

What is Extreme Programming?

Don Wells

Don@extremeprogramming.org

<http://www.ExtremeProgramming.org>

What is a software methodology anyway?

It is a set of rules for developing software.

Where do these rules come from?

Experience on real projects, other engineering domains, and theories.

The Agile Manifesto

www.agilemanifesto.org

- People matter the most
- How much software works is what you measure
- Customers and developers work together
- Respond to change
- Evolutionary delivery of software
- Keep things simple
- Emergence

Characteristics of Extreme Programming (XP)

- Lightweight: The minimum needed to run a project.
- Disciplined: The rules must be followed until changed.
- Customer driven: The customer makes all business decisions.
- Humanistic: Based on human strengths and weaknesses.

The Minimum Needed

I would rather add one more element to a trivial system than try to delete 90% of a complex one.

XP Values

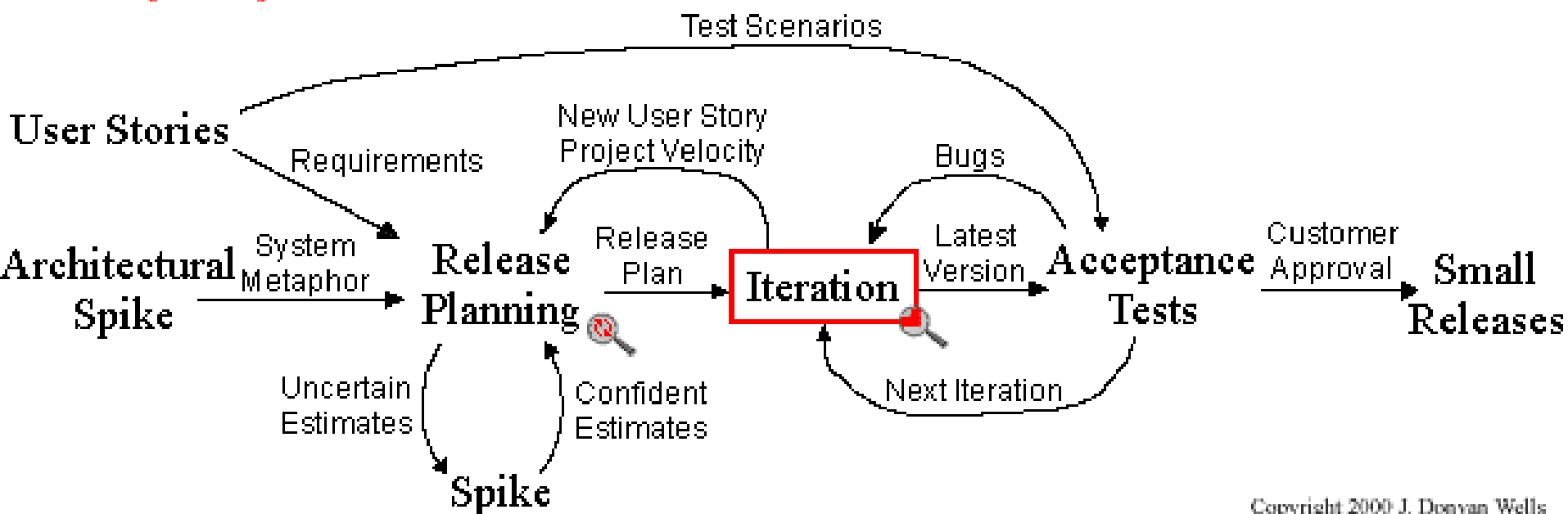
- Communication
- Feedback
- Simplicity
- Courage

The XP Process

- Planning: Concentric loops of planning and feedback.
- Designing: Simple, distributed across the team, based on oral tradition.
- Coding: High quality from start to finish, team level development.
- Testing: Two levels of automated testing, one for the customer and one for the developer.



Extreme Programming Project



User Stories

- Just enough to estimate the size of the effort.
- Written on a card.
- A few sentences only.
- No techno-babble included.

