XP

Extreme programming

EXTREME PROGRAMMING
THE TWO OF YOU WILL BE A CODE-WRITING TEAM.
scottadams@aol.com
www.dilbert.com
STUDIES PROVE THAT TWO PROGRAMMERS ON ONE COMPUTER IS THE MOST PRODUCTIVE ARRANGEMENT.
© 2002 United Feature Syndicate, Inc.
SOMETIMES I CAN WHISTLE THROUGH BOTH NOSTRILS. I'VE SAVED A FORTUNE IN HARMONICAS.

CONTENT

- Why
- History
- What
  - Values
  - Principles
  - Practices

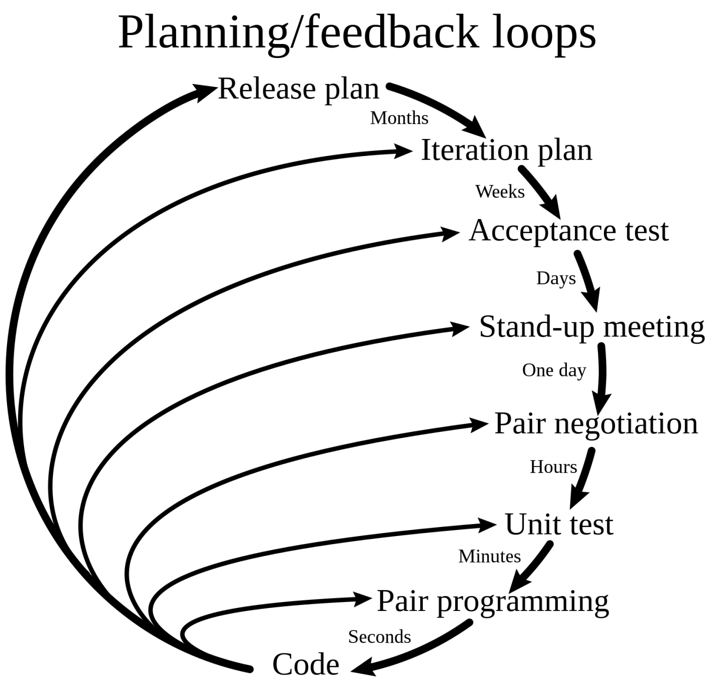
WHY

Better quality in software

…and responsiveness to changing requirements

By

- frequent releases
- improved productivity



https://en.wikipedia.org/wiki/Extreme_programming

HISTORY

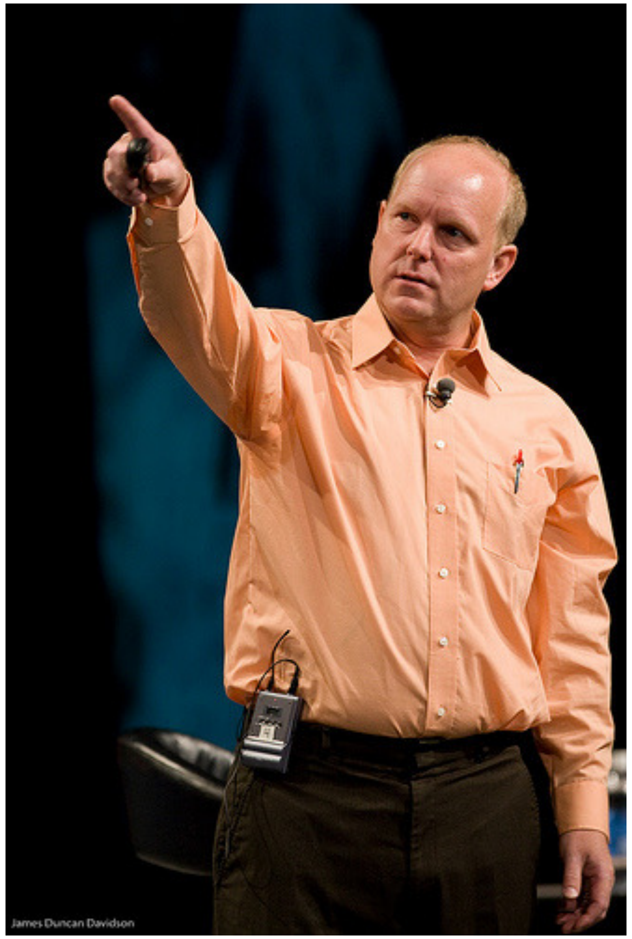
Was developed by Kent Beck while working on Chryslers payroll system (C3)

