AGILE METHODS – EXTREME PROGRAMMING

Dr. Enda Barrett

OÉ Gaillimh
NUI Galway

Overview

- **XP**
  - **General concepts**
- Specific XP concepts
  - Pair programming
  - Test Driven Development
  - Continuous Integration

Scrum Summary

- Scrum is a **project management** methodology

Characteristics of Scrum

- Self-organizing teams
  - No need for project manager (in-theory)
- Product progresses in a series of month-long "sprints"…could be biweekly also
- Assumes that the software cannot be well defined and requirements will change frequently
- Requirements are captured as items in a list of "product backlog"
- No specific engineering practices prescribed
  - XP, TDD, FDD…
- Best approach is to start with Scrum and then invent your own version using XP, TDD, FDD

Daily Scrum/Standup

- Parameters
  - Daily
  - 15-minutes
  - Stand-up
  - Not for problem solving
  - Only team members, ScrumMaster, Product Owners should talk
  - Should help to avoid additional unnecessary meetings
  - Commitment in front of peers to complete tasks

Answer three questions

1 What did you do yesterday?

2 What will you do today?

3 Is anything in your way?

Daily SCRUM/Standup

- Is NOT a problem solving session
- Is NOT a way to collect information about WHO is behind the schedule
- Is a meeting in which team members make commitments to each other and to the Scrum Master
- Is a good way for a Scrum Master to track the progress of the team

Scrum Framework

Roles

- Product owner
- ScrumMaster
- Team

Ceremonies

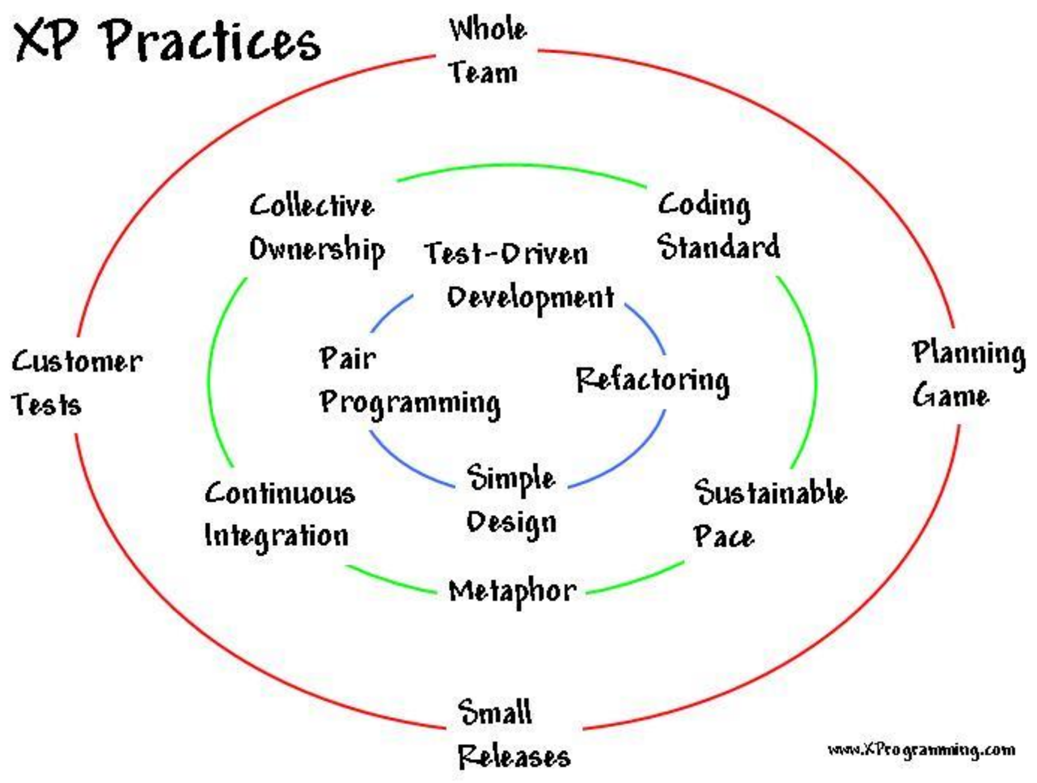
- Sprints
- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

We need an Agile Development method

- eXtreme Programming (XP)
  - One of the most popular agile **software development** methods

eXtreme Programming

- Pair programming
- Refactoring
- Test Driven Development
- Continuous Integration
- Metaphor
- Small releases
- Simple Design
- Customer tests