User Stories

- Stories must be backed up with a conversation
- Separate business and technical decisions
- Knowledge doesn't fit on paper
- Is it true? “These customers don't know what they want”

Plan to An Appropriate Level

- Release level is very course
- Plan several releases
- Iteration level is very fine
- Plan one maybe two iterations

Planning

- User stories are written.
- Release planning and iteration planning create the schedule.
- Make frequent small releases.
- The Project Velocity is measured.
- The project is divided into iterations.
- Iteration planning starts each iteration.
- A stand-up meeting starts each day.

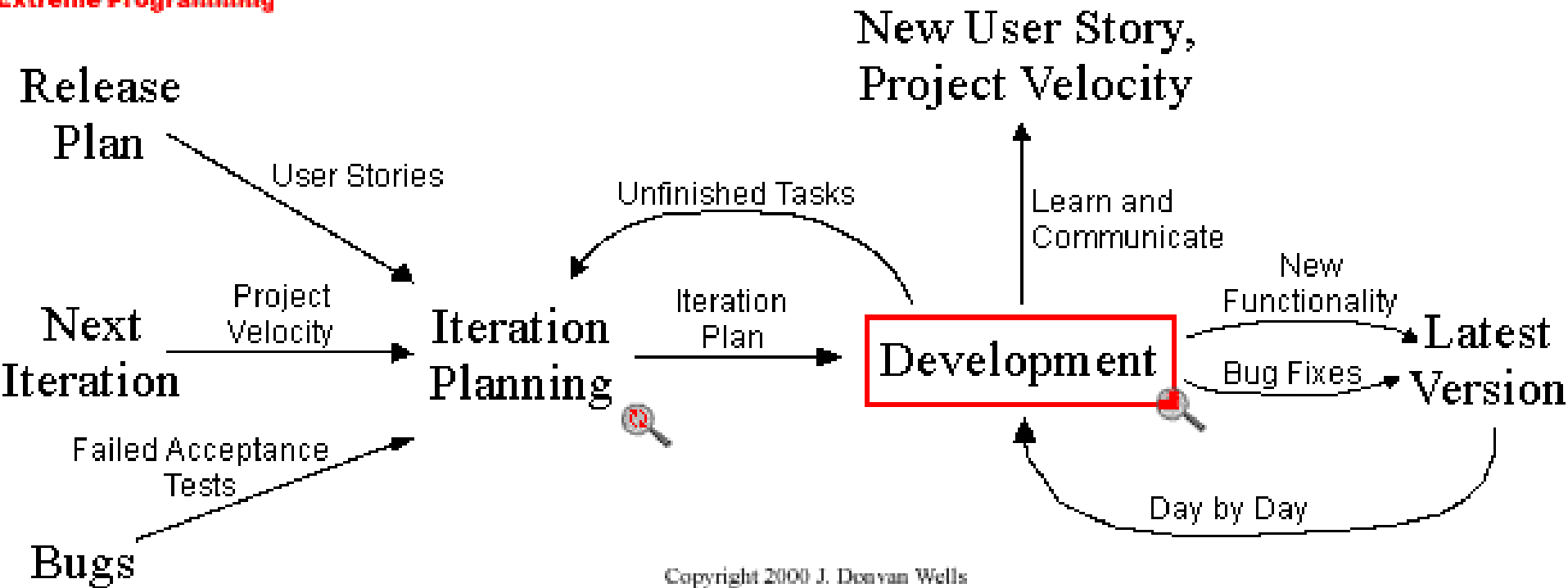
Release Planning

- Developers estimate stories.
- Developers combine stories.
- Customers write new stories.
- Customers break stories into pieces.
- Customers choose the most important stories.
- Management determines the project velocity.
- Management calculates the next release date.
- (Customers choose stories not to do.)



Iteration

Zoom Out



Iteration Planning

- Customers choose stories from the release plan to do.
- Customers substitute new stories into the release plan.
- Management determines the project velocity based on Yesterday's Weather.
- Developers break stories into tasks.
- Developers estimate tasks.
- Management calculates iteration content.
- (Customers choose stories not to do.)

