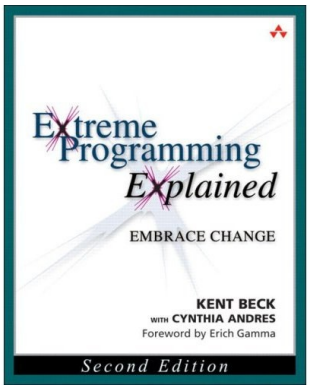
302

Extreme Programming (XP)

@rchatley #doc302

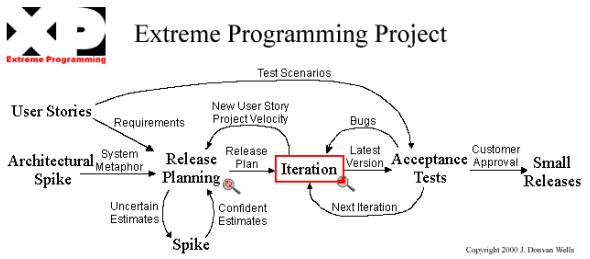
Here we give a brief introduction to Extreme Programming (XP). XP is an agile method which is used by many agile teams in industry, although it is probably not as prevalent as Scrum, as it is harder to do - but generally gets better results.



Kent Beck introduced XP in this book. The first edition is known as “the white book”, and this second edition “the green book”. In the second edition Beck updated his thinking, and how he presented XP, but the core messages remain the same.

XP is a distillation of a set of practices, principles and values that Beck noticed were present in a team that he was working on, where things seemed to be going well. For each thing that the team identified as working well, they tried to do more of that - they called it “turning up”. For example, test-driven development seemed to help them with building a quality product, so they tried to do more of that.

XP does not define any particular roles within the team, or any management structure. It does though say that the team should be cross-functional and have in it everyone needed to fully deliver a feature - this practice is called “whole team”. Don’t have a separate design team who you ask to do things for you, have a designer on the feature team. Don’t have a database team, have a database specialist on the feature team if you need one. XP also says that the team should “sit together” so that they can communicate easily, and close to the customer, so that they can discuss and clarify requirements without having to book a meeting or communicate over a narrowband channel like email.



http://www.extremeprogramming.org/rules.html

XP teams organise work into iterations, time boxed periods of work that help them to plan, review, and keep a rhythm to their work. Iterations might last a week, two weeks, three weeks, or (very occasionally) longer. However they do not necessarily batch work up for a big release at the end of the iteration - if it's finished, they can release any time. Also, they do not necessarily pick one theme for the iteration, they focus on doing the next most valuable things.

Iterations typically conclude with a retrospective, to see how the team has been doing, and if there is anything that could be improved in the way that they work.

Iteration planning

An iteration starts with a planning game, where the customer and the team discuss what are the highest priority stories for the next iteration, what seems likely to be achievable (to the point where it is released within the iteration), and what the high-level acceptance criteria for stories are. A team might also plan to do some technological research work, or some infrastructural work during the iteration, just to keep the velocity up and open up future options. If we don't have enough information to estimate a story, we can do a technological "spike" - that is a quick prototype that is designed to tell us enough about a proposed solution that we can estimate doing it properly. Spikes are not intended to be production code, so we might not pair on them, or use TDD.

Technical Practices

Test First Programming

Pair Programming

Continuous Integration

Incremental Design

Ten Minute Build

XP mandates a number of technical practices which it thinks are necessary to be able to deliver quality software and release regularly. Also sometimes picked out as a separate practice is refactoring - a constituent part of coming up with a design incrementally and following test-driven development. XP teams consider refactoring an important tool to improve the design of the codebase continually, making it easier to make future changes.

Another notable XP practice is pair programming, where two developers work together on a task at one computer. This gives continuous design and code review, and shares knowledge, so it is unlikely that there is only one person on the team who knows a certain bit of the system, or a certain technology. Having specialists can be a risk, especially if that person is not around when a problem arises.