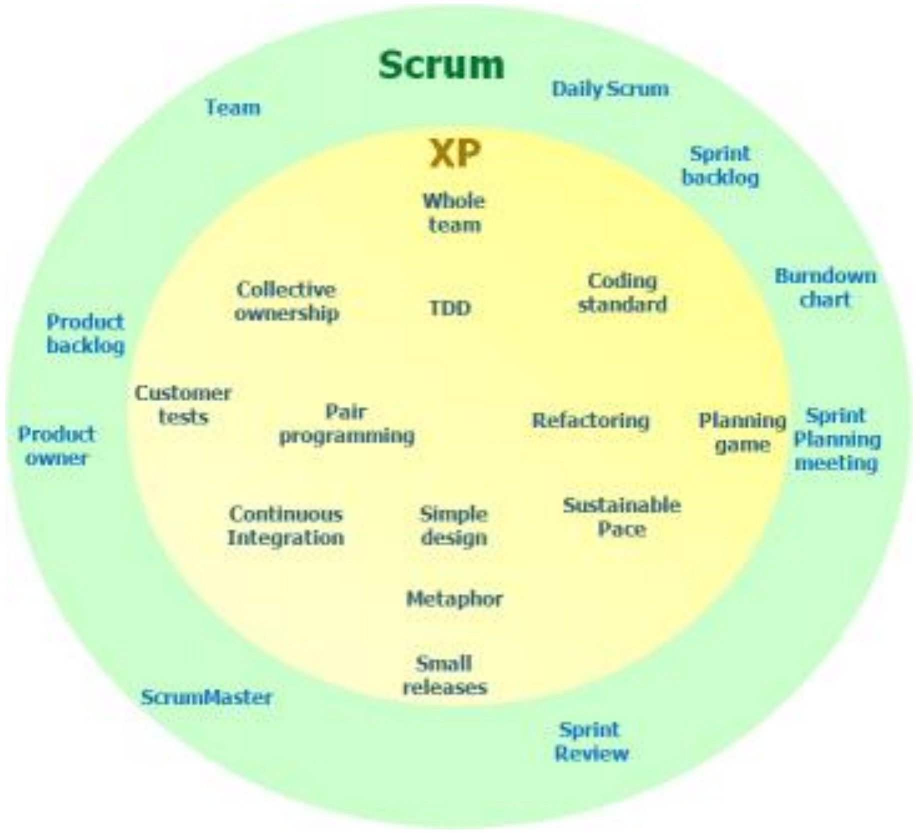


Complete Agile Process

Scrum + XP

Overview

How Scrum and XP can work together



Overview

- **XP**
  - **General concepts**

- Specific XP concepts
  - Pair programming
  - Test Driven Development
  - Continuous Integration