As described in *Extreme Programming explained* from 1999

Starting the Agile revolution that led to the Agile Manifesto

Introduces a number of

- Values
- Principles
- Practices



WHAT

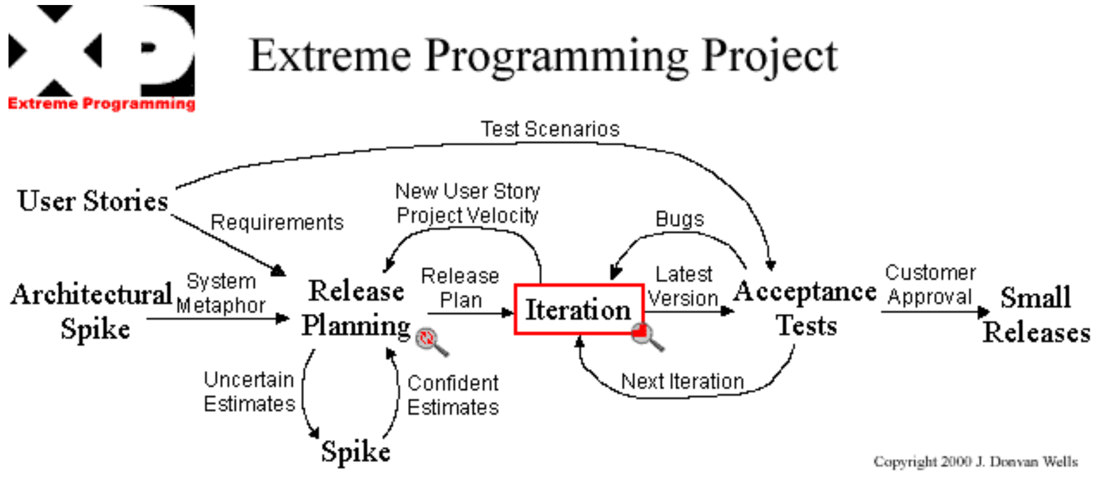
Lightweight agile process

- Mostly for small-to-medium sized teams

- Social change
  - increases collaboration

- Focus on creating value for the customer
  - by delivering often
  - improving quality
  - eliminating defects

