Einführung von XP in der Praxis

Seminar “Agile vs. klassische Methoden der Software-Entwicklung”

David Kocher, dk@cyberduck.ch

Einführung

- XP bezeichnet sich als “leichte Methode” des Software Engieneerings
- Welches sind die Praktiken um eine hohe Qualität und schneller Business Value garantieren?

Planungsstrategie

- Releaseplanung
- Iterationsplanung

Customer Story

- Beschreibung einer Anforderung
- vom Kunden verfasst
- Funktionalitätsmass
- Grundlage für Akzeptanztests

Prinzipien von guten Story Cards

- für den Kunden verständlich
- keine detaillierte Spezifikation
- Umsetzung bringt direkter Nutzen (Funktionalität)
- testfähig
- klein (mehrere Stories in einer Iteration)

Story Card

Customer Story and Task Card B/W Development \ COLA

DATE: 3/19/98 TYPE OF ACTIVITY: NEW: X FIX: ___ ENHANCE: ___ FUNC. TEST ___

STORY NUMBER: 1275 PRIORITY: USER: ___ TECH: ___

PRIOR REFERENCE: ___ RISK: ___ TECH ESTIMATE: ___

TASK DESCRIPTION:
SPLIT COLA: When the COLA rate chgs. in the middle of the B/W Pay Period we will want to pay the 1ST week of the pay period at the OLD COLA rate and the 2ND week of the Pay Period at the NEW COLA rate. Should occur "automatically based on system design.

NOTES:
For the OT, we will run a m/frame program that will pay or calc the COLA on the 2ND week of OT. The plant currently retransmits the hours data for the 2ND week exclusively so that we can calc COLA. This will come into the Model as a "2144" COLA

TASK TRACKING: Gross Pay Adjustment. Create RM Boundary and Place in DEEntExcessCOLA BIN.

Date	Status	To Do	Comments

Planungsspiel

- Sowohl in der Releaseplanung als auch in der Iterationsplanung eingesetzt
- Abstimmen der Anfoderungen des Kunden mit der Programmierkapazität

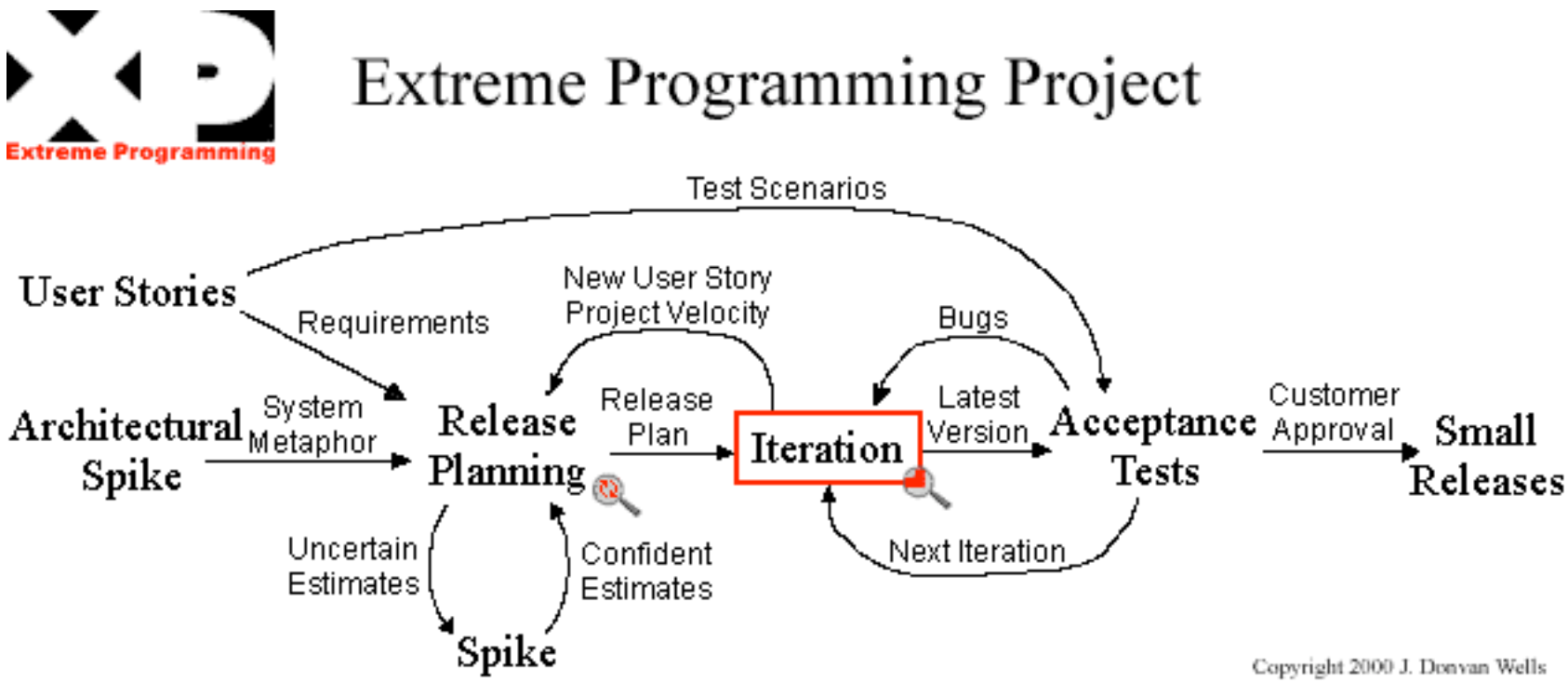
Entscheidungen im Planungsspiel

- Umfang
- Priorität

Releaseplanung

- Projekt auf die gesamte Dauer geschätzt
- lediglich Momentaufnahme
- Aufwand der Benutzergeschichten in “Ideal Programming Weeks” messen
- Zusammenstellung der Releases
- Zeitpunkt der Releases

Releaseplanung



Aufwandschätzung

- Einfach halten (Keep it simple)
- Vergleiche mit früheren Projekten (Use what happened in the past)
- Erfahrungen (Learn from the experience)
- Teamarbeit
- “Spike” schreiben, wenn keine Erfahrung

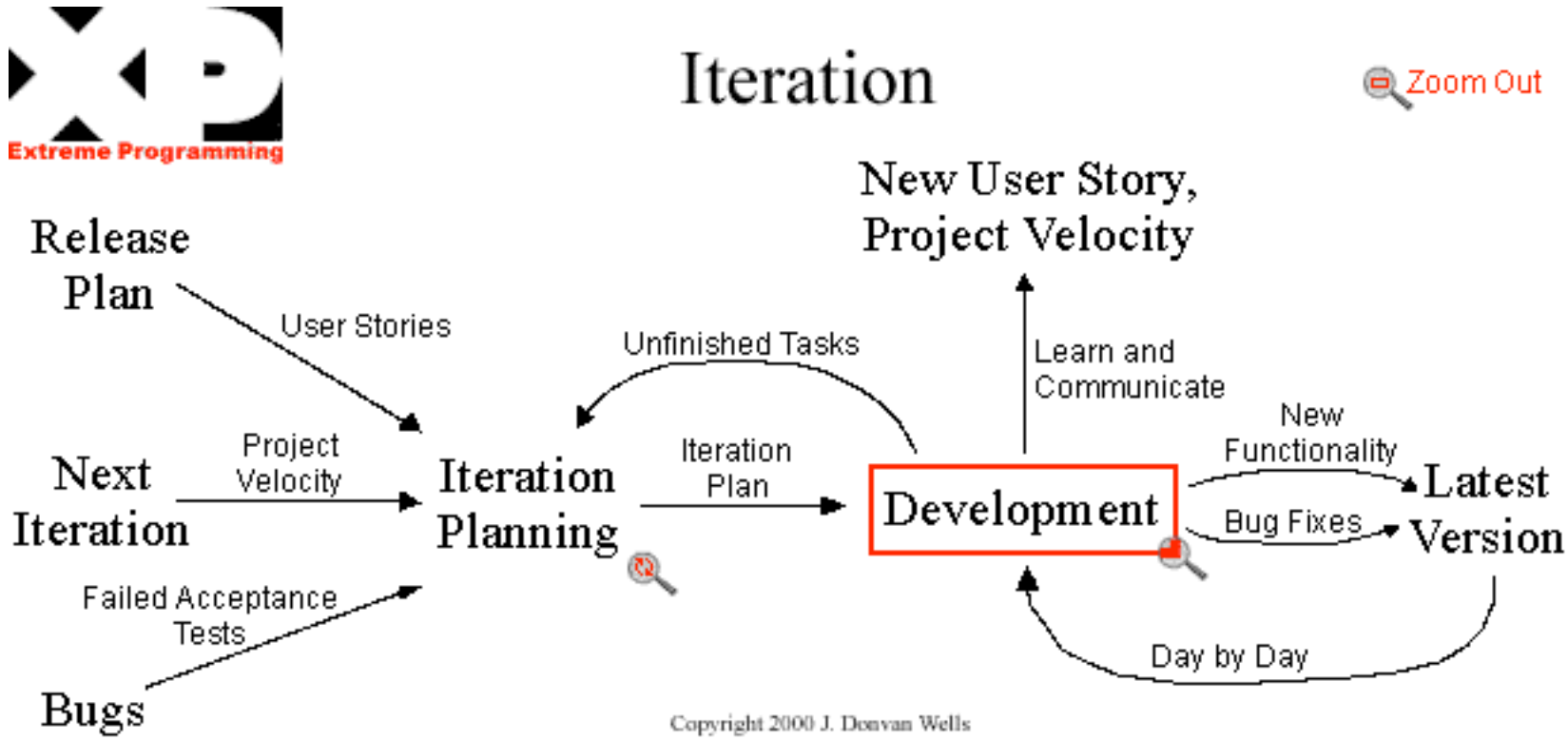
Commitment

- Business Value (high-value first)
- Technisches Risiko (high-risk first)