Who's who in XP

- One team with 3 roles
 - Managers
 - Customers
 - Developers

Managers (Boss, Coach, Admin)

- Keep the pace steady
- Call for planning meetings
- Watch long term goals
- Make progress visible
- Remove obstacles
- Manage the customer relationship

Customer (Stakeholder, User, Analyst, Marketing, Buyer, QA)

- Create requirements
- Set priorities
- Set release dates
- Planning
- Explain requirements and answer questions
- Create and interpret acceptance tests
- Solution Analysis
- Make business decisions

Developer (Architect, Designer, Coder, DBA)

- Choose their own tasks
- Estimate their own work
- Planning
- Sets the pace
- Design and Architecture
- Unit Testing
- Coding
- Technical decisions

Designing

- Simplicity.
- Choose a system metaphor.
- Create spike solutions to reduce risk.
- No functionality is added early.
- Refactor whenever and wherever possible.

Simplicity

- Do the simplest thing that could possibly work
- Only solve today's problems
- Simple isn't easy
- A simple solution takes less time to build
- System metaphor

System Metaphor

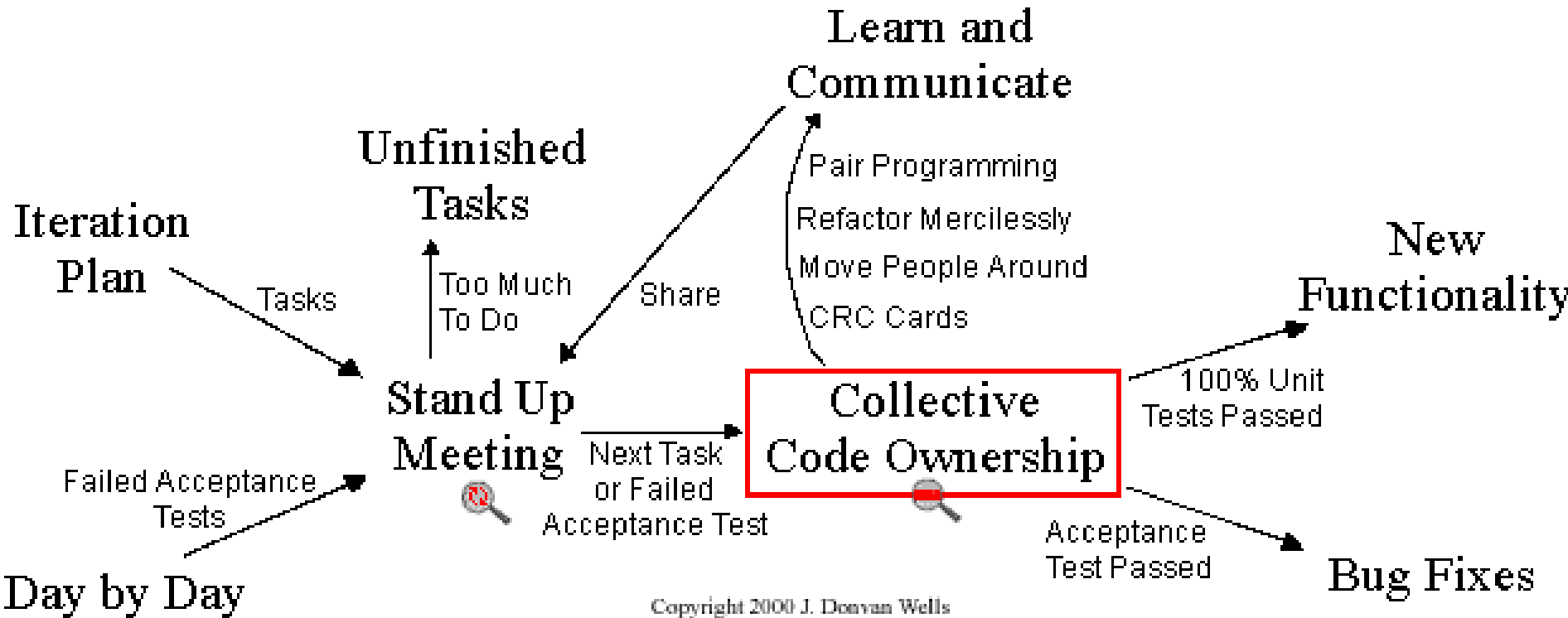
- A story that can be told about how the system works.
- A few simple flexible objects forming the core design.
- Keep it very simple and clean.
- The core design can be read and understood quickly.
- Extend the core design if needed but don't mess up the core.

