Extreme Programming (XP)

Extreme Programming

Extreme Programming

Skills and Practices, a software product manager will use.

why is it called Extreme Programming?

XP takes practices used by many other methodologies and takes them to an extreme level.

Extreme Programming & client satisfaction

Client satisfaction

Extreme Programming is all about client satisfaction: the job of your development team » to make the client as happy as possible

Extreme Programming can help in the cases:

- what if you get a grumpy client?
- a client who is not willing to listen and collaborate?

Extreme Programming & Agile Manifesto

Extreme Programming also values:

(principles of Agile Manifesto)

- constantly delivering software
- responding to change
- effective teamwork
- self-organization

Five aspects of development that Extreme Programming focuses on improving:

- Communication
- Simplicity
- Feedback
- Respect
- Courage

Extreme Programming & Agile Manifesto

Aspects covered in the Manifesto:

- Communication
- Simplicity
- Feedback

Respect and Courage

Respect and Courage

Making a great software product depends heavily on all of the people involved.

In Extreme Programming, everyone is considered equal / on the same level

- client
- software product manager
- development team

Respect

Each individual is respected and valued for what they bring

- business experience
- application domain expertise
- knowledge of agile practices
- development skills
- enthusiasm
- ...

Courage

The extreme programming website states:

- "We will tell the truth about progress and estimates."
- "We don't document excuses for failure because we plan to succeed."
- "We don't fear anything because no one ever works alone."
- "We will adapt to changes whenever they happen."

Context

You are a software product manager for a small startup that is working on creating an application that matches parents with local babysitters and nannies.
You and your development team decided early on that you will adopt the Extreme Programming methodology. Your development team is organized into pairs, with each pair working in front of a single workstation.

Which of the five aspects of Extreme Programming do you think that this improves?

- A. Communication.
- B. Simplicity.
- C. Feedback.
- D. Respect.
- E. Courage.

Quiz - A

Which of the five aspects of Extreme Programming do you think that this improves?

- ✓ A. Communication.
- ✓ B. Simplicity.
- ✓ C. Feedback.
- ✓ D. Respect.
- ✓ E. Courage.

pair programming

Pair programming

two programmers communicate and work collaboratively through problems

"Pair programming" involves all aspects of development

benefits of pair programming & having two perspectives

- simpler and higher quality code
- immediate feedback while working directly with someone

pair programming & respect & courage

- encourages respect since you can use the complimentary qualities and skills of the programmers to solve the problem.
- It improves courage since a programmer does not work alone.

Extreme Programming & rules

Extreme Programming is organized into a set of rules.