- 12 key practices taken to their extreme



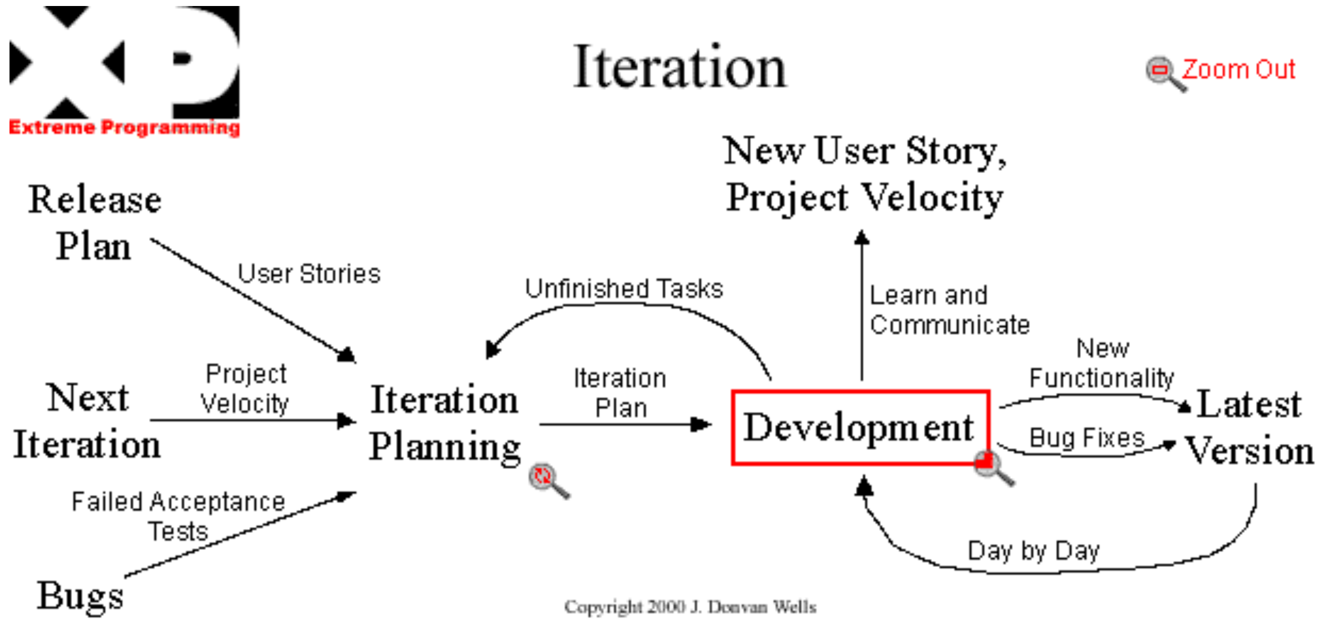
Source: http://www.extremeprogramming.org/

XP PARADIGM

- Stay aware
- Adapt
- Change

Change will happen

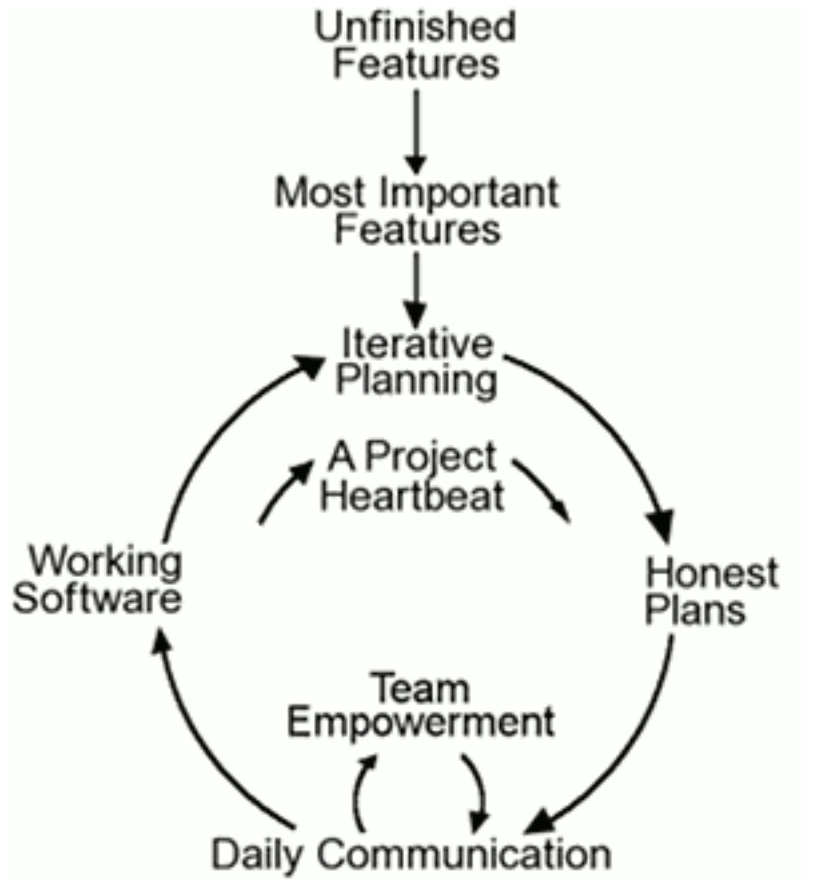
XP lets you adapt



Source: http://www.extremeprogramming.org/

5 VALUES

- Communication
  - building software requires good communication. customers + colleagues
- Simplicity
  - start with the simplest solution (YAGNI)
- Feedback
  - from system, customer, and team
- Courage
  - Speak the truth, seek answers. To adapt
- Respect (added in the second edition)
  - team members and project



