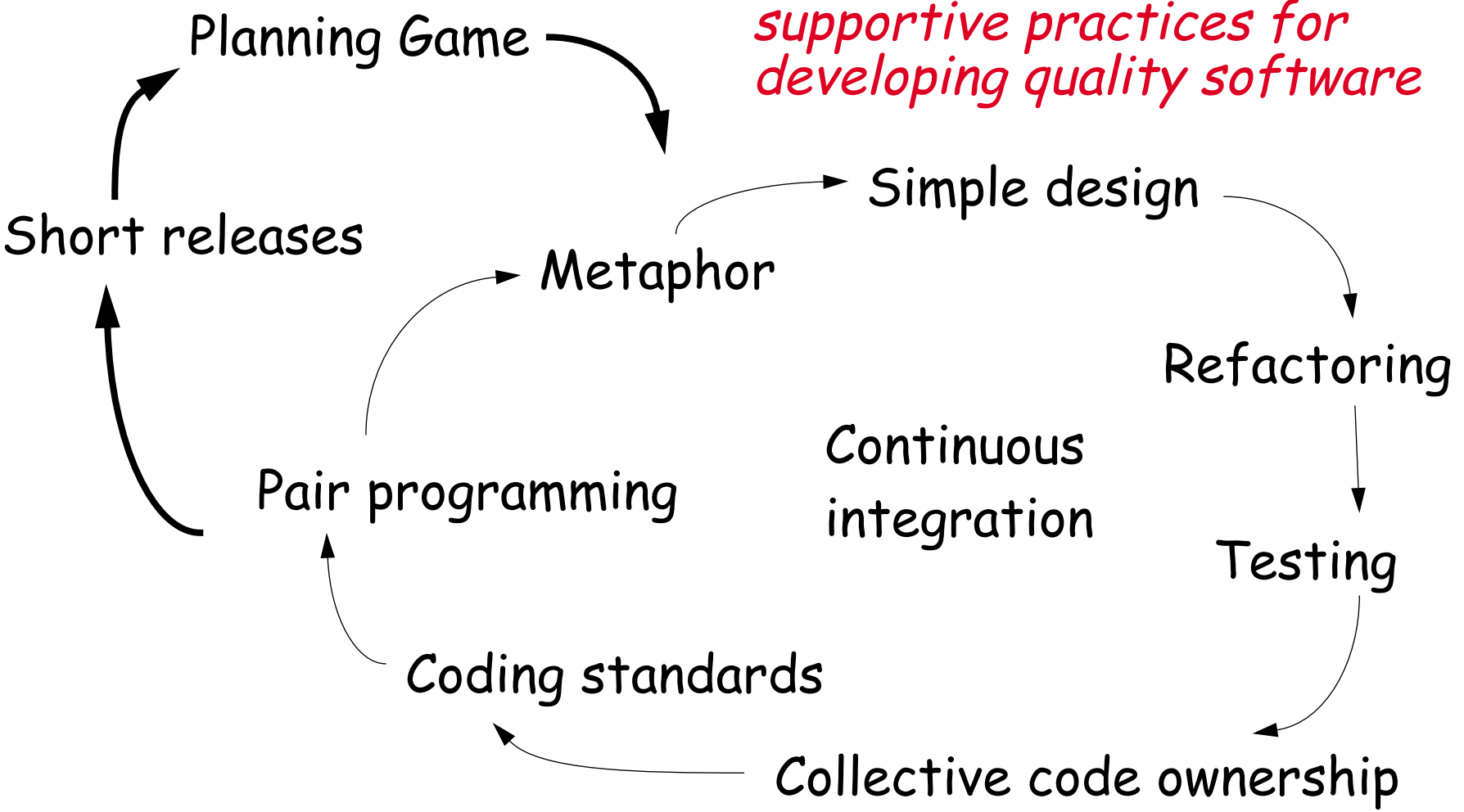
Extreme Programming

XP is a set of mutually supportive practices for developing quality software



Why we plan

We want to ensure that

- we are always working on the most *important* things
- we are *coordinated* with other people
- when *unexpected* events occur, we understand the *consequences* on priorities and coordination

Plans must be

- easy to *make* and *update*
- *understandable* by everyone that uses them

Separation of Roles

- Customer makes *business* decisions
- Developers make *technical* decisions

Business Decisions	Technical Decisions
Scope	Estimates
Dates of the releases	Dates within an iteration
Priority	Team velocity
	Warnings about technical risks

The Customer owns "what you get" while the Developers own "what it costs".