Traditional XP

a set of code centric rules for development

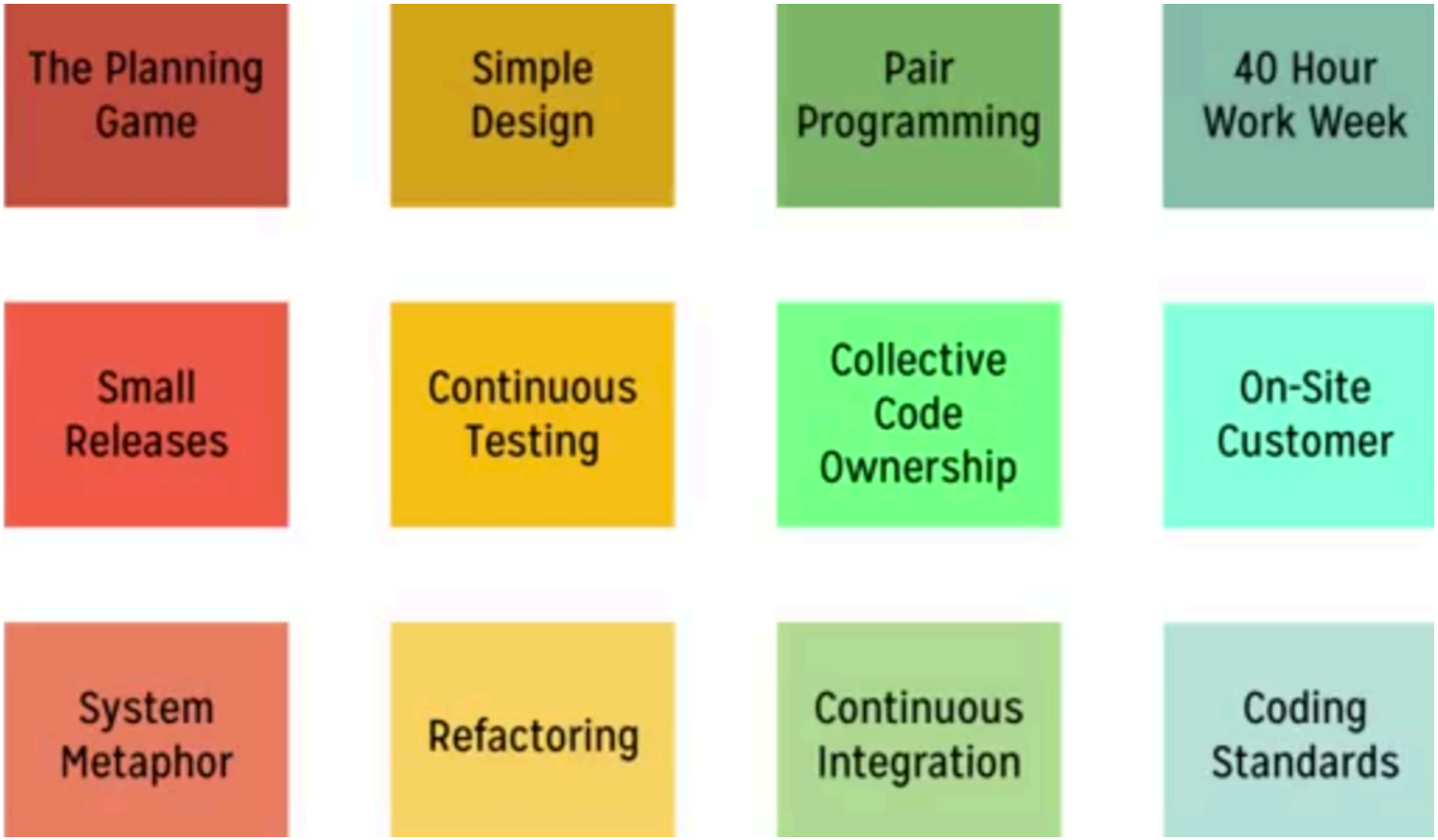
Modern XP

- focus much on management.
- Extreme Programming has implemented many of the management rules from Scrum.
- » Scrum now can almost be seen as a subset of XP.

Extreme Programming – defined by 12 basic principles

XP rules & jigsaw puzzle

- single rule vs the big picuture
- Extreme Programming – defined by 12 basic principles



Practice 1: The Planning Game

planning

(the client and the development team)

- it is to plan the product.
- The initial session determines how often to see a release of a working product.

Examples: planning at different scales

- a large planning session at the beginning of development
- smaller ones after each iteration

planning rules: (regardless of the scale)

1. features & user stories

(the client and development team)

- come up with a list of new features for the product.
- Each of these features is expressed as a user story.
  (a user story describes a feature from a user point of view)

2. estimates

(the development team)

- creates estimates for how much effort they can allot in the next iteration
- how much effort each user story will take to complete

3. prioritizing features & when to release

(the client and development team)

- prioritizes the stories
- decide when to release a working version of each feature

Practice 2: Small Releases 1/2

small & frequent releases

With Extreme Programming you want releases to be **as small and as frequent as possible**.

features make good business sense & early releases

- You development team will plan with the client which features **make good business sense** and can be relived early in the project.
- You want to **develop important features early** so you have more time to refine them.

Practice 2: Small Releases 2/2

balance

The development team and the client need to find the right **balance**

- between **what the client wants to complete**.
- and **what can be developed early**.

reason for small releases

You want releases to be as small as possible
» so that it's easy to make estimations. (It's easier to look at a week or two ahead, rather than a month.)

Practice 3: System Metaphor

to make your system easier to explain to others

should be able to explain your product easily to a third party, who has **no technical experience**.

an application description by the client

- a face recognition application
- it can tell the name and information about a person.

example metaphor

"The application is like a popular person at a party.
Who knows everyone there, and everything about them."

a good metaphor?

Given the metaphor but not the explanation of the application, people can determine what the app does

Quiz - Q

Context

- Erica is working as a software product manager for a development team that is creating a social media application.
- Her team spends two weeks completely planning out every detail of the project.
- Including the design and schedules.
- She measures the iteration in hours, not weeks.
- Her team System Metaphor is, it's like Twitter, but with longer post.