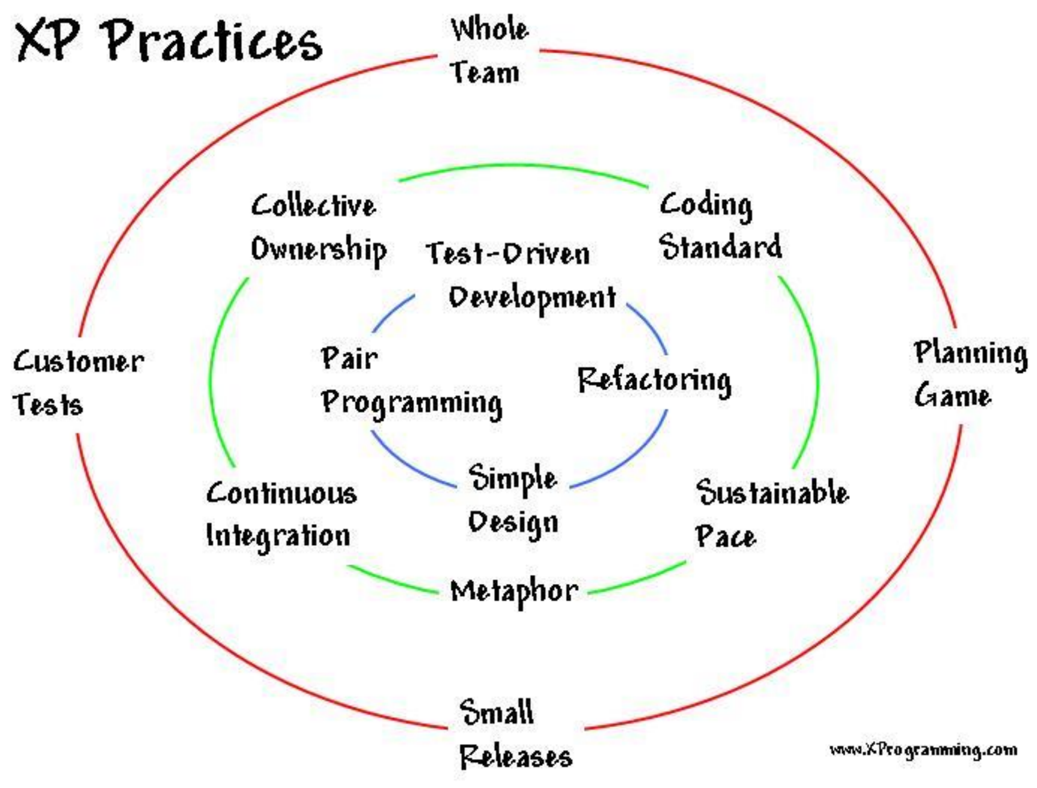
Principles of XP

Kent Beck

- **Communication**
  - Software development is inherently a team sport that relies on communication to transfer knowledge from one team member to everyone else on the team. XP stresses the importance of the appropriate kind of communication – face to face discussion with the aid of a white board or other drawing mechanism.
- **Simplicity**
  - Simplicity means "what is the simplest thing that will work?" The purpose of this is to avoid waste and do only absolutely necessary things such as keep the design of the system as simple as possible so that it is easier to maintain, support, and revise. Simplicity also means address only the requirements that you know about; don't try to predict the future.
- **Feedback**
  - Through constant feedback about their previous efforts, teams can identify areas for improvement and revise their practices. Feedback also supports simple design. Your team builds something, gathers feedback on your design and implementation, and then adjust your product going forward.
- **Courage**
  - Kent Beck defined courage as "effective action in the face of fear" (Extreme Programming Explained P. 20). This definition shows a preference for action based on other principles so that the results aren't harmful to the team. You need courage to raise organizational issues that reduce your team's effectiveness. You need courage to stop doing something that doesn't work and try something else. You need courage to accept and act on feedback, even when it's difficult to accept.
- **Respect**
  - The members of your team need to respect each other in order to communicate with each other, provide and accept feedback that honors your relationship, and to work together to identify simple designs and solutions.

Source:https://www.agilealliance.org/glossary/xp

Practices of XP



Further Reading and Descriptions

http://ronjeffries.com/xprog/what-is-extreme-programming/.

Whole Team

- All the contributors to an XP project sit together, members of one team. This team must include a business representative (Product Owner) – the "Customer" – who provides the requirements, sets the priorities, and steers the project.

https://ronjeffries.com/xprog/what-is-extreme-programming/#whole

Planning Game

- XP planning addresses two key questions in software development: predicting what will be accomplished by the due date, and determining what to do next.
- *Release Planning* is a practice where the Customer presents the desired features to the programmers, and the programmers estimate their difficulty.
- *Iteration Planning* is the practice whereby the team is given direction every couple of weeks. (Sprints)

https://ronjeffries.com/xprog/what-is-extreme-programming

Customer Tests

- As part of presenting each desired feature, the XP Customer defines one or more automated acceptance tests to show that the feature is working. The team builds these tests and uses them to prove to themselves, and to the customer, that the feature is implemented correctly.

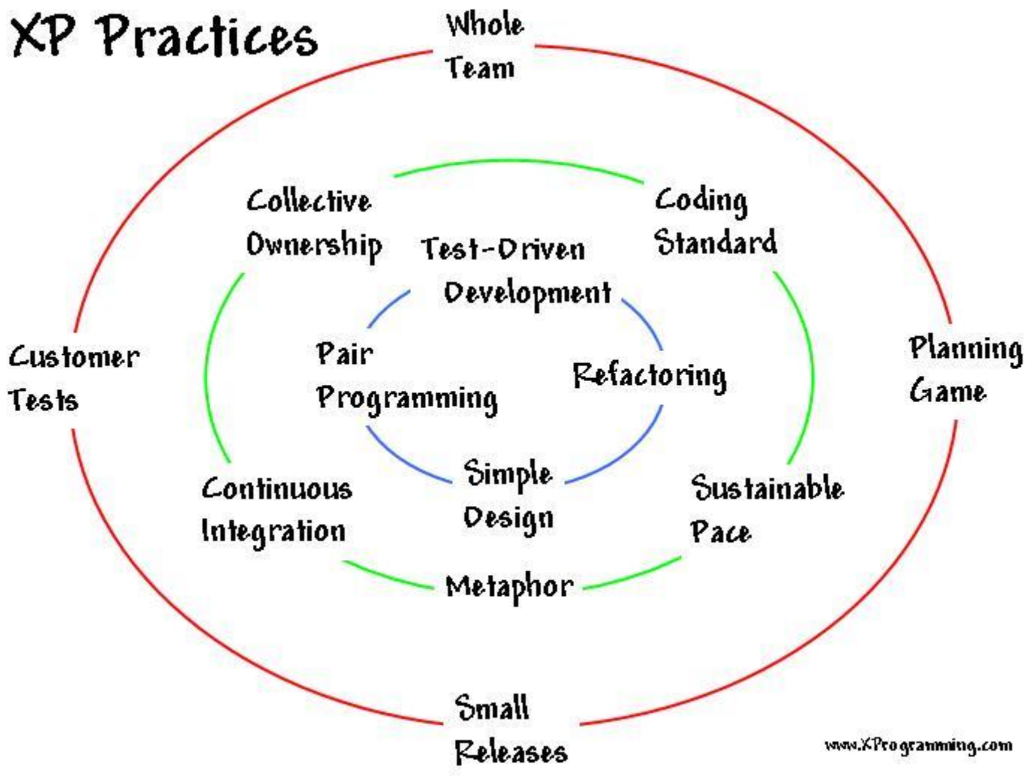
https://ronjeffries.com/xprog/what-is-extreme-programming/#whole

Small Releases

- XP teams practice small releases in two important ways:
  - First, the team releases running, tested software, delivering business value chosen by the Customer, every iteration.
  - Second, XP teams release to their end users frequently as well.

https://ronjeffries.com/xprog/what-is-extreme-programming/#whole

Practices of XP



Further Reading and Descriptions

http://ronjeffries.com/xprog/what-is-extreme-programming/.

