פיתוח מערכות תוכנה מבוססות Java

משחק התכנון

אוהד ברזילי

ohadbr@tau.ac.il

Based on: K. Beck: Extreme Programming Explained.
E. M. Burke and B.M. Coyner: Java Extreme Programming Cookbook.
L. Crispin and T. House: Testing Extreme Programming
http://www.extremeprogramming.org

And slides of: Kent Beck and Ward Cunningham,
Laurie Williams, Vera Peeters and Pascal Van Cauwenberghe,
Ian Sommerville:
http://www.comp.lancs.ac.uk/computing/resources/IanS/SE7/Presentations/index.html

The Planning Game:
How it works

The Planning Game Rationale

- http://c2.com/cgi/wiki?PlanningGame

- Planning is an emotional minefield. Of course **Development** would like to program faster. Of course the project manager would like to be able to say exactly how fast **Development** can go. Of course **Business** would like to be able to say exactly what they want. Of course **Business** would rather not change its mind. When any of the participants in planning begin acting these wishes (or rather in accordance with the fears that lie behind each wish), then planning doesn't work well.

- The Planning Game: Create a little emotional distance from planning by treating it as a game (hence the name). The game has a **goal, playing pieces, players**, and **rules** for allowable moves.

The Planning Game (1)

Pieces: The basic playing piece is the UserStory. Each Story is written on an **index card.** Stories have a **value** and a **cost,** although this is a little tricky because the value of some Stories depends on the presence or absence of other Stories (see StoryDependenciesInXp), and the values and costs change over time.

Goal: The goal of the game is to put the greatest possible value of stories into production over the life of the game.

Players: The players are **Business** and **Development.**

Moves:

- ***Write Story***: **Business** can write a new Story at any time. For purpose of the Planning Game, writing a Story just means assigning it a value (in practice, it has to have enough information for **Development** to assign it a cost).

The Planning Game (2)

- ***Estimate Story**:* **Development** takes every story and assigns it a cost of 1, 2, or 3 weeks of IdealProgrammingTime (c.f. ExtremeTimeSpans) If the estimate is higher, **Business** splits the story. (This may result in the story being implemented over more than one Iteration.) If the estimate is lower, **Business** merges it with another story. (Sometimes we just batch small stories willy-nilly until they add up to at least a week, for estimation purposes. Don't try that at home.) We use a LoadFactor (see ProjectVelocity) of 3 real weeks per ideal week to convert the final schedule to real time.

- ***Make Commitment:*** **Business** and **Development** work together to decide what stories constitute the next release and when it will be ready to put into production. There are two ways to drive the commitment, **Story Driven** and **Date Driven**.

The Planning Game (3)

- ***Story Driven Commitment:*** **Business** starts putting the Stories for the next release on the table. As each Story is introduced, **Development** calculates and announces the release date. This move stops when **Business** is satisfied that the Stories on the table make sense as the next release.

- ***Date Driven Commitment:*** **Business** picks a release date. **Development** calculates and announces the cumulative cost of Stories they can accomplish between now and the date. **Business** picks Stories whose cost adds up to that number.

The Planning Game (4)

- ***Value (and Risk?) First:*** **Development** orders the Stories in a commitment so:
  1. A fully working but sketchy system is completed immediately (like in a couple of weeks)
  2. More valuable Stories are moved earlier in the schedule (BusinessValueFirst)
  3. Riskier Stories are moved earlier in the schedule (WorstThingsFirst)

- ***Overcommitment Recovery***: **Development** had predicted they could do 150 units of stories between now and the deadline. Based on measuring ProjectVelocity, they find and immediately announce that they can only do 100. **Business** selects the 100 units of Stories to retain, deferring the other Stories to a future release. (Or highly unlikely: **Business** decides to defer the deadline to get the extra 50 units done.)

The Planning Game (5)

- ***Change Value:*** **Business** changes the value of a Story. In response, **Development** may change the order of Stories not yet completed.

- ***Introduce New Story***: **Business** writes a new Story. **Development** estimates it. **Business** defers Stories in the current Commitment whose cumulative cost is the cost of the new Story. **Development** re-evaluates Value and Risk First.

