Agile Methodology and eXtreme Programming

Charles Carter

August 30, 2017

Table of Contents

Agile Manifesto

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

Agile Manifesto

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

Agile Manifesto

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

Agile Manifesto

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

Agile Manifesto

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.

Twelve Principles

12 principles - delivery

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

12 principles - change

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

12 principles - frequency

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

12 principles - collaboration

4. Business people and developers must work together daily throughout the project.

12 principles - motivation

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

12 principles - dialog

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

12 principles - working software

7. Working software is the primary measure of progress.

12 principles - sustainability

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

12 principles - excellance

9. Continuous attention to technical excellence and good design enhances agility.

12 principles - simplicity

10. Simplicity–the art of maximizing the amount of work not done–is essential.

12 principles - self-organization

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

12 principles - reflection

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

Rules of eXtreme Programming

- Planning

Rules of eXtreme Programming

- Planning
- Managing

Rules of eXtreme Programming

- Planning
- Managing
- Designing

Rules of eXtreme Programming

- Planning
- Managing
- Designing
- Coding

Rules of eXtreme Programming

- Planning
- Managing
- Designing
- Coding
- Testing

XP - Planning

- User stories are written.

XP - Planning

- User stories are written.
- Release plan — Release planning creates the release schedule.

XP - Planning

- User stories are written.
- Release plan — Release planning creates the release schedule.
- Release often — Make frequent small releases.

XP - Planning

- User stories are written.
- Release plan — Release planning creates the release schedule.
- Release often — Make frequent small releases.
- Iterative — The project is divided into iterations.

XP - Planning

- User stories are written.
- Release plan — Release planning creates the release schedule.
- Release often — Make frequent small releases.
- Iterative — The project is divided into iterations.
- Iteration planning — Iteration planning starts each iteration.

XP — Managing

- Optimize last — Give the team a dedicated open work space.

XP — Managing

- Optimize last — Give the team a dedicated open work space.
- Steady pace — Set a sustainable pace.

XP — Managing

- Optimize last — Give the team a dedicated open work space.
- Steady pace — Set a sustainable pace.
- Stand-up meeting — A stand up meeting starts each day.

XP — Managing

- Optimize last — Give the team a dedicated open work space.
- Steady pace — Set a sustainable pace.
- Stand-up meeting — A stand up meeting starts each day.
- Project velocity — The Project Velocity is measured.

XP — Managing

- Optimize last — Give the team a dedicated open work space.
- Steady pace — Set a sustainable pace.
- Stand-up meeting — A stand up meeting starts each day.
- Project velocity — The Project Velocity is measured.
- Move people around — Move people around.

XP — Managing

- Optimize last — Give the team a dedicated open work space.
- Steady pace — Set a sustainable pace.
- Stand-up meeting — A stand up meeting starts each day.
- Project velocity — The Project Velocity is measured.
- Move people around — Move people around.
- Fix XP — Fix XP when it breaks.

XP — Designing

- Simplicity — Simplicity.

XP — Designing

- Simplicity — Simplicity.
- System Metaphor — Choose a system metaphor.

XP — Designing

- Simplicity — Simplicity.
- System Metaphor — Choose a system metaphor.
- CRC cards — Use CRC cards for design sessions.

XP — Designing

- Simplicity — Simplicity.
- System Metaphor — Choose a system metaphor.
- CRC cards — Use CRC cards for design sessions.
- Spike solution — Create spike solutions to reduce risk.

XP — Designing

- Simplicity — Simplicity.
- System Metaphor — Choose a system metaphor.
- CRC cards — Use CRC cards for design sessions.
- Spike solution — Create spike solutions to reduce risk.
- Nothing early — No functionality is added early.

XP — Designing

- Simplicity — Simplicity.
- System Metaphor — Choose a system metaphor.
- CRC cards — Use CRC cards for design sessions.
- Spike solution — Create spike solutions to reduce risk.
- Nothing early — No functionality is added early.
- Refactor — Refactor whenever and wherever possible.

XP — Coding

- Customer on-site — The customer is always available.

XP — Coding

- Customer on-site — The customer is always available.
- Coding standard — Code must be written to agreed standards.

XP — Coding

- Customer on-site — The customer is always available.
- Coding standard — Code must be written to agreed standards.
- Test Driven Development — Code the unit test first.

XP — Coding

- Customer on-site — The customer is always available.
- Coding standard — Code must be written to agreed standards.
- Test Driven Development — Code the unit test first.
- Pair programming — All production code is pair programmed.

XP — Coding

- Customer on-site — The customer is always available.
- Coding standard — Code must be written to agreed standards.
- Test Driven Development — Code the unit test first.
- Pair programming — All production code is pair programmed.
- Serial integration — Only one pair integrates code at a time.

XP — Coding

- Customer on-site — The customer is always available.
- Coding standard — Code must be written to agreed standards.
- Test Driven Development — Code the unit test first.
- Pair programming — All production code is pair programmed.
- Serial integration — Only one pair integrates code at a time.
- Continuous integration — Integrate often.

XP — Coding

- Customer on-site — The customer is always available.
- Coding standard — Code must be written to agreed standards.
- Test Driven Development — Code the unit test first.
- Pair programming — All production code is pair programmed.
- Serial integration — Only one pair integrates code at a time.
- Continuous integration — Integrate often.
- Continuous integration — Set up a dedicated integration computer.

XP — Coding

- Customer on-site — The customer is always available.
- Coding standard — Code must be written to agreed standards.
- Test Driven Development — Code the unit test first.
- Pair programming — All production code is pair programmed.
- Serial integration — Only one pair integrates code at a time.
- Continuous integration — Integrate often.
- Continuous integration — Set up a dedicated integration computer.
- Collective ownership — Use collective ownership.

XP — Testing

- Unit tests — All code must have unit tests.

XP — Testing

- Unit tests — All code must have unit tests.
- Unit tests — All code must pass all unit tests before it can be released.

XP — Testing

- Unit tests — All code must have unit tests.
- Unit tests — All code must pass all unit tests before it can be released.
- Tests — When a bug is found tests are created.

XP — Testing

- Unit tests — All code must have unit tests.
- Unit tests — All code must pass all unit tests before it can be released.
- Tests — When a bug is found tests are created.
- Acceptance tests — Acceptance tests are run often and the score is published.

Afterword

This just scratches the surface. Agile methodology has much, much more functionality than illustrated here. If you are interested in Agile, read the documentation for the details.