The Planning Game

A game with a set of rules that ensures that Customer and Developers don't become mortal enemies

Goal:

Maximize the value of the software produced by Developers.

Overview:

1. ***Release Planning:*** Customer selects the *scope* of the next release

2. ***Iteration Planning:*** Developers decide on *what to do* and in *which order*

The Release Planning Game

	Customer	Developers
Exploration Phase	*Write Story*	
		Estimate Story
	Split Story	
Commitment Phase	*Sort Stories by Value*	
		Sort Stories by Risk
		Set Velocity
	Choose Scope	
Steering Phase	*Iteration*	
		Recovery
	New Story	*Reestimate*

Planning Game: Exploration Phase

Purpose:

Get an appreciation for what the system should eventually do.

The Moves:

- ***Customer:*** *Write a story*. Discuss it until everybody understands it.
- ***Developers:*** *Estimate* a story in terms of effort.
- ***Customer:*** *Split* a story, if Developers don't understand or can't estimate it.
- ***Developers:*** Do a *spike solution* to enable estimation.
- ***Customer:*** *Toss* stories that are no longer wanted or are covered by a split story.

User Stories

Principles of good stories:

- *Customers* write stories. Developers do *not* write stories.
- Stories must be *understandable* to the customer
- The *shorter* the better. No detailed specification!
  - ☞ Write stories on index cards
- Each story must provide something of *value* to the customer
- A story must be *testable*
  - ☞ then we can know when it is done

Writing stories is an iterative process, requiring interaction between Customer and Developers.

Stories

A story contains:

- a name
- the story itself
- an estimate

Example:

- When the GPS has contact with two or fewer satellites for more than 60 seconds, it should display the message "Poor satellite contact", and wait for confirmation from the user. If contact improves before confirmation, clear the message automatically.

Initial Estimation of Stories

With no history, the first plan is the hardest and least accurate (fortunately, you only have to do it once)

How to start estimating:

- Begin with the stories that you feel the most comfortable estimating.
- Intuitively imagine how long it will take you.
- Base other estimates on the comparison with those first stories.

Spike Solutions:

Do a quick implementation of the whole story.

- Do not look for the perfect solution!
- Just try to find out how long something takes

Planning Game: Commitment Phase

Purpose:

Customer: *to choose scope and date of next delivery*

Developers: *to confidently commit to deliver the next release*

The Moves:

- ***Customer:*** *Sort* by stories by *value*

  (1) Stories without which the system will not function

  (2) Less essential stories, but still providing significant business value

  (3) Nice-to-have stories

  ☞ Customer wants the release to be as *valuable* as possible

- ***Developers:*** *Sort* stories by *risk*

  (1) Stories that can be estimated precisely *(low risk)*

  (2) Stories that can be estimated reasonably well

  (3) Stories that cannot be estimated *(high risk)*

  ☞ Developers want to tackle *high-risk first*, or at least *make risk visible*

- ***Developers:*** Set team *velocity*

  How much ideal engineering time per calendar month/week can the team offer?

  ☞ this is the *budget* that is available to Customer

- ***Customer:*** Choose *scope* of the release, by either

  — fixing the date and choosing stories based on estimates and velocity

  — fixing the stories and calculating the delivery date

Planning Game: Steering Phase

Purpose: *Update the plan based on what is learned.*

The Moves:

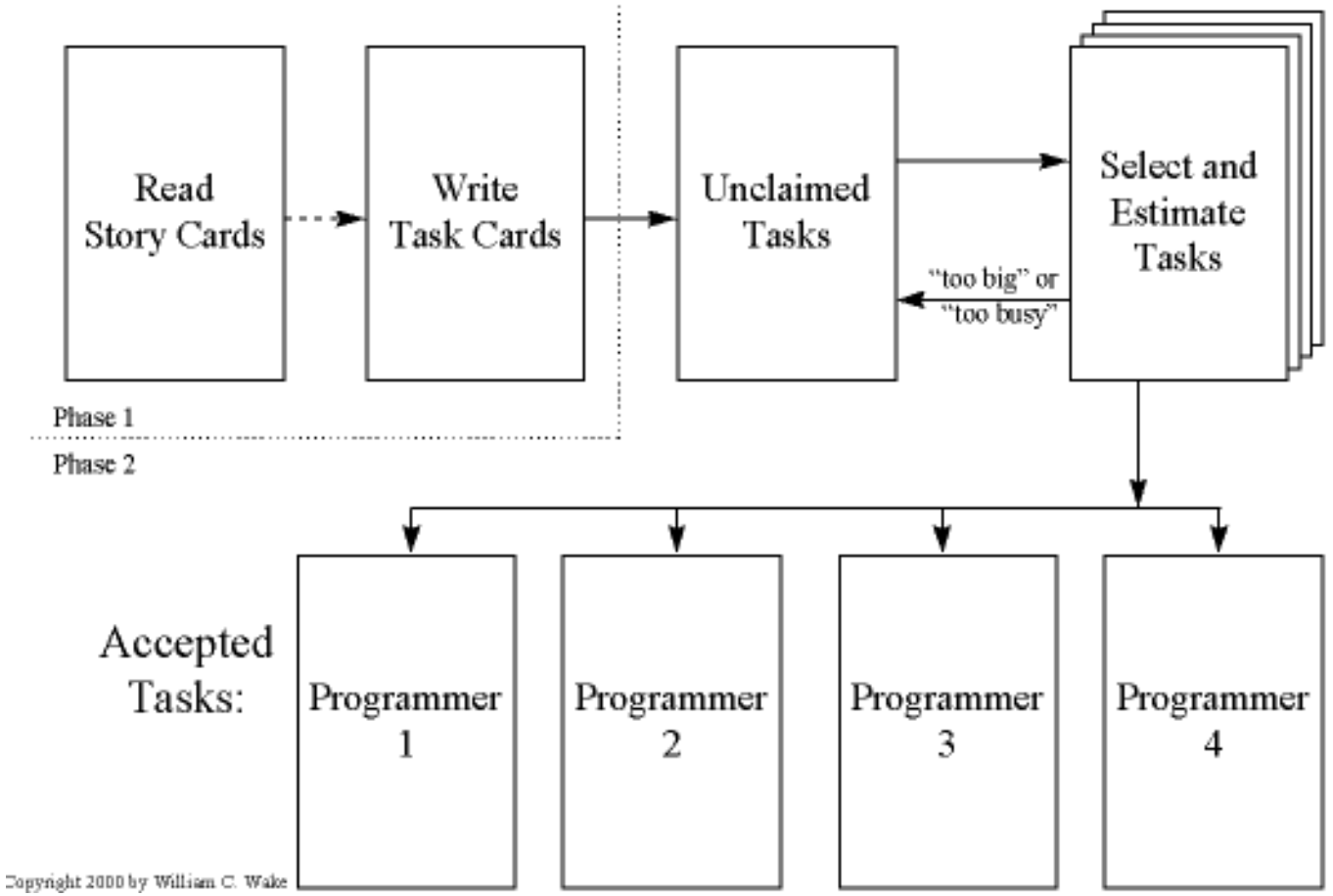
- ***Iteration:*** Customer *picks* one iteration worth of the most valuable *stories*.
  - ☞ see Iteration Planning
- ***Get stories done:*** Customer should only accept stories that are *100% done*.
- ***Recovery:*** Developers realize *velocity is wrong*
  - — Developers re-estimate velocity.
  - — Customer can defer (or split) stories to maintain release date.

...

Planning Game: Steering Phase ...

- ***New Story:*** Customer identifies *new, more valuable stories*
  - Developers estimate story
  - Customer removes estimated points from incomplete part of existing plan, and inserts the new story.
- ***Reestimate:*** Developers feel that *plan is no longer accurate*
  - Developers re-estimate velocity and all stories.
  - Customer sets new scope plan.

Iteration Planning



Iteration Planning

- ***Customer*** *selects* stories to be implemented in this iteration.
- ***Customer*** *explains* the stories in detail to the Developers
  - —Resolve ambiguities and unclear parts in discussion

...

Iteration Planning ...

- ***Developers*** brainstorm *engineering tasks*
  - — A task is small enough that everybody fully understands it and can estimate it.
  - — Use short CRC or UML sessions to determine how a story is accomplished.
  - ☞ Observing the design process builds common knowledge and confidence throughout the team
- ***Developers/pairs*** *sign up* for work and estimates
  - — Assignments are not forced upon anybody (Principle of Accepted Responsibility)
  - — The person responsible for a task gets to do the estimate