Coding standards

- XP teams follow a common coding standard, so that all the code in the system looks as if it was written by a single – very competent – individual. The specifics of the standard are not important: what is important is that all the code looks familiar, in support of collective ownership.

http://ronjeffries.com/xprog/what-is-extreme-programming/.

Collective Ownership

- On an Extreme Programming project, any pair of programmers can improve any code at any time. This means that all code gets the benefit of many people's attention, which increases code quality and reduces defects.

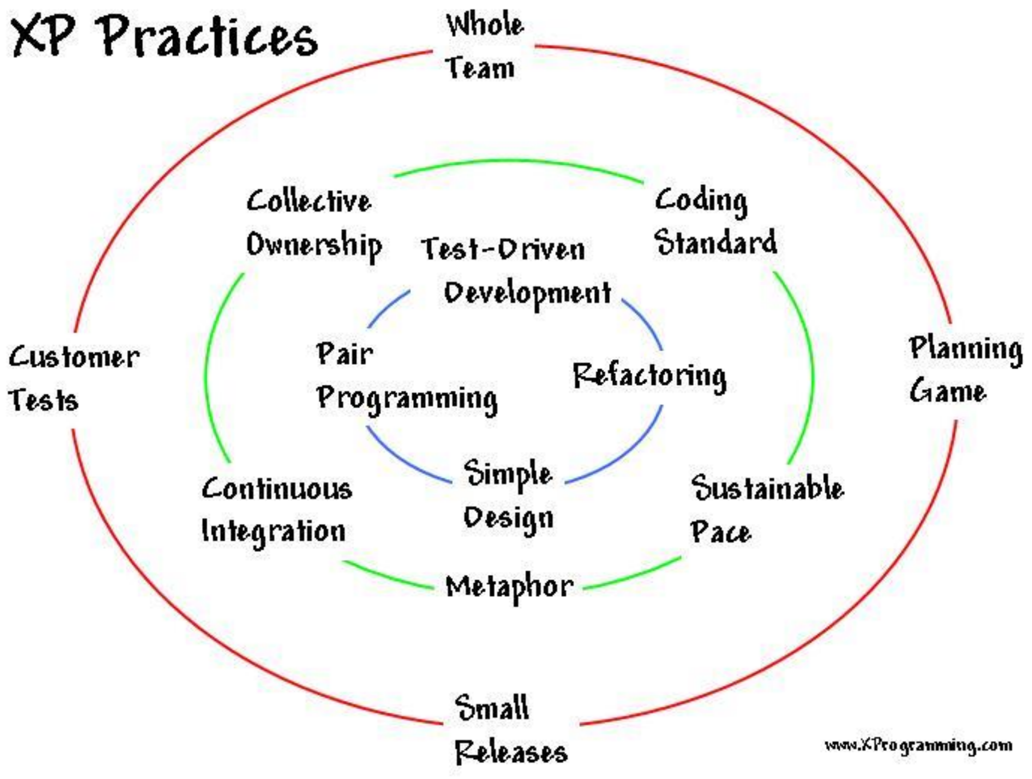
http://ronjeffries.com/xprog/what-is-extreme-programming/.

Metaphor

- Extreme Programming teams develop a common vision of how the program works, which we call the "metaphor". At its best, the metaphor is a simple evocative description of how the program works, such as "this program works like a hive of bees, going out for pollen and bringing it back to the hive" as a description for an agent-based information retrieval system.

http://ronjeffries.com/xprog/what-is-extreme-programming/.

Practices of XP



Further Reading and Descriptions

http://ronjeffries.com/xprog/what-is-extreme-programming/.

Refactoring

- The refactoring process focuses on removal of duplication (a sure sign of poor design), and on increasing the "*cohesion*" of the code, while lowering the "*coupling*". High cohesion and low coupling have been recognized as the hallmarks of well-designed code for at least thirty years. The result is that XP teams start with a good, simple design, and always have a good, simple design for the software.

http://ronjeffries.com/xprog/what-is-extreme-programming/.

Simple Design

- XP teams build software to a simple but always adequate design. They start simple, and through testing and design improvement, they keep it that way. An XP team keeps the design exactly suited for the current functionality of the system. There is no wasted motion, and the software is always ready for what's next.

http://ronjeffries.com/xprog/what-is-extreme-programming/.