Refactoring

- Small changes that do not change the function of the program.
- Increase design consistency.
- Simplify as you go.
- The design you need may not be the one you thought.



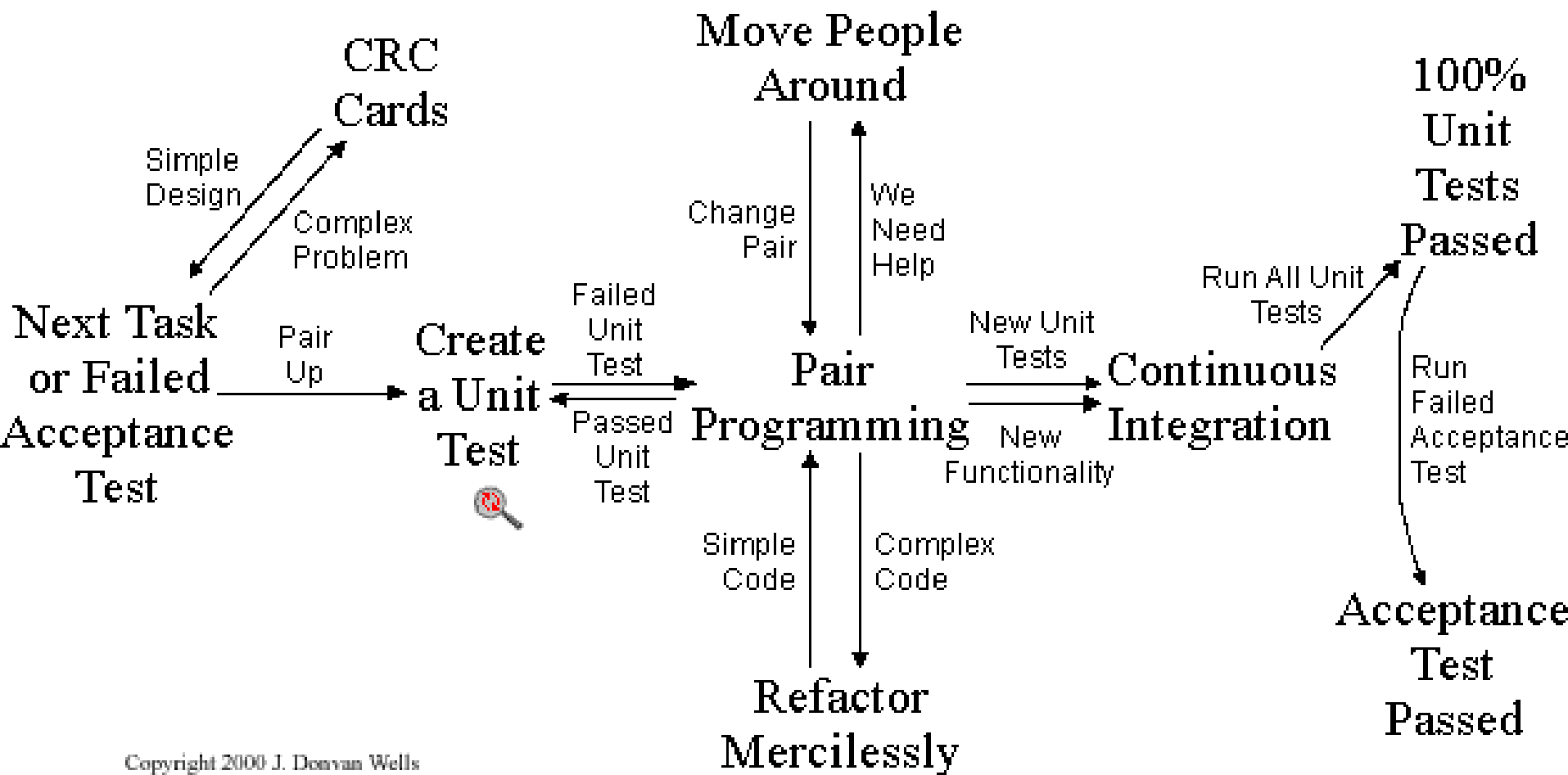
Development





Collective Code Ownership

Zoom Out



Coding

- The customer is always available.
- Code must be written to agreed standards.
- Code the unit test first.
- All code is pair programmed.
- Only one pair integrates code at a time.
- Integrate often.
- Use collective code ownership.
- Steady Pace

The Customer Is Always Available

- Keep your customer on-site.
- A user story is a promise to have a detailed conversation later.
- Never wait to get a question answered.
- The customer determines what the system does.

Code the Unit Test First

- Tightest of the planning loops.
- Design for testability is built in.
- Know exactly when you are done.
- Don't do more than you need.
- The test is the specification.
- Create a test, make it pass, create another test.

Pair Programming

- Two people one computer.
- Switch drivers often.
- Switch partners often.
- Same cost, less bugs, less code, less stress.

Integrate Often

- Integrate every few hours.
- Always work with the latest version of everything.
- Integration is not a big event.

Collective Code Ownership

- Anyone can change any line of code in the system at any time.
- Who ever finds a problem knows enough to design a solution.
- Refactor whatever, whenever possible.
- Maximize the number of good ideas.

Testing

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

What Is So New?

Attitude!

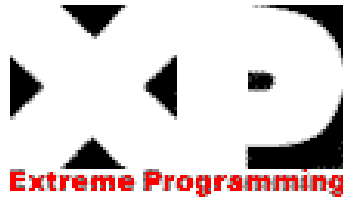
- A truly iterative process
- New assumptions about the cost of change
- Team work, real team work
- Testing as a part of development
- Alternative documentation
- Push the process down to the developers