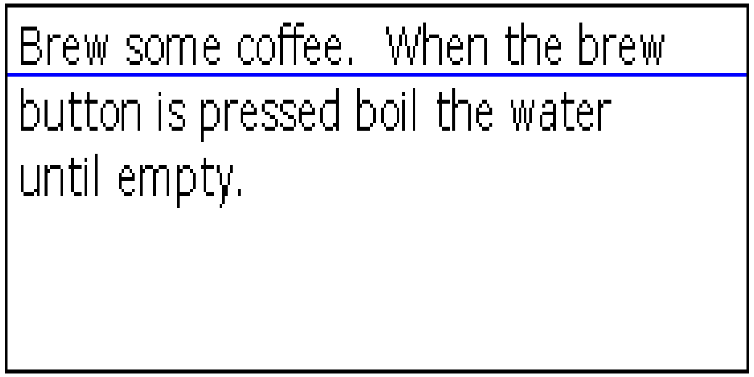
- ***Split Story:*** **Business** splits a Story into two or more. **Business** assigns a value to each, and **Development** assigns a cost to each. Typically this is done because resources do not permit the whole story to be done soon enough.

The Planning Game (6)

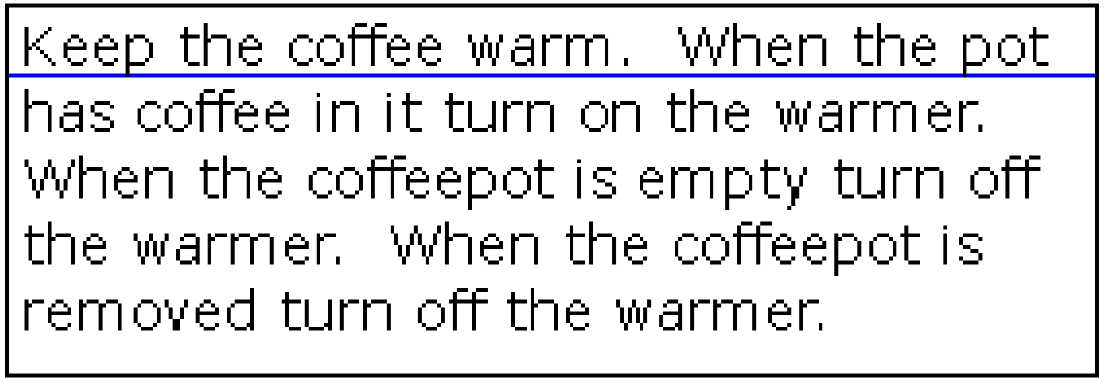
- ***Spike:*** **Business** can divert Project resources to do a throwaway Spike to fight a fire or prove a concept. If this Spike is anything more than a temporary fix, **Business** makes a UserStory to account for it. That Story is scheduled according to Value And Risk First. Regular spikes, especially fire-fighting ones, will affect the LoadFactor.

- ***Re-estimate:*** **Development** estimates the remaining stories in the Commitment again. This can spark an OvercommitmentRecovery.

Story Cards for a Coffee Maker

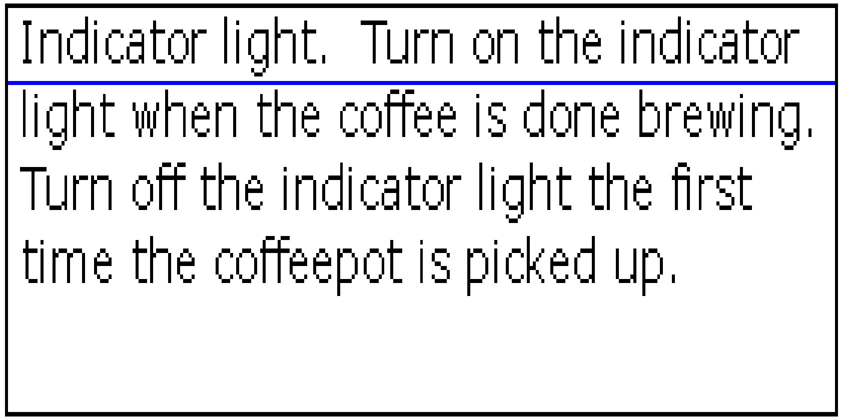


Brew some coffee. When the brew button is pressed boil the water until empty.