Overview

- ~~XP~~
  - ~~General concepts~~

- Specific XP concepts
  - Pair programming
  - Test Driven Development
  - Continuous Integration

Pair Programming

"You'll never work alone"

Not without precedent

Pair programming

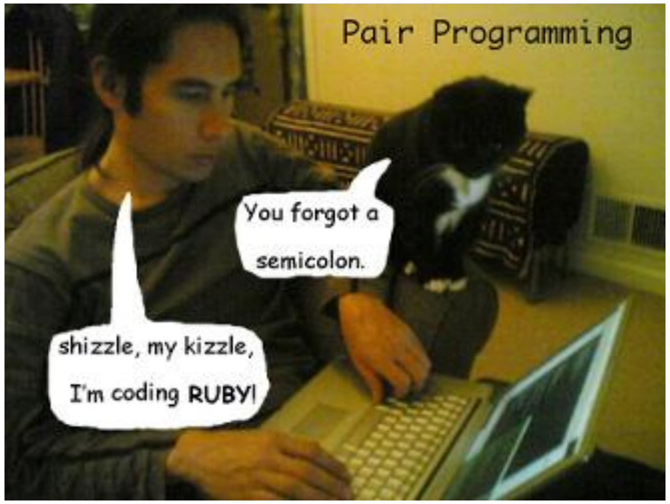
- Two developers working on the same task as a team
- One controls the keyboard one sits looking over their shoulder
- Driver
  - This person writes the code
- Navigator
  - This person reviews each line as it is typed

Advantages

- Higher code quality
  - Fewer bugs, code rewrites, integrations problems
- Expert novice pairing can help the novice to learn about the system and best practices
- Tends to produce more design alternatives and catches design defects earlier

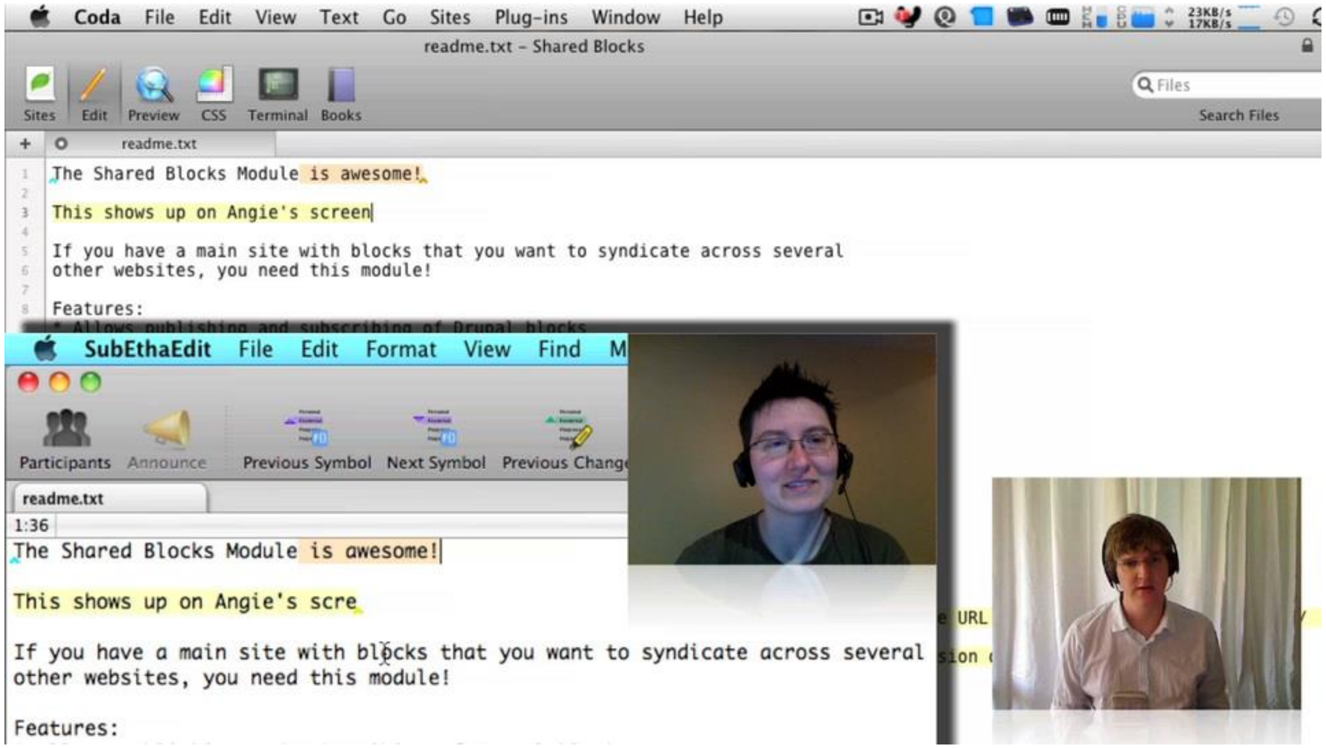
Disadvantages

- There is a high probability of disengagement
- “Watch the master” phenomenon
- Working relationship needs to be good
- Hard sell to management
  - Two people working on 1 feature