A Truly Iterative Process

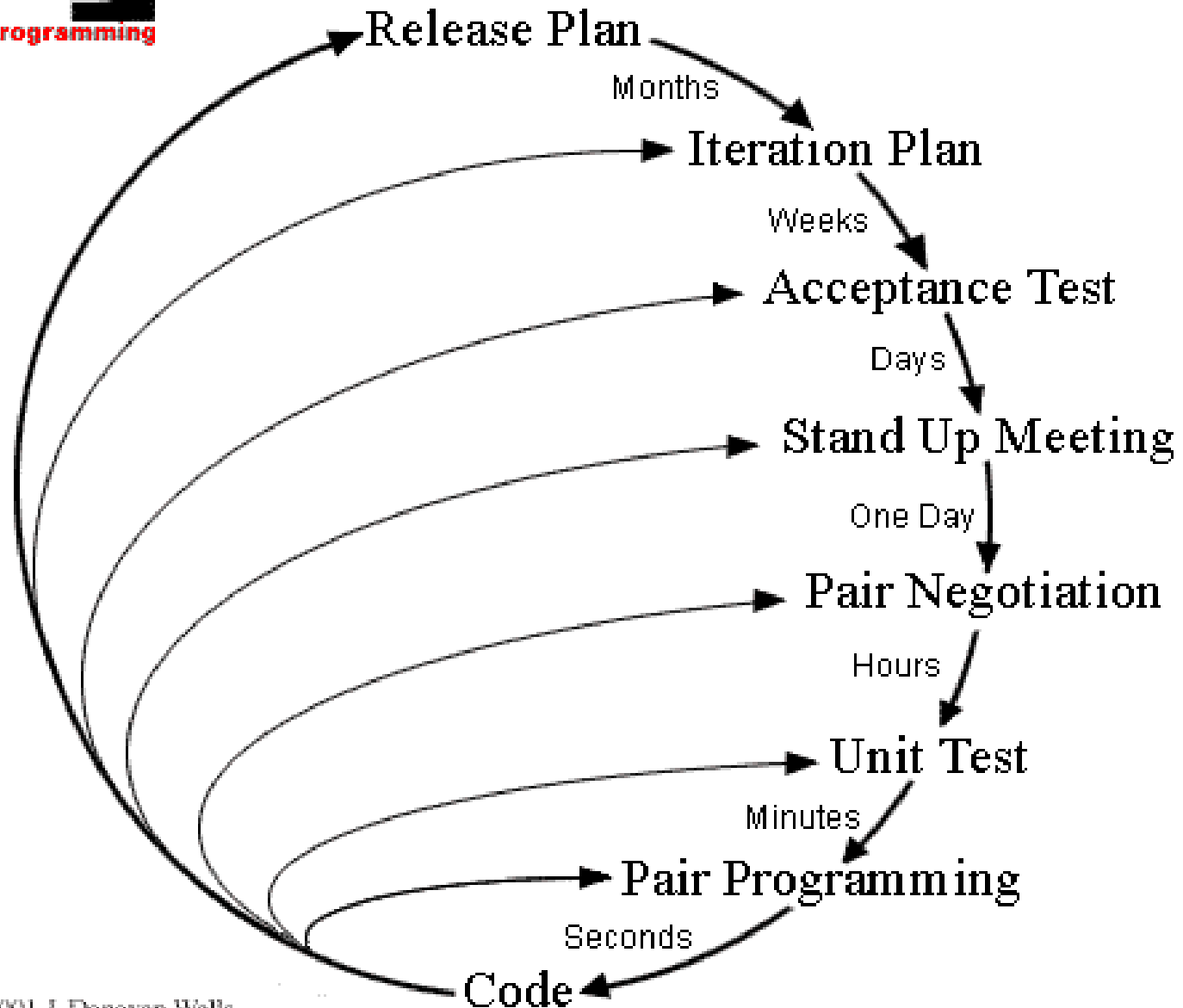
- Maintenance is the normal state
- Sustainable pace
- Feedback drives everything
- Focus on fixed delivery with variable scope
- Always ready for production
- Take the deadlines seriously