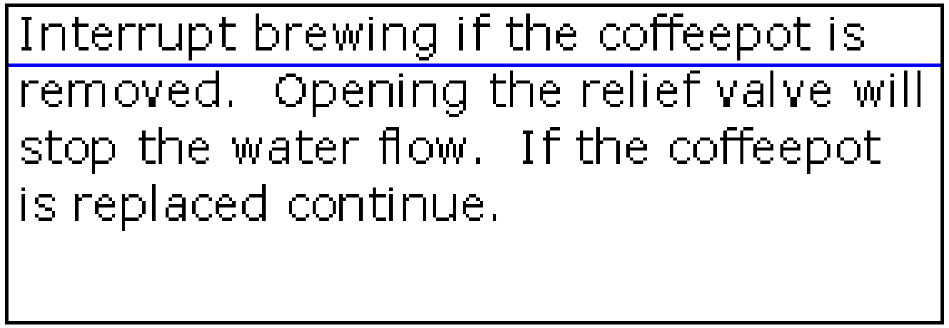


Keep the coffee warm. When the pot has coffee in it turn on the warmer. When the coffeepot is empty turn off the warmer. When the coffeepot is removed turn off the warmer.

Story Cards for a Coffee Maker



Indicator light. Turn on the indicator light when the coffee is done brewing. Turn off the indicator light the first time the coffeepot is picked up.



Interrupt brewing if the coffeepot is removed. Opening the relief valve will stop the water flow. If the coffeepot is replaced continue.

Story card for document downloading

Downloading and printing an article

First, you select the article that you want from a displayed list. You then have to tell the system how you will pay for it - this can either be through a subscription, through a company account or by credit card.

After this, you get a copyright form from the system to fill in and, when you have submitted this, the article you want is downloaded onto your computer.

You then choose a printer and a copy of the article is printed. You tell the system if printing has been successful.

If the article is a print-only article, you can't keep the PDF version so it is automatically deleted from your computer.

Release Planning

A release round includes 3 phases:

- Exploration phase
- Commitment Phase
- Steering Phase

Release Planning

Exploration phase:

- **Goal:** Next release planned that maximizes value/effort
- **Result:** list of stories (and tasks) to be included in next release
- **Moves:** Write a story, estimate a story, split a story.

Commitment phase:

- **Goal:** Customer sorts stories by value;
  Programmers sort stories by risk.
- **Moves:** sort by value, sort by risk, set velocity, choose scope.

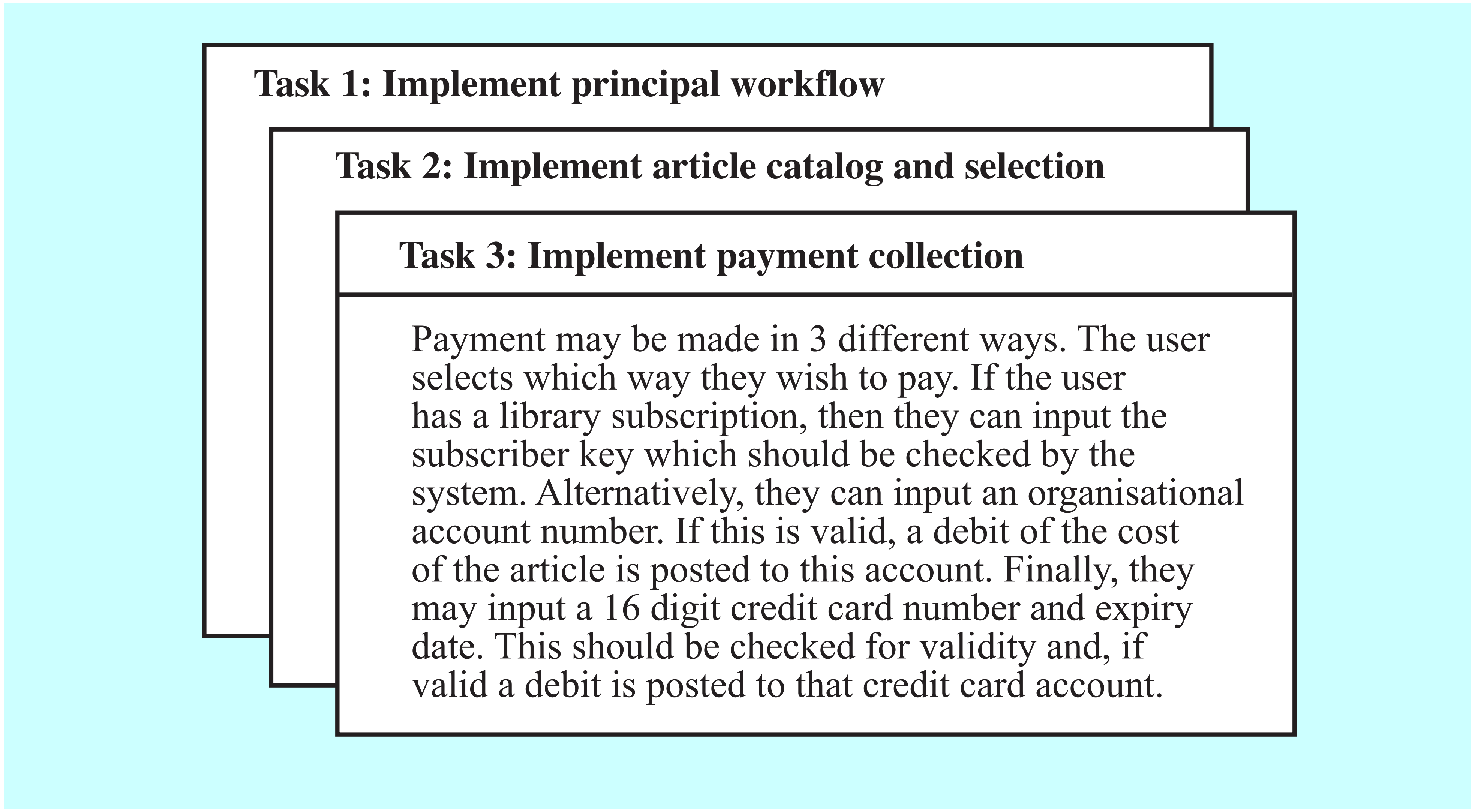
Steering phase:

- **Goal:** Update the plan.
- **Moves:** iteration, recovery, new story, re-estimate.

Iteration Planning

- An iteration takes from 1-3 weeks**.**
- Stories are split into **tasks**.
- Same game as in the release round.

Task cards for document downloading



Task 1: Implement principal workflow

Task 2: Implement article catalog and selection

Task 3: Implement payment collection

Payment may be made in 3 different ways. The user selects which way they wish to pay. If the user has a library subscription, then they can input the subscriber key which should be checked by the system. Alternatively, they can input an organisational account number. If this is valid, a debit of the cost of the article is posted to this account. Finally, they may input a 16 digit credit card number and expiry date. This should be checked for validity and, if valid a debit is posted to that credit card account.

Project parameters and evaluation

- 4 control variables:
  - ***Cost.***
  - ***Time***
  - ***Quality***
  - ***Scope***

- 3 control variables – selected by Customers, Managers.
  The remaining variable – selected by the Development team.

- Suggestion: Fix Cost, Time, Quality
  → Tune Scope accordingly.

XP and Fixed Price

- How can you do a fixed price / fixed date / fixed scope contract if you play the Planning Game?

  → You will end up with a

  fixed price / fixed date / roughly variable scope

  contract.

- Beck says: "Every project I've worked on that had fixed price and scope ended with both parties saying, "The requirements weren't clear."

- Instead of fixed price/date/scope, the XP team offers something more like a subscription.

- A 12-month contract might put the system into production after three or four months, with monthly or bimonthly releases thereafter.

XP and Visual Design

- **Advantages of visual design:** Provides
  - Clues on a design problem:
    - » Too many elements in the picture.
    - » Obvious asymmetry.
    - » Many more lines than boxes (high coupling).
  - Speed.

