eXtreme Programming
מתודולוגיה לפיתוח פרויקטי תוכנה

ד"ר אורית חזן
המחלקה להוראת הטכנולוגיה והמדעים, הטכניון
oritha@techunix.technion.ac.il

eXtreme Programming
מתודולוגיה לפיתוח פרוייקטי תוכנה

- What is eXtreme Programming
- Why eXtreme Programming?
  - Social analysis
  - Cognitive analysis

eXtreme Programming
מתודולוגיה לפיתוח פרוייקטי תוכנה

- **על-סמך ניסיונכם עד היום בפיתוח תוכנה:**
  - **מהן הבעיות המרכזיות המאפיינות פרויקטי תוכנה?**

3

Problems in software development

- **Google: "problems with software development"**
  - Requirements are complex
  - Clients usually do not know all the requirements in advance
  - Requirements may be changing
  - Frequent changes are difficult to manage
  - Process bureaucracy (documents over development)
  - It takes longer
  - The result is not right the first time
  - It costs more
  - Applying the wrong process for the product

4

בעיות מאפיינות של פרוייקטי תוכנה

- **הסיבוכיות האמיתית בפיתוח תוכנה מקורה בהיבט האנושי (ולא הטכנולוגי)**
  - **לקוחות, לו"ז לחוץ ואי-עמידה בו, באגים, הבנה של תוכניות, תקשורת בין חברי צוות, אי-שיתוף מידע, אינטגרציה, ...**

5

פרויקטי תוכנה: נתונים

Size of project	Early	On-Time	Delayed	Cancelled	Sum
1 function point	14.68%	83.16%	1.92%	**0.25%**	100.00%
10 function points	11.08%	81.25%	5.67%	**2.00%**	100.00%
100 function points	6.06%	74.77%	11.83%	**7.33%**	100.00%
1,000 function points	1.24%	60.76%	17.67%	**20.33%**	100.00%
10,000 function points	0.14%	28.00%	23.83%	**48.00%**	100.00%
100,000 function points	0.00%	13.67%	21.33%	**65.00%**	100.00%
Average	5.53%	56.94%	13.71%	**23.82%**	100.00%

Table 1: Percentage of projects early, on-time, late, canceled
(from *Patterns of Software Systems Failure and Success,* by Capers Jones)

6

פרויקטי תוכנה: נתונים

- **75% ממוצרי התוכנה הגדולים הנשלחים ללקוחות נחשבים ככישלון: או שאינם בשימוש כלל או שאינם מתאימים לדרישות הלקוחות.**

Based on: Mullet, D. (July, 1999). The Software Crisis, Benchmarks Online - a monthly publication of Academic Computing Services **2**(7).

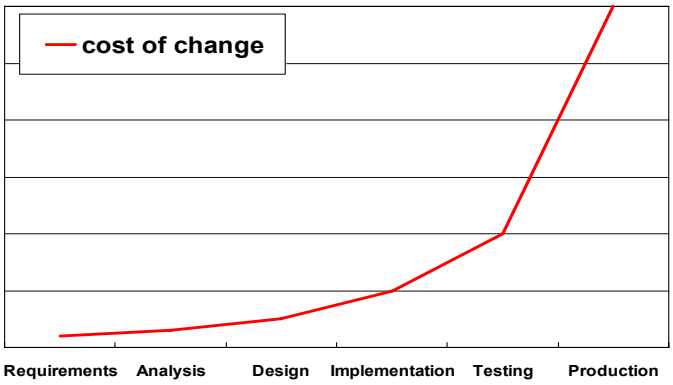
- **עלות תיקונם של באגים בתוכנה בארה"ב נאמדת בכל שנה ב- 59.5 ביליון $**

The National Institute of Standards and Technology (NIST), New Release of June 28, 2002.

- לשם השוואה: ב- Q2 של 2003 הושקעו בארה"ב בתוכנה 200 ביליון $

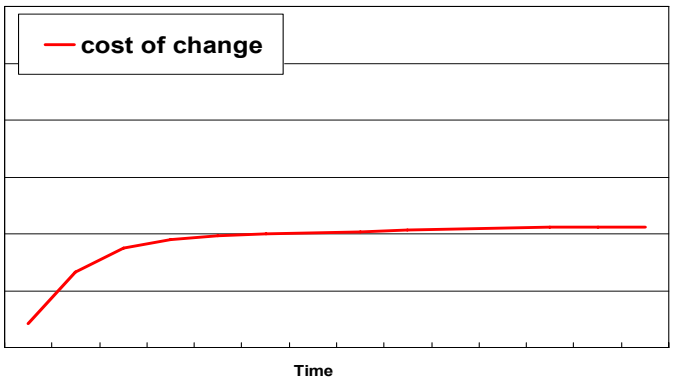
7

Economics of software development



8

What if…



9

What is eXtreme Programming

- eXtreme Programming צמחה בתעשייה.
- Differences from traditional methodologies
  - Emphasis on people vs. development activities & schedule
  - XP specifies how to behave; still leaves freedom
- 12 practices
- 4 values: feedback, simplicity, communication, courage
- The meaning of ‘eXtreme’
- Optimum: teams up to 12 developers; can be adjusted to bigger teams.