Which of these practices are considered extreme?

- A. Two week planning period.
- B. Iteration measured in hours.
- C. System metaphor.

Quiz - A

Which of these practices are considered extreme?

- A. Two week planning period.
- ✓ B. Iteration measured in hours.
- C. System metaphor.

explanation

- In Extreme Programming, you want iterations to be as small as possible while still delivering a working product.
- Erica's System Metaphor is more of a comparison than a metaphor.
- planning in Extreme Programming should be very simple, not two week.

Practice 4: Simple design

"Designs should be as simple as they can be."

Don't over engineer your design for a future that may not come.

design & changes

- Your requirements are going to be constantly changing.
- So there's no point in making detailed designs for something that's not going to stick.
- Design what you need to make the requirements you're working on today.

Practice 5: Continuous Testing

Extreme programming & Test Driven Development

- Test Driven Development, TDD: tests are written for a feature or a requirement before the source code is written.
- The idea is to use these tests as executable forms of requirements.
- When a test passes, the corresponding requirement is satisfied.

Extreme programming & extreme testing

- Each requirement should have tests, and each test should correspond to some requirement.
- Tests should also be simple.
  - » If a test was hard to write, it's a sign to rework the code.

Extreme programming & testing

Automating the tests and having simple tests, will allow testing to be done **continuously**.

Extreme programming & two ways of testing in XP

1. acceptance tests & feature

- for the client, to test that each feature of the overall product works as specified.
- These tests cover large parts of the product.
- can be automated or can be **a script of user interface actions** that a human follows.

2. unit tests & lower level functionality

automated tests written by developers to test lower level functionality.

Practice 6: Refactoring

Refactoring

"Restructuring the design of the code without changing the behavior of the code."

Refactoring & changes

- Refactoring encourages responding to new changes.
- It improves the design, so that new features can be added easily.

example: refactoring & removing duplicate and unnecessary code

- Because developers created the unit test, they can confidently remove the code without breaking the product.
- If the test still pass after deletion, it was unnecessary code.
- » extreme refactoring: if the code is the simplest thing that could possibly work.

Quiz

Context

You are a software product manager working with a development team who is producing an app for a cinema that allows users to check show times and buy tickets.

Which of the following, if any, would be acceptance tests?

A. A test that confirms when a user buys a ticket, a valid ticket appears.

B. A test that confirms when a user creates an account, their information is added to the database.

C. A test that confirms when a user logs in, that their log-in information matches an account in the database And/or

D. A test that confirms when a user selects a show time, movies playing at that time appear.

Quiz – A

Context

You are a software product manager working with a development team who is producing an app for a cinema that allows users to check show times and buy tickets.

Which of the following, if any, would be acceptance tests?

- ✓ A. A test that confirms when a user buys a ticket, a valid ticket appears.
- B. A test that confirms when a user creates an account, their information is added to the database.
- C. A test that confirms when a user logs in, that their log-in information matches an account in the database And/or
- ✓ D. A test that confirms when a user selects a show time, movies playing at that time appear.

Practice 7: Pair Programming

[recall] Pair Programming

"Two developers work side by side at one computer to develop code."

extrem code review

- Pair programming takes code review to the extreme.
- Instead of periodically reviewing code, code is reviewed all the time by the other person in the pair.

benefit: innovation & high quality products & fostering courage

- doubling the human resources for the feature?
- produce more than two programmers working separately and independently & at a higher quality.
- Programmers are more likely to take some calculated risks because they're not doing it alone.

Practice 8: Collective Code Ownership

"Anyone can add code to any part of the product."

Encourage the entire development team to contribute new ideas to any part of the product.

- When things go wrong it is the result of the team, not individuals.
- Similarly, when things are a success, it is the team's success.

Practice 9: continuous integration

"developers combine their code often."

- at least once daily, but to be considered extreme, it could happen a lot more frequently.
- All tests should pass 100% before and after the integration.

Practice 10: 40 hour work week

work-life balance?

- It could seem obvious to be respectful of a developer's work-life balance.
- But if you get a team of passionate developers, it may be hard to pull them away from the keyboard.
- In crunch time, XP allows **up to one week of overtime**.

multiple weeks of overtime

If multiple weeks of overtime occurs that's a red flag that something is wrong with the **management** of the project.