Source: http://www.extremeprogramming.org/

PRINCIPLES

Principles that form a basis of XP – based on the values.
Better idea of what the practices are intended to accomplish

- Humanity
- Economics
- Mutual benefit
- Self-similarity
- Improvement
- Diversity
- Reflection
- Flow
- Opportunity
- Redundancy
- Faillure
- Quality
- Baby steps
- Accepted responsibity

PRACTICES

The day-to-day things

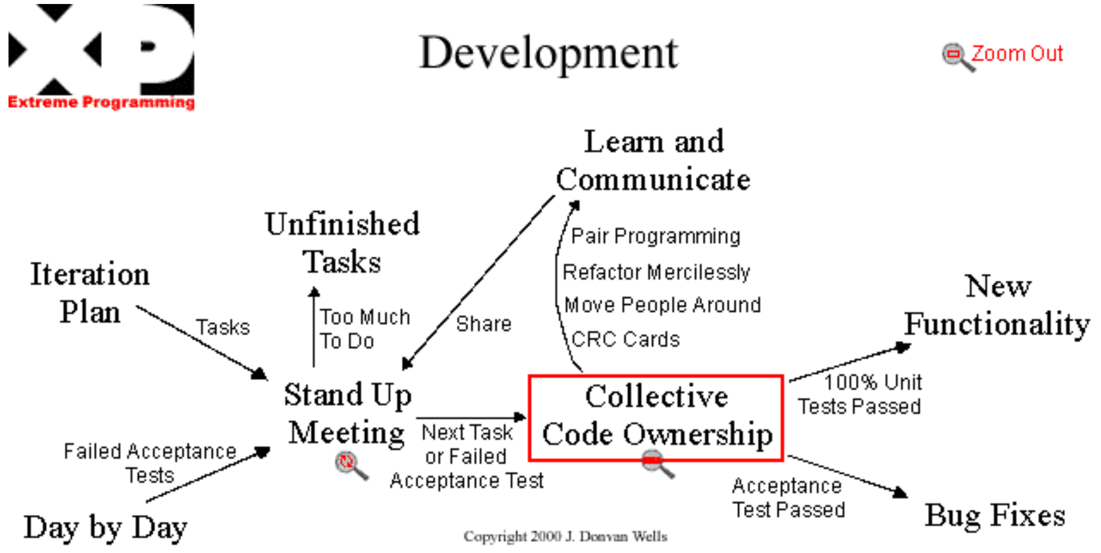
- Sit together
- Whole team
- Informative workspace
- Energized work

- Pair programming
- User Stories
- Weekly cycle
- Quartable cycle
- Slack
- 10 min build
- Continuous integration
- Test first
- Incremental design

RULES

Managing:

- Open workspace
- Sustainable pace
- Stand up meetings
- Velocity
- Move people
- FIX XP