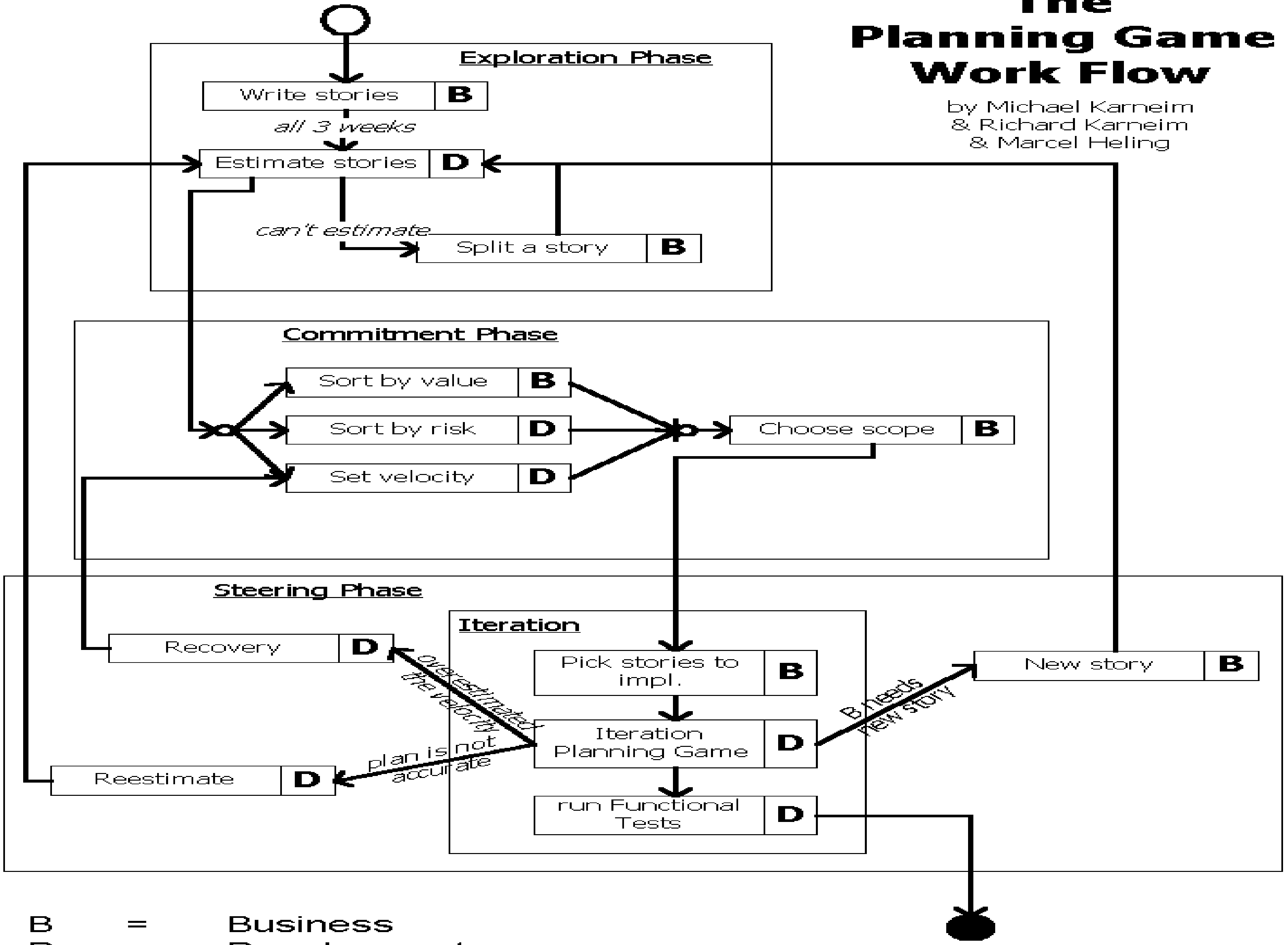
- **Problems of visual design:** No feedback about
  - Test passing.
  - Simple code.

- **Strategy:**
  - Draw a few pictures at a time.
  - Implement in testing + code (+ refactoring).
  - Do not save implemented pictures, since the decisions will probably change.
  - Draw pictures on a whiteboard.

- **Possibly**: Use a reverse engineering tool for getting a visual description of the system, If needed.

The Planning Game Work Flow

by Michael Karneim & Richard Karneim & Marcel Heling



B = Business
D = Development

XP Roles and responsibilities

Programmer - writes tests and then code.

Customer - writes stories and functional tests.

Tester - helps customer write tests and runs them.

Tracker - gives feedback on estimates and process on iterations.

Coach - person responsible for whole process.

Consultant - supplies specific technical knowledge needed.

Manager - makes decisions.

Handling Problems

Underestimation

- Sometimes too great a commitment will be made.
- Check to see if rules are being followed.
- If stories cannot be completed, ask the user to choose a subset.
  - Other stories will be finished later.

Uncooperative Customers

- Some customers won’t play the game.
- XP relies on trust.
- Don’t move on based on guesses.
- If customer never makes an effort, perhaps the system isn’t worth being built.