Feedback Is Important

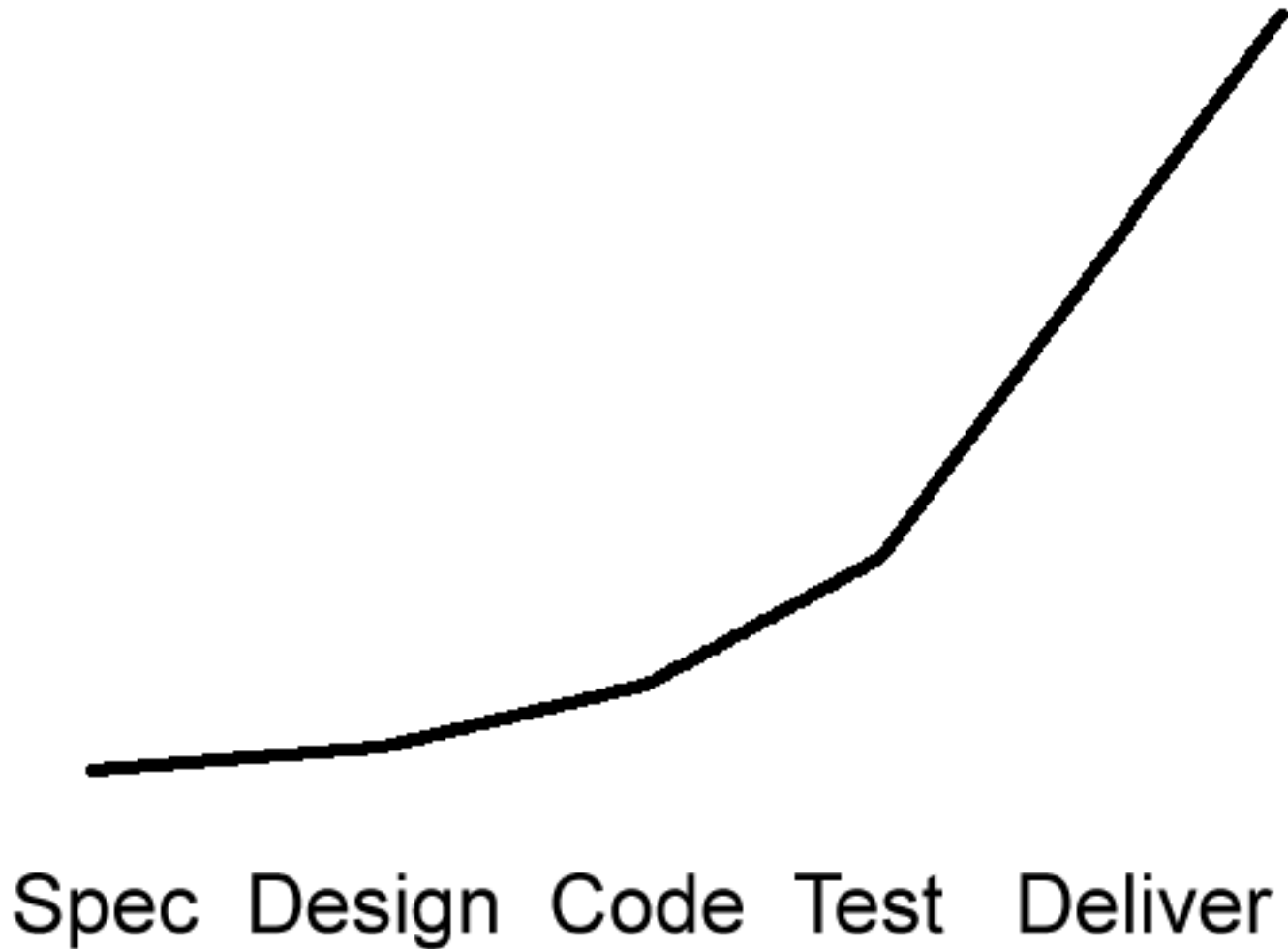
- Estimates not based on a measurement are guesses, estimates based on measurements are predictions.



Planning/Feedback Loops Zoom Out



Traditional Cost of Change Curve



Proposed Cost of Change Curve



Iteration 1 Iteration 2 Iteration 3

Team Work, Real Team Work

- Stand up meetings.
- Pair programming.
- Collective Code Ownership.
- The customer is here with us.
- Tell the truth.

What Makes a Team?

- Everyone contributes at their own level
- Everyone is in the yoke
- Everyone is of equal value to the project's success
- If you miss something, your team will not

The team's organization will be reflected in the code!

- Be cooperative, not competitive

Testing As a Part of Development

- If you test as you go testing can not be taken away
- Unit tests help make the code more testable and thus more reliable.
- The coverage you need for legacy code is not as much as you think.

Alternative Documentation

- Planning instead of a plan
- Stories instead of requirements
- Face-to-face conversation instead of design documents
- Tests instead of specifications
- The code speaks for it self instead of comments

When Should You Use XP

- Small project team
- Test-as-you go
- Open Workspace
- Collocated
- One team
- Willing to work together

What Makes It Hard?

- Managers need people skills
- Teams must come together
- Customers must lead
- Developers are part of the process
- No one works alone
- There is no royal family
- Developers need discipline
- Dedicated open work area

Transitioning to XP

- Add one practice at a time
- Fix your worst problem first
- Encouragement for compliance
- No consequences for non-compliance

Summary

- If it isn't fun you're doing something wrong
- Team work is more important than process
- Complex problems often have simple solutions
- Deliver something, anything, of value real soon
- Find new ways to get more from less