Summer School

eXtreme programming principles & practices

Sylvia Ilieva, Eliza Stefanova
Sofia University

Radoslav Nemchev
Nemetschek

Penko Ivanov
Rila Solutions
eXtreme programming

Principles

Sylvia Ilieva
Sofia University
Contents

• Agile methodologies
• What is XP?
• Risk: the basic problem and how XP address risks
• Basic principles
• Basic activities
• Practices
Agile alliance

- February 2001 – group of 17 people
- [www.AgileAlliance.org](http://www.AgileAlliance.org)
- Agile word
- The manifesto
- Principles and recommendations
Methodology

- Software development methodology – the set of conventions the team adopts
- Methodology can be lightened
  - By getting running software out for examination sooner
  - By increasing the richness of the communication channels
- Light delivers more quickly and with reduced cost
Light methodologies

- XP
- Adaptive Software Development
- Scrum
- Crystal
- Feature Driven Development
- DSDM (Dynamic System Development Method)
Why “extreme”

XP takes commonsense principles and practices to extreme levels.

• If code reviews are good, we'll review code all the time (pair programming).

• If testing is good, everybody will test all the time (unit testing), even the customers (functional testing).
Why “extreme”-2

- If design is good, we'll make it part of everybody's daily business (refactoring).
- If simplicity is good, we'll always leave the system with the simplest design that supports its current functionality (the simplest thing that could possibly work).
- If architecture is important, everybody will work defining and refining the architecture all the time (metaphor).
Why “extreme”-3

- If integration testing is important, then we'll integrate and test several times a day (continuous integration).
- If short iterations are good, we'll make the iterations really, really short—seconds and minutes and hours, not weeks and months and years (the Planning Game).
XP promises

• To reduce project risk,
• To improve responsiveness to business changes,
• To improve productivity throughout the life of a system,
• To add fun to building software in teams all at the same time.
What Is XP?

XP is a
- lightweight,
- efficient,
- low-risk,
- flexible,
- predictable,
- scientific, and
- fun
way to develop software.
Innovation of XP

• Putting all proven practices (for the implementation strategy and for the management strategy) under one umbrella

• Making sure they are practiced as thoroughly as possible.

• Making sure the practices support each other to the greatest possible degree.

---
eXtreme programming - principles & practices
Risk: The Basic Problem

- Schedule slips
- Project canceled
- System goes sour
- Defect rate
- Business misunderstood
- Business changes
- False feature rich
- Staff turnover

*eXtreme programming - principles & practices*
How does XP address risks?

• Schedule slips
  – short Release cycles - a few months at most,
  – Iterations of customer requested features - one- to four-week.
  – Tasks - one- to three-day
  – implementing the highest priority features first
How does XP address risks?

- **Project canceled**
  - XP asks the customer to choose the smallest release that makes the most business sense.

- **System goes sour**
  - comprehensive suite of tests, which are run and re-run after every change.
  - XP always keeps the system in prime condition.
How does XP address risks?

- Defect rate - XP tests from the perspective of
  - programmers - writing tests function-by-function
  - customers - writing tests program-feature-by-program-feature.

- Business misunderstood
  - customer to be an integral part of the team.
How does XP address risks?

- **Business changes**
  - XP shortens the release cycle, so there is less change during the development of a single release.

- **False feature rich**
  - only the highest priority tasks are addressed.

- **Staff turnover**
  - programmers accept responsibility for estimating and completing their own work
  - human contact among the team
XP development cycle

• Pairs of programmers program together.
• Development is driven by tests. You test first, then code.
• Pairs don't just make test cases run. They also evolve the design of the system. Pairs add value to the analysis, design, implementation, and testing of the system.
• Integration immediately follows development, including integration testing.
Four variables

- Cost
- Time
- Quality
- Scope
Four Values

- Communication
- Simplicity
- Feedback
- Courage
Basic principles

• Rapid feedback
• Assume simplicity
• Incremental change
• Embracing change
• Quality work
Less central principles

• Teach learning
• Small initial investment
• Play to win
• Open, honest communication
• Accepted responsibility
• Local adaptation
Basic Activities

- Coding
- Testing
- Listening
- Designing
Practices

- **The Planning Game** — Quickly determine the scope of the next release by combining business priorities and technical estimates. As reality overtakes the plan, update the plan.

- **Small releases** — Put a simple system into production quickly, then release new versions on a very short cycle.
Practices - 2

- **Metaphor** — Guide all development with a simple shared story of how the whole system works.

- **Simple design** — The system should be designed as simply as possible at any given moment. Extra complexity is removed as soon as it is discovered.
Practices - 3

- **Testing** — Programmers continually write unit tests, which must run flawlessly for development to continue. Customers write tests demonstrating that features are finished.

- **Refactoring** — Programmers restructure the system without changing its behavior to remove duplication, improve communication, simplify, or add flexibility.
Practices - 4

- **Pair programming** — All production code is written with two programmers at one machine.
- **Collective ownership** — Anyone can change any code anywhere in the system at any time.
- **Continuous integration** — Integrate and build the system many times a day, every time a task is completed.
Practices - 5

• **40 hour week** — Work no more than 40 hours a week as a rule. Never work overtime a second week in a row.

• **On-site customer** — Include a real, live user on the team, available full-time to answer questions.

• **Coding standards** — Programmers write all code in accordance with rules emphasizing communication through the code.