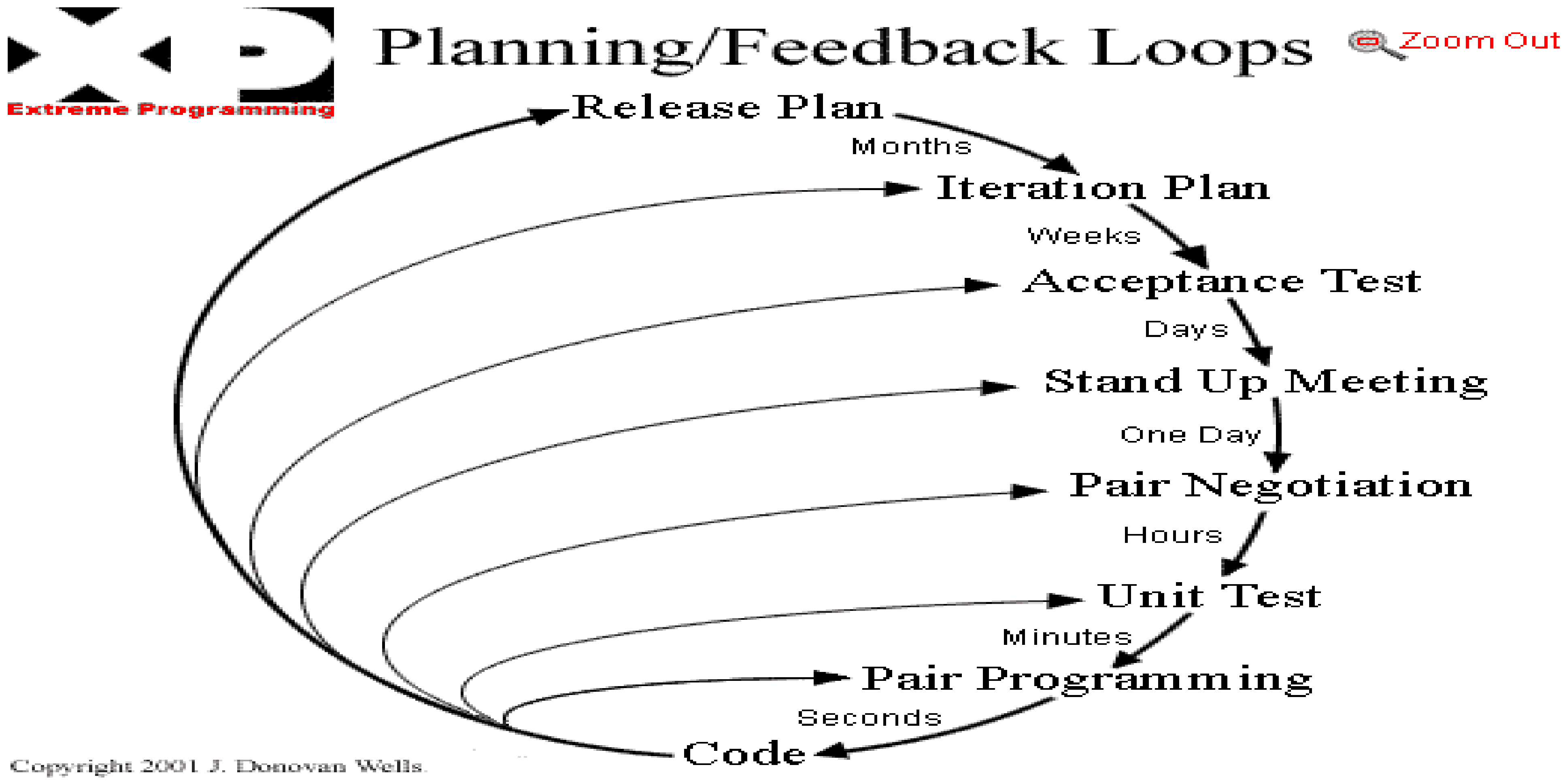
Turnover

- If programmers leave, they don’t take any information that only they have.
- Tests exist for every feature, so nothing can be broken by ignorance.
- New people can be trained by pairing with experienced programmers.

Changing Requirements

- This isn’t a problem for XP as it is for other development models.
- Have only planned for today, won’t have to change our plans.
- New features will just be added to the stories.

XP Planning/Feedback times



Extreme Programming
Planning/Feedback Loops
Zoom Out
Release Plan
Months
Iteration Plan
Weeks
Acceptance Test
Days
Stand Up Meeting
One Day
Pair Negotiation
Hours
Unit Test
Minutes
Pair Programming
Seconds
Code
Copyright 2001 J. Donovan Wells.