Remote pair programming

- Using communications technology
  - Screen sharing
  - IM clients, VOIP etc


Coda File Edit View Text Go Sites Plug-ins Window Help
readme.txt – Shared Blocks
Sites Edit Preview CSS Terminal Books
Files
Search Files
readme.txt
SubEthaEdit File Edit Format View Find
Participants Announce Previous Symbol Next Symbol Previous Change
readme.txt
1:36

```
The Shared Blocks Module is awesome!

This shows up on Angie's screen

If you have a main site with blocks that you want to syndicate across several
other websites, you need this module!

Features:
```

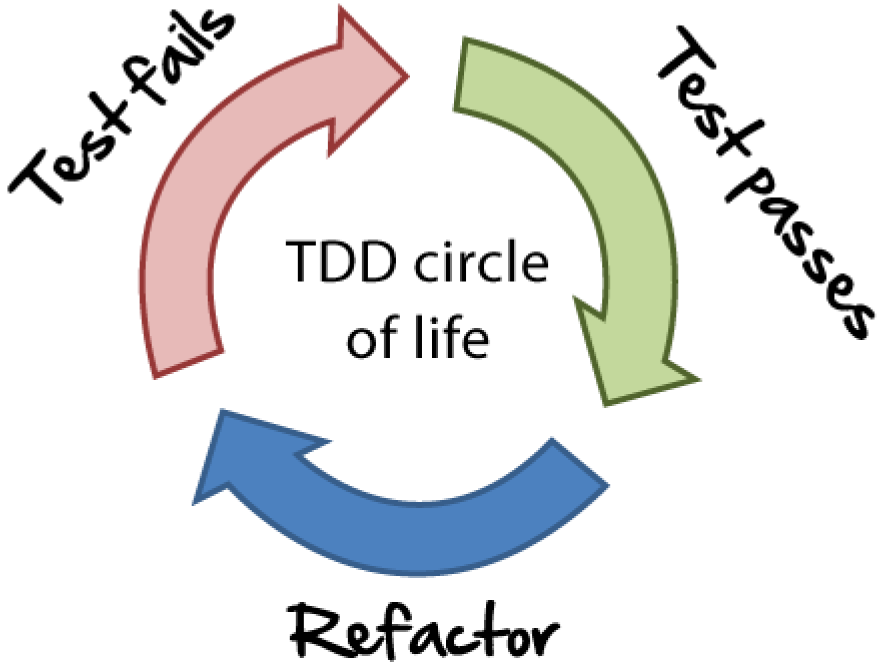
```
The Shared Blocks Module is awesome!

This shows up on Angie's scre

If you have a main site with blocks that you want to syndicate across several
other websites, you need this module!

Features:
```

Overview

- ~~XP~~
  - ~~General concepts~~

- Specific XP concepts
  - ~~Pair programming~~
  - **Test Driven Development**
  - Continuous Integration

Test Driven Development (TDD)



TDD Cycle

- Create a test
  - Each new feature requires a test
- Run all tests
  - Make sure the new test fails
- Write the code
  - Doesn’t have to be perfect, just pass the test
- Run all tests
  - If all tests pass, requirements are met
- Refactor code
  - TDD can result in duplication, this should be removed
- Repeat the process

Principles of TDD

- Never write new functionality without a failing test
- Continually make small incremental changes
- Keep the system running at all times
  - No one can make a change that breaks the system
  - Failures must be addressed immediately

Advantages

- Discourages “gold plating” of implementation

- Forces the developer to specify an end criteria

- Encourages loose coupling

Disadvantages

- Big time investment
- Complexity in writing appropriate test cases
- Design changes
- Mock code to pass tests
- Customer may not wish to get involved in creating acceptance tests

Interesting - IBM Study

- Study carried out by IBM focussed on a team that had been practising TDD for 5 years and delivered 10 releases of a software product

