weaknesses
- How XP could work
  - practices support each other's weaknesses
  - exponential change cost is collapsed

- Practices
  - The planning game
  - Short releases
  - Metaphor
  - Simple design
  - Testing
  - Refactoring
  - Pair programming
  - Collective ownership
  - Continuous integration
  - 40-hour week
  - On-site customer
  - Coding standard

SoberIT
Software Business and Engineering Institute

Roles

- Programmer
  - communication
- Customer
  - stories and functional tests
- Tester
  - helps customer write tests
  - runs functional tests regularly
- Tracker
  - collects data face-to-face
  - gives feedback on estimates
  - keeps an eye on the big picture

- Coach
  - responsible of the process
  - notices deviations from it
  - indirect intervention
- Consultant
  - teaches the team to solve a problem
  - code is rewritten
- Big Boss
  - courage
  - trust
  - resources