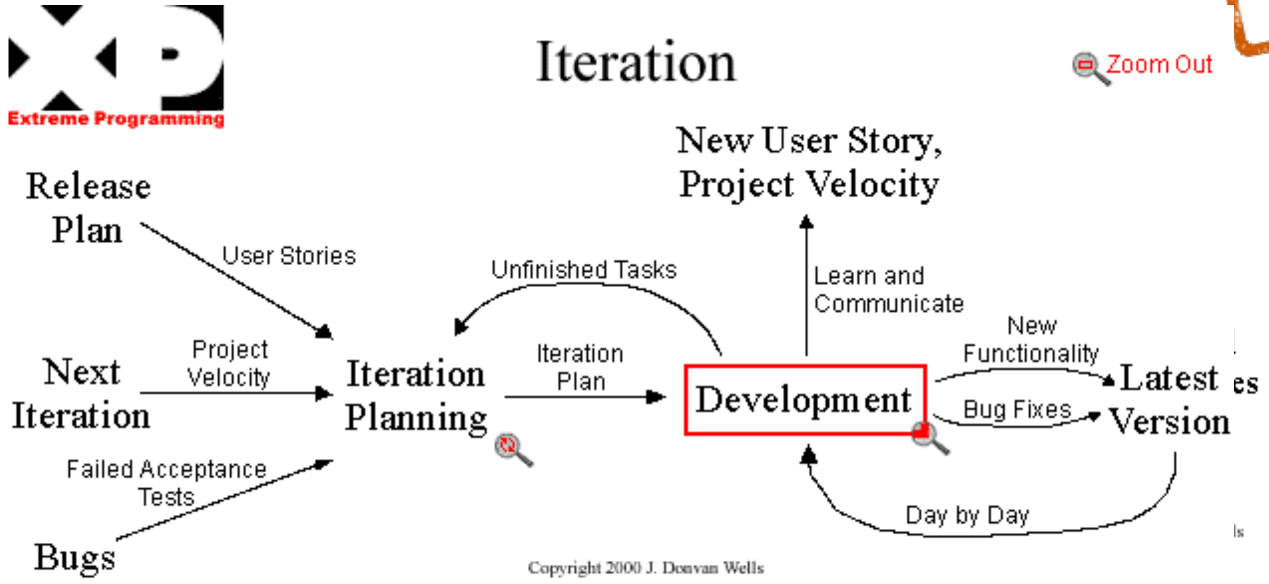


Source: http://www.extremeprogramming.org/

RULES

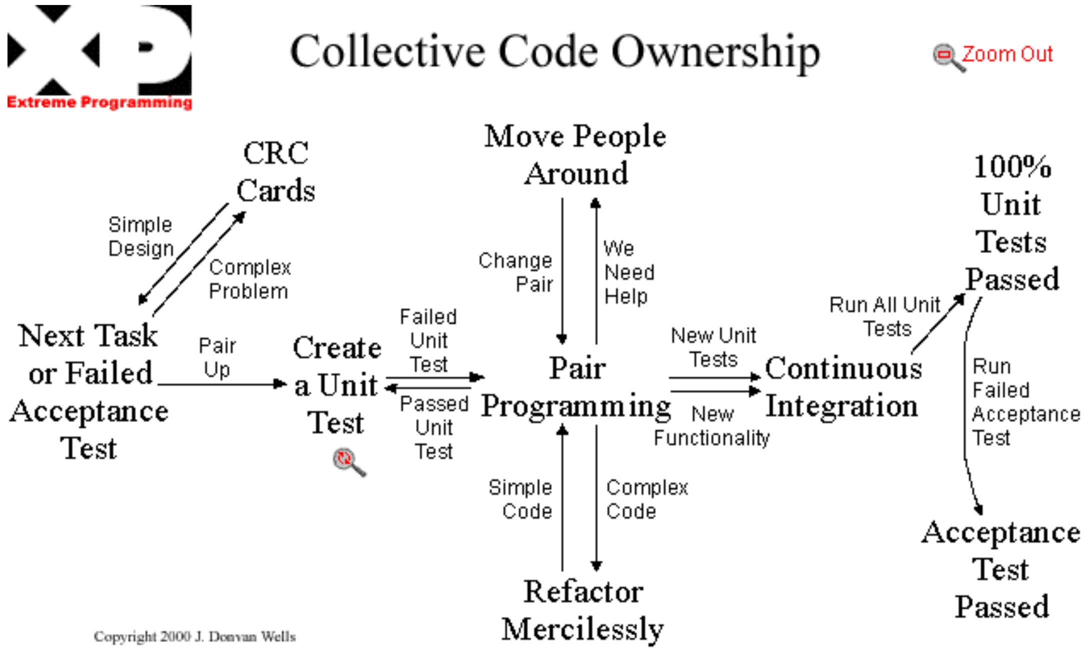
Planning:

- User stories
- Release planning
- Frequent releases
- Iterative
- Iteration planning



Source: http://www.extremeprogramming.org/

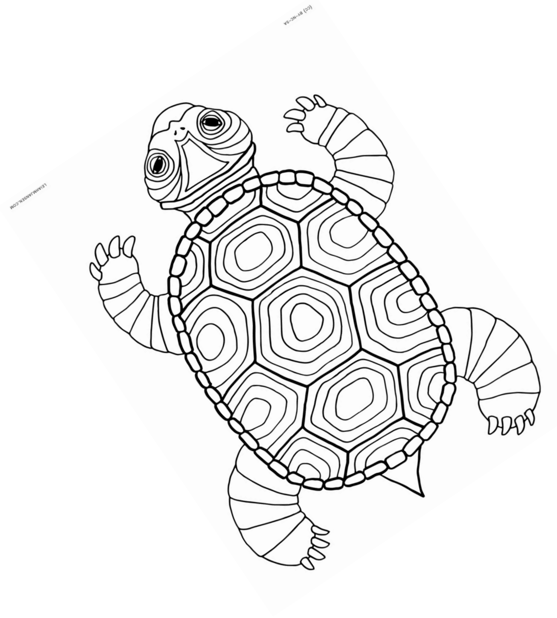
RULES



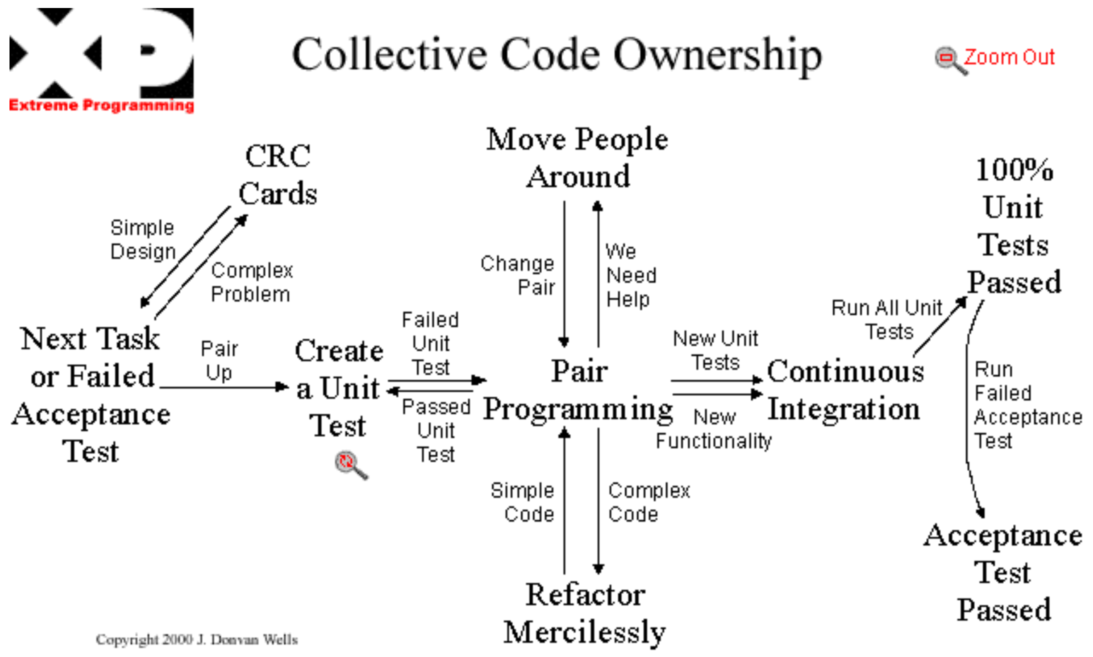
Source: http://www.extremeprogramming.org/

Designing:

- Simplicity
- System metaphor
- CRC Cards
- Spike solutions
- No functionality added early
- Refactor