- Quality was the big winner, much improved, fewer defects/bugs etc

- Productivity did **decrease** but not dramatically

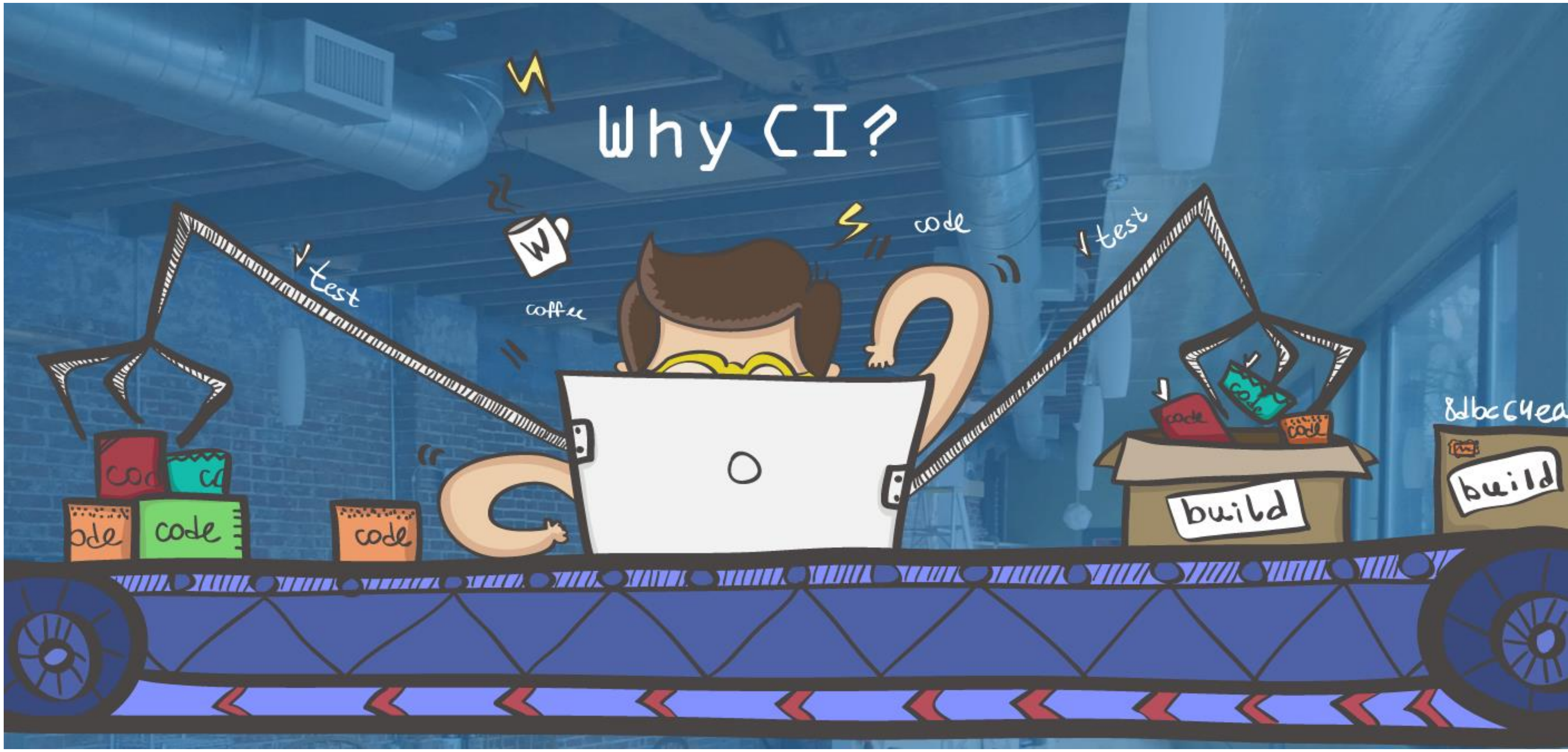
Sanchez, J., Laurie Williams, and E. Michael Maximilien. "A Longitudinal Study of the Use of a Test-Driven Development Practice in Industry." Proc. Agile. 2007.

Learning objectives

- ~~XP~~
  - ~~General concepts~~

- Specific XP concepts
  - ~~Pair programming~~
  - ~~Test Driven Development~~
  - **Continuous Integration**

Continuous Integration



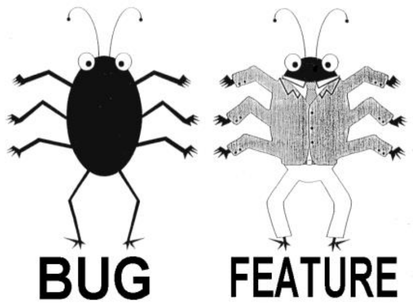
Continuous Integration (CI)

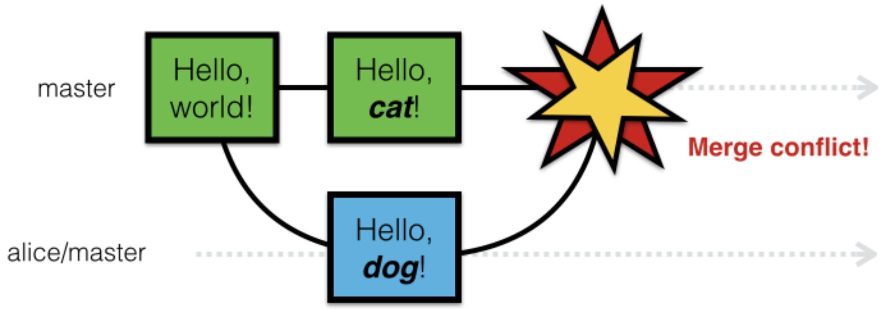
- CI is a practice where members integrate their changes frequently.
  - Often daily