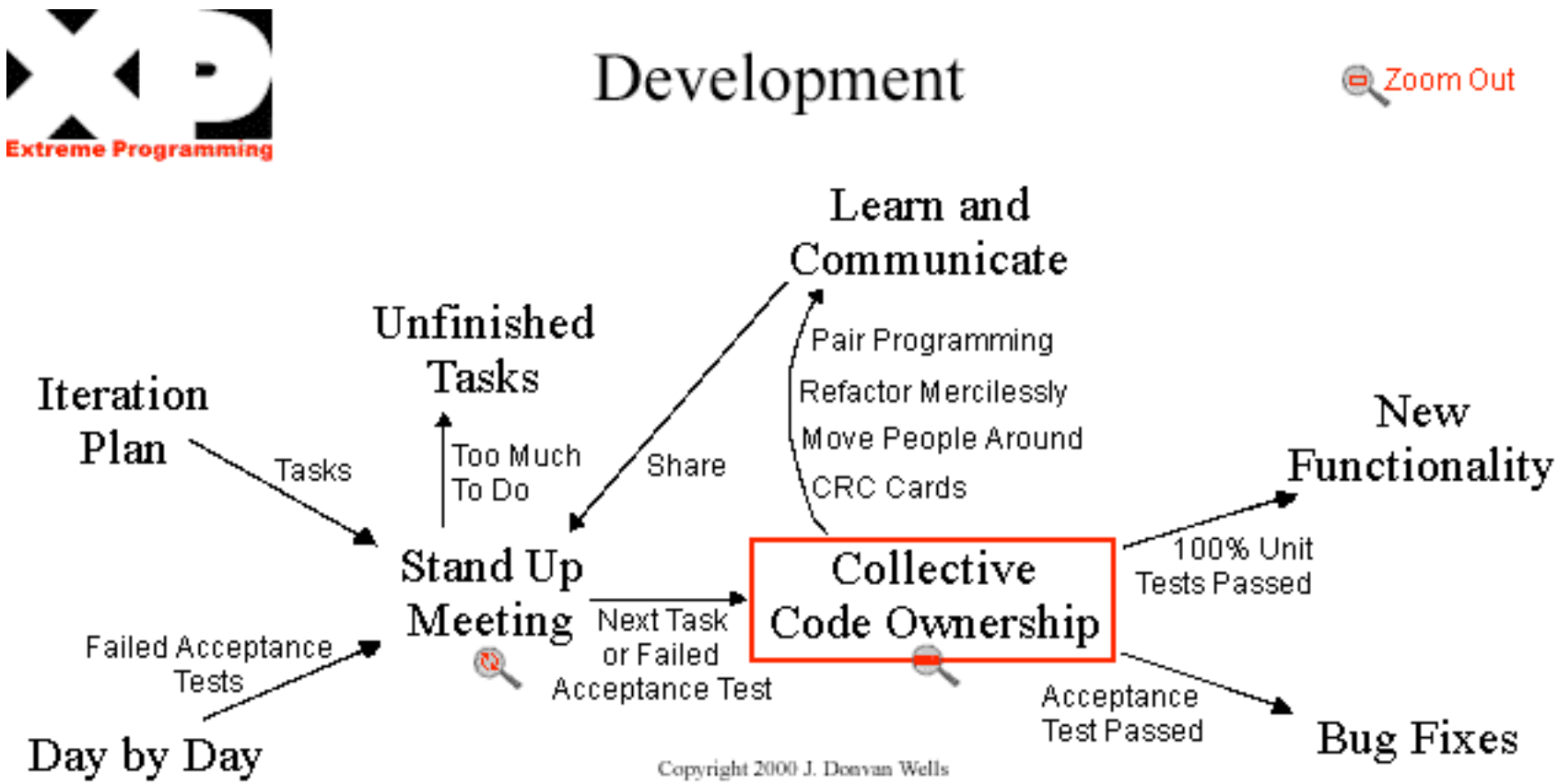
Iterationsplanung

- nur Entwickler
- aufteilen der Benutzergeschichten in einzelne Tasks
- Zeithorizont 1-4 Wochen (eine Iteration)

Iterationsplanung



Entwicklungsstrategie



Schlichtes Design

- Änderungen im Design der Normalfall
- geringstmöglicher Aufwand
- Über die Zeit weiterentwickelt

Fortlaufende Integration

- Nachdem eine Aufgabe vollständig implementiert wurde wird die Codeänderung sofort in das System integriert.
- Integration nur wenn Unit Tests 100%
- separater Arbeitsplatz

Kleine Releases

- Investition soll früh rentieren
- Anforderungen ändern sich während der Projektlaufzeit
- Projektverlauf einfacher zu beeinflussen

Collective Code Ownership

- Jeder Programmierer darf jedes Stück Code jederzeit editieren

Vorteile von Collective Code Ownership

- Komplexer Code schneller eliminiert
- Projektrisiko reduziert; geringere Abhängigkeit vom einzelnen Programmierer
- Wissen wird im Team verteilt
- Abwechslung für Programmierer

Pair Programming

- Dialog zwischen zwei Entwicklern, die zusammen programmieren, anlysieren, designen und testen.
- höhere Qualität
- höhere Produktivität
- Wissensunterschiede im Team reduziert (Paarweises Lernen)
- zufriedene Entwickler

Refactoring

- Vereinfachung des Code
- Gleiche Funktionalität
- Beispiel: “switch”-Statements ersetzen durch Polymorphismus
-

Martin Fowler: "Refactoring - Improving the Design of Existing Code",
Addison-Wesley, ISBN 0-201-48567-2

Kunde vor Ort

- Kunde Bestandteil des Programmierteams
- Beantwortet Detailfragen in Bezug auf die Customer Stories

Teststrategie

“The idea is that the developers are responsible for proving to their customers that the code works correctly; it's not the customer's job to prove that the code is broken.” -- Kent Beck

Ziele des Testens

- Nach Refactoring kann verifiziert werden, dass Änderung keine Nebeneffekte hat
- Kunde und Entwickler sind überzeugt von der Qualität

Unit Tests

- Interface einer Methode ist unklar
- Implementation ist kompliziert
- seltene Ausnahme im Code
- Refactoring steht an
- xUnit Testing Framework, http://xprogramming.com/software.htm

Funktionale Tests (Akzeptanztests)

- Kunde kann prüfen ob System funktioniert
- Programmierer sieht was fehlt
- Müssen bis zum Ende des Release nicht zu 100% durchlaufen

Rollenverteilung

- Programmierer - kommunikativ!
- Kunde - Teil des Entwicklerteams
- Tester - Akzeptanztests
- Coach - verteidigt XP Prinzipien

Lebenszyklus eines XP Projekts

- Exploration
- Planung
- Iterationen
- Pflege
- Tod

Grenzen von XP

- Unternehmenskultur
- Grosse Entwicklerteam
- Lange Feedbackzeiten

Diskussion