RULES

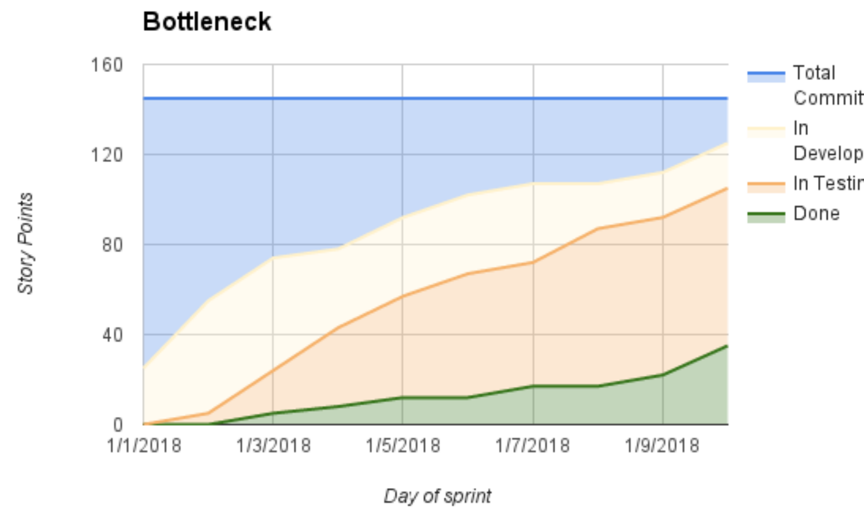


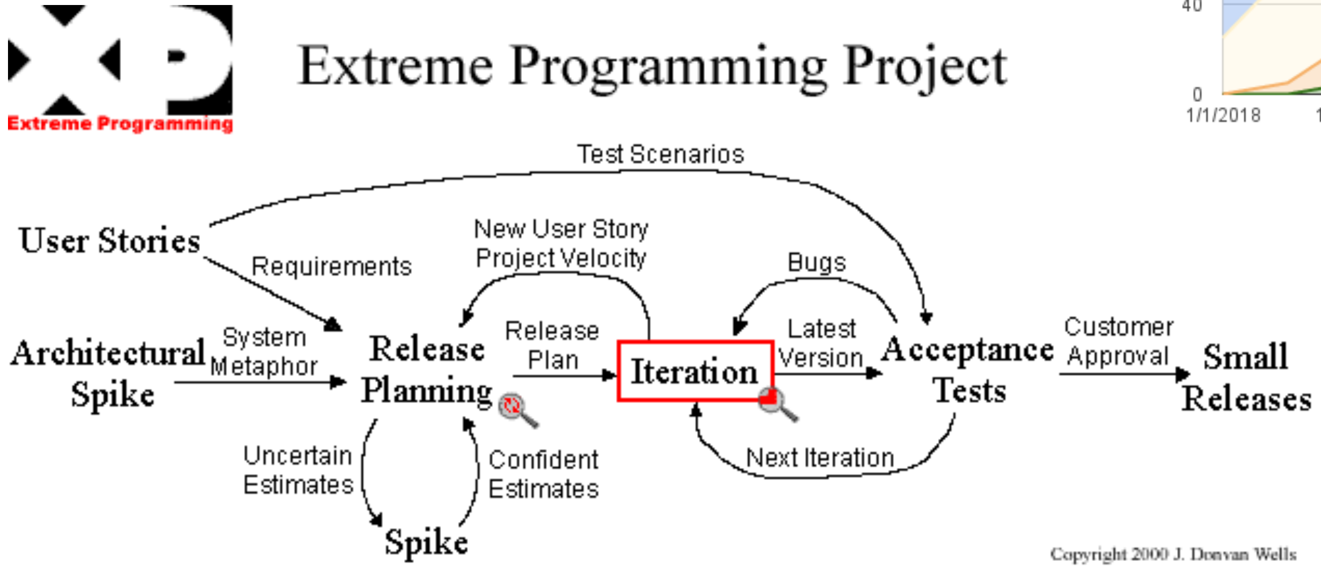
Source: http://www.extremeprogramming.org/

Coding:

- Customer available
- Code standards
- TDD
- Pair program
- Sequential code integration
- Integrate often
- CI environment
- Collective ownership

RULES





Source: http://www.extremeprogramming.org/

Testing:

- Test all code
- Pass before release
- Prove bug by unit-test
  - Then fix
- Acceptance are run often

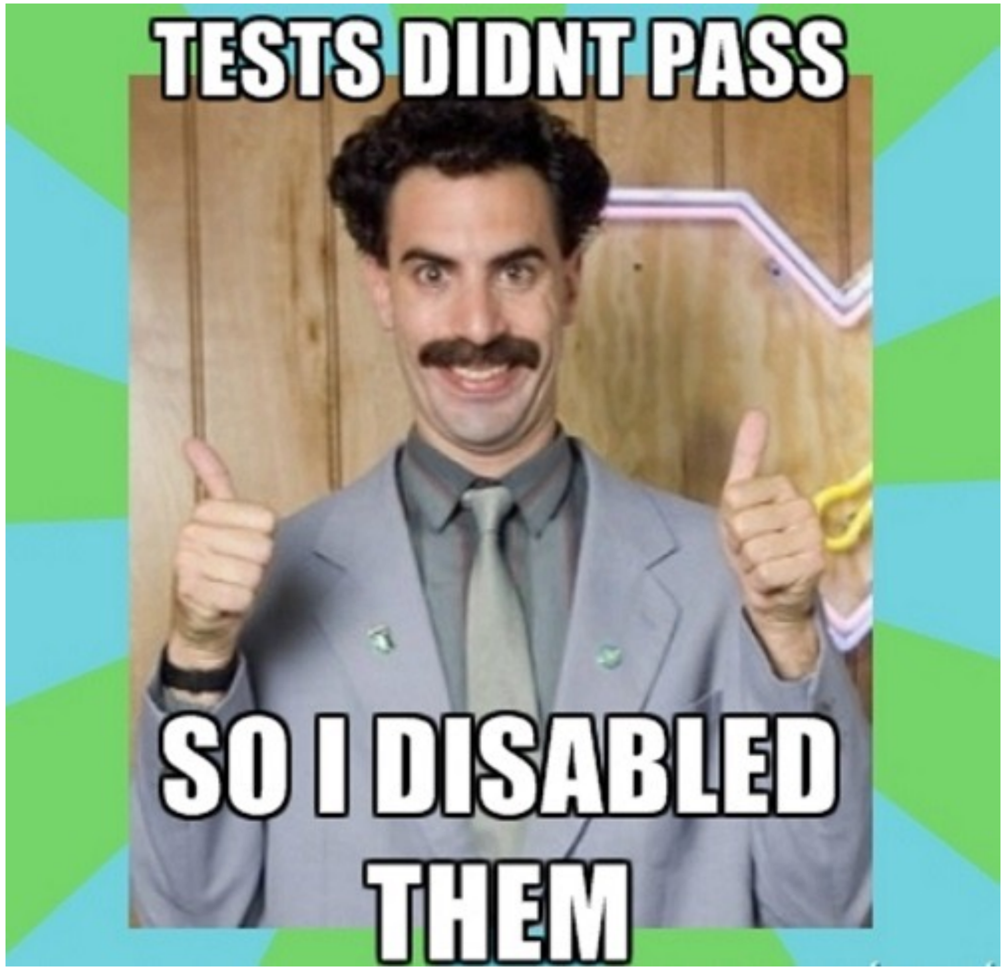
STARTING WITH XP

New project

- User stories
  - 1-3 weeks
- Spike solution
  - Risk mitigation
- Release planning
  - Invite the whole team
- Begin iterative development

Now you have started

STARTING WITH XP



Existing project

- What is slowing the project down
- Start by fixing this

Etc.

- Many bugs ☞ automated acceptance tests
- Requirement specifications ☞ start with user stories
- One/two developers are bottlenecks ☞ collective code ownership

AARHUS
UNIVERSITY

REFERENCES

- https://xkcd.com/2166/