- Each integration is verified by an automated build including tests to detect integration errors as early as possible.
  - Often upon commit, builds are run to make sure everything is okay

Development before CI

- Lots of bugs
- Infrequent commits
- Difficult integration
- Infrequent releases
- Testing happens late

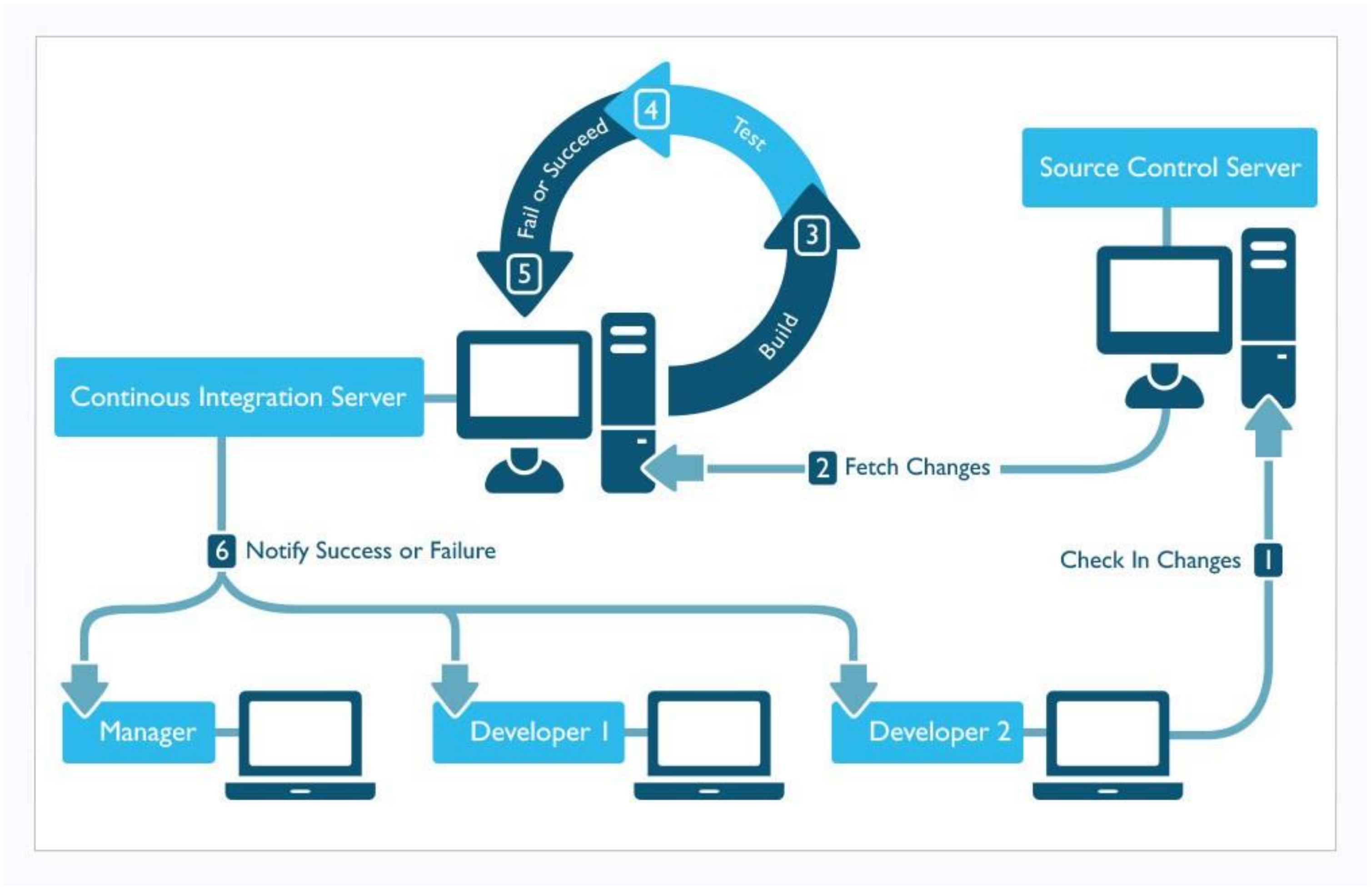




Benefits of CI

- Fail early/fast
  - Detect problems as early as possible
- Facilitates continuous deployments
  - Deploying every good build live to production
- Enables automated testing
  - Tests are run during the build process

Overview



https://insights.sei.cmu.edu/devops/2015/01/continuous-integration-in-devops-1.html

Drawbacks of using CI

- Initial setup required
  - Can take a couple of weeks to get it running properly within an organisation
- Excellent tests must be developed
  - CI will run all the automated tests but this requires substantial up front development effort.

Popular CI software

CODESHIP

circleci

Bamboo

Solano Labs

snap

semaphore

AppVeyor

drone.io