Practice 11: on site customer

invites the client to be part of the development team.

Clients are

- always around to clarify and answer questions about anything that might come up.
- involved in every stage of development.

Practice 12: coding standards

"All developers code to the same standards."

- The development team follow common coding conventions and format their code the same way.
- This standard should be agreed upon at the beginning of development.
- » People should not be able to look at code and tell who wrote it.
- » Makes it easy to read and encourages collective ownership

context

- **Danielle** is the product manager for a five person development team creating a payroll application that allows small business owners to pay their employees.
- After a prototype delivery, the client sends out a nasty email to the development team, saying that they discovered a bug in the application that calculates the taxes incorrectly.
- This bug would cost the business owners a great deal of money, and employees would end up getting more money than they have earned.
- **Timmy** was the programmer who wrote the unit test for this feature.
- **Steven** was the programmer who wrote the source code.
- **Danielle**, the product manager, tested the product before the release.

The client is very upset.

Who's to blame for this error? Choose all that apply.

A. Danielle, B. Timmy, C. Steven
D. the other three developers on the team.

Quiz – A

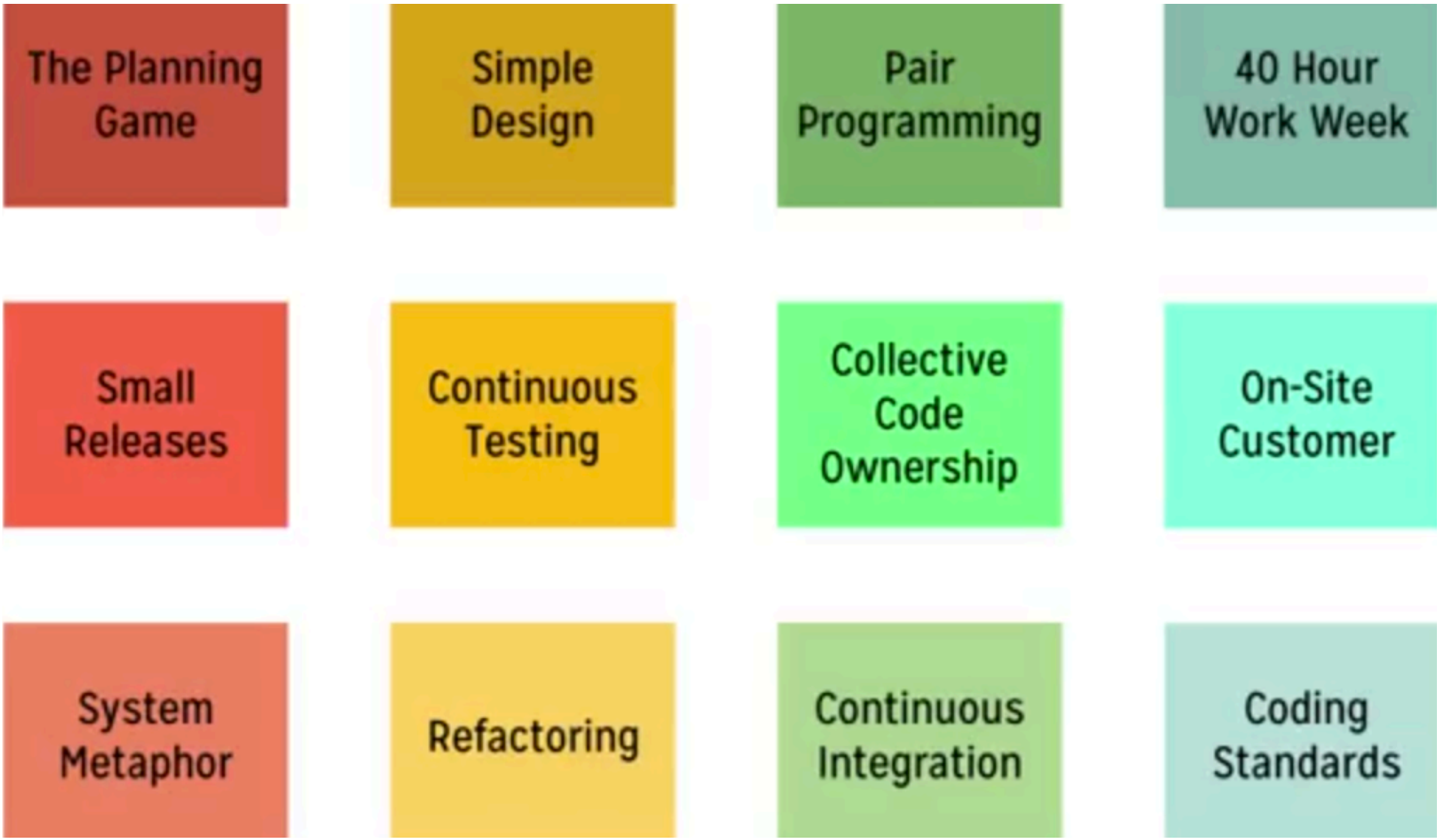
Who's to blame for this error? Choose all that apply.

- ✓ A. Danielle,
- ✓ B. Timmy,
- ✓ C. Steven,
- ✓ D. the other three developers on the team.

extreme programming & collective ownership

- All members of the team are responsible for errors that occur.
- Had the client been very satisfied with the product, the entire team would get credit for the success.

Extreme Programming – defined by 12 basic principles / practices



Extreme programming

Extreme programming may be expressed differently

- traditionally described by 12 basic practices
- by 29 Extreme programming rules – Don Wells, author of the first version of Extreme programming rules

Other management practices

There are however a few management practices that do not fit into either of those categories.

Practice 1/3: “Give a team a dedicated open work space”

A dedicated work space would enhance productivity and teamwork.

example

- XP suggests that the workspace have multiple computers that belong to no one.
  They should be located in the middle of the room and not along the walls.
- » to encourage the development team to work together.
- a meeting table with white boards
- » for collaboration in team and brain storming

Practice 2/3: move people around

moving people around

to encourage communication and flexible work relationships.

- Everyone knows how to develop and operate all parts of the product.
- The success of a project should not depend on one person.
- the most useful team members are the ones who are good at multiple skills
- moving people around vs heavily document features

Practice 3/3: fix XP when it breaks

change it when something does not work

- **when** it breaks (not if breaks)
- It is probabilily going to break.
- It's not going to run smoothly throughout the **entire life cycle** of your product.
- Not all of the practices of XP will work well with your team.
- follow the rules until they break
- determine what's working and what's not in retrospective meetings

Quiz

What are some areas of development that you are going to need to change in order to have Extreme Programming running precisely in this office?

A. The Workspace
B. Pair Programming
C. Client Availability
D. small frequent releases
E. developer versatility

- You have been hired by a company that say they have tried to implement the Extreme programming methodology exactly. You walk into the workplace and see a maze of cubicles. As you walk around the work space and peek into cubicles, you see pairs of programmers working together.
- You pop into a cubicle and ask the developers where they can find the client. One developer hands you a business card that has the client's phone number on it. He says you can leave a message there and the client will get back to you.
- You go into the next cubicle and ask the two programmers working there when the next release is due. They tell you that the releases are due every second Friday. They have a release coming up this Friday.
- You ask these programmers what they're working on. They explain that they are the database team, and that they exclusively work on maintaining and building the database.
- You realize that there are many areas of Extreme programming that they are not following properly.

Quiz

What are some areas of development that you are going to need to change in order to have Extreme Programming running precisely in this office?

A. The Workspace
B. Pair Programming
C. Client Availability
D. small frequent releases
E. developer versatility

Quiz

What are some areas of development that you are going to need to change in order to have Extreme Programming running precisely in this office?

- ✓ A. The Workspace?
- B. Pair Programming?
- ✓ C. Client Availability?
- D. small frequent releases
- ✓ E. developer versatility?

downfalls of XP

All or Nothing Approach

In adopting XP, you are encouraged to adopt all the rules of XP to get the full benefits.

- The individual practices do not have as much value as the overall methodology.
- But sometimes, implementing all of the practices is just not practical or possible.

designed for small development teams

What happens when you have over 20 people on your development team? You are going to run into many issues with **collective ownership** and **integration**.

downfalls of XP

a lack of up front planning

Compared to other methodologies, XP does not encourage real up front software architecture planning.
» can lead to a lot of reworking in the future.

pair programming

» Personality conflicts?

difficult to arrange an on-site client

It's uncommon that a client be made available for the time required by XP.

consider other methodologies?

Extreme programming outlines many great practices for software development (but may not work for your team).