10

מקור השם

- ערכים ומיומנויות שהתגלו כאפקטביים נלקחים לקיצוניות

11

Why XP?

- Survey:
  - 31 XP/Agile-methods early adopter projects
  - 14 firms
  - Findings:
    - Cost reduction: 5-7% on average
    - Time to market compression: 25-50% reduction
      - This datum will be explained later

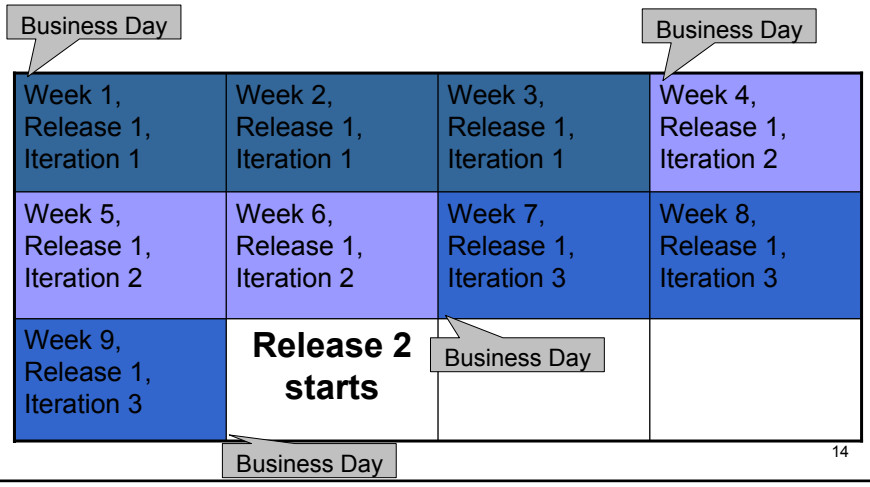
12

Why XP?

- big companies using XP in at least some capacity
  - Ford Motor, Chrysler, IBM, HP
- smaller software houses:
  - Mayford Technologies
  - RoleModel Software
- tutorials: Industrial Logic, Object Mentor

13

Project Timetable: 1 release - 3 iterations (2 months - 9 weeks)



14

ערכים

המתודולוגיה מושתתת על 4 ערכים:

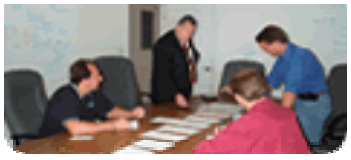
- משוב (feedback)
- פשטות (Simplicity)
- תקשורת (Communication)
- אומץ (Courage)

ערכים אלו מבוטאים ב 12 מיומנויות תוכנה, בימי עבודה משני סוגים

15

Business Day

- On-site customer
- Planning game
- Small releases
- Simple design
- Metaphor



Source: http://www.rolemodelsoftware.com/

16

Business Day – Reflection

- 5 practices (out of 12)
  - Planning game
  - On-site customer
  - Small releases
  - Simple design
  - Metaphor
- Planning game
  - All developers participate
  - All have the same load
  - All developers get an overview of the entire development process
  - Simple means
  - Very detailed
  - Levels of abstraction

17

Business Day – Reflection

- **5 practices** (out of 12)
  - Planning game
  - On-site customer
  - Small releases
  - Simple design
  - Metaphor
- On-site customer
  - Customer's on-going feedback
- Small releases
  - On-going opportunity to update/change requirements

18

Business Day – Reflection

- **5 practices** (out of 12)
  - Planning game
  - On-site customer
  - Small releases
  - Simple design
  - Metaphor
- Simple design
  - Develop only what is needed for your development task
- Metaphor
  - Bridges customers-developers-business gaps

19

Development Day

Source: http://www.rolemodelsoftware.com/

- Stand-up meeting
- The development environment
- Pair programming
- Test driven development (acceptance, unit-test)
- Code standards
- Refactoring
- Simple design
- Continuous integration (one integration machine)
- Collective ownership
- Sustainable pace (40-hour week)

20

סביבת העבודה

21

Development Day - Reflection

- The development environment
  - All see all; fosters communication
- Stand-up meeting
  - All know what all do
- Pair programming
  - Each task is thought on two levels of abstraction
- Unit test (automatic test first)
  - First: improves understanding; Automatic: testing is easy
  - Developers program and test
  - Testing becomes manageable
  - Success vs. failure

22

Development Day - Reflection

- Continuous integration
  - Reduces integration risks in later stages
- Collective ownership
  - Important in companies with high turnover
- Coding standards
- Refactoring and simple design
  - Code improvement is part of the methodology (though it doesn't produce code), gradual process
- Sustainable pace (40-hour week)
  - Intense and productive work, developers are not tired

23

Development and Business Days – Reflection

Code/Technical Perspective	Human/Social Perspective
Refactoring	Collective ownership
Simple design	Pair programming
Coding standards	Sustainable pace
Testing	On-site customer
Continuous integration	Planning game
Small releases	Metaphor

24

The 12 XP practices

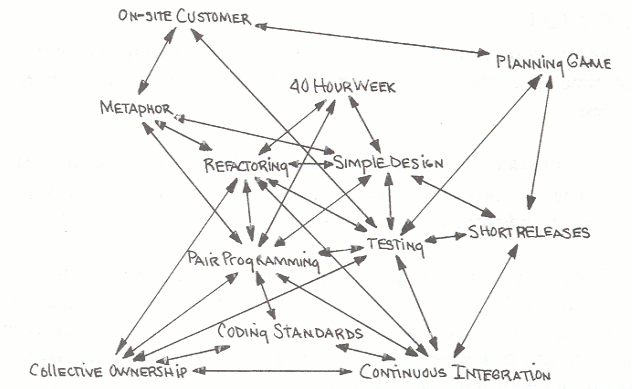


FIGURE 4. The practices support each other

Note:
nothing is new; gathering the practices together is XP uniqueness

Source: Beck, K. (2000). *eXtreme Programming explained*, Addison Wesley.

25

What is eXtreme Programming

- Differences from traditional methodologies
  - All developers are involved with requirements-design-code-testing
  - Emphasis on people vs. development activities & schedule
  - XP specifies how to behave; still leaves freedom and place for creativity
- The meaning of 'eXtreme'
- 12 practices
- 4 values: feedback, simplicity, communication, courage

26

What is eXtreme Programming

- Agile Software Development Methodology
  - **Other agile methods:** SCRUM, Feature Driven Development, DSDM
  - All acknowledge that the main issue of software development is people: customers, communication
- Manifesto for Agile Software Development: http://agilemanifesto.org/
- eXtreme Programming: Kent Beck, 1996, Chrysler

27

Why XP?

- You do not do XP to save money;

  However, XP shortens time to market
- XP is a mature software development method

28

Why XP?

- Survey:
  - 31 XP/Agile-methods early adopter projects, 14 firms
  - Findings:
    - Cost reduction: 5-7% on average
    - Time to market compression: 25-50% reduction in time

29

Why XP? – Analysis

- **Shorter development period:**
  - Code is easy-to-work with:
    - less bugs: unit tests
    - code is more readable & workable (invest now to gain benefits later):pair programming, refactoring, coding standards
  - Development is manageable and controlled:
    - accurate estimation: small releases
    - meets customer needs: customer on-site, planning game, acceptance tests

30

Why XP? – Analysis

- **Shorter development period (cont):**
  - Knowledge sharing, if one leaves everything continues as usual: pair programming, collective ownership
  - Production is increased: pair programming (work all the time), sustainable pace
  - Cost for requirements change/update/elaboration is CONSTANT: simple design, planning game (redundant features are not added by customer and developers)

31

Why XP?

Barry W. Boehm (1981). *Software Engineering Economics,* Englewood Cliffs, N.J.: Prentice Hall.

- 63 software development projects in corporations such as IBM.

Phase of requirement change	Cost Ratio
Requirements	1
Design	3-6
Coding	10
Development testing	15-40
Acceptance testing	30-70
Operation	40-1000

32

Why XP?

- Under the assumption that "the later a requirements is introduced the more expensive it is", customers (and developers) try to make a "complete" list of requirements.
- Under the assumption that "cost for introducing an update in the requirements is constant", customers (and developers) do not assume what the customer will need and develop exactly and only what is needed.

33

Why XP?

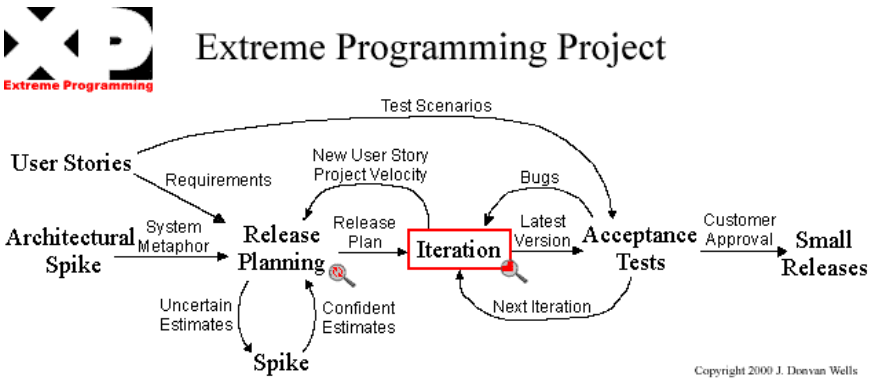
- You do not use XP to save money; However, XP shortens time to market
- XP is a mature software development method (at least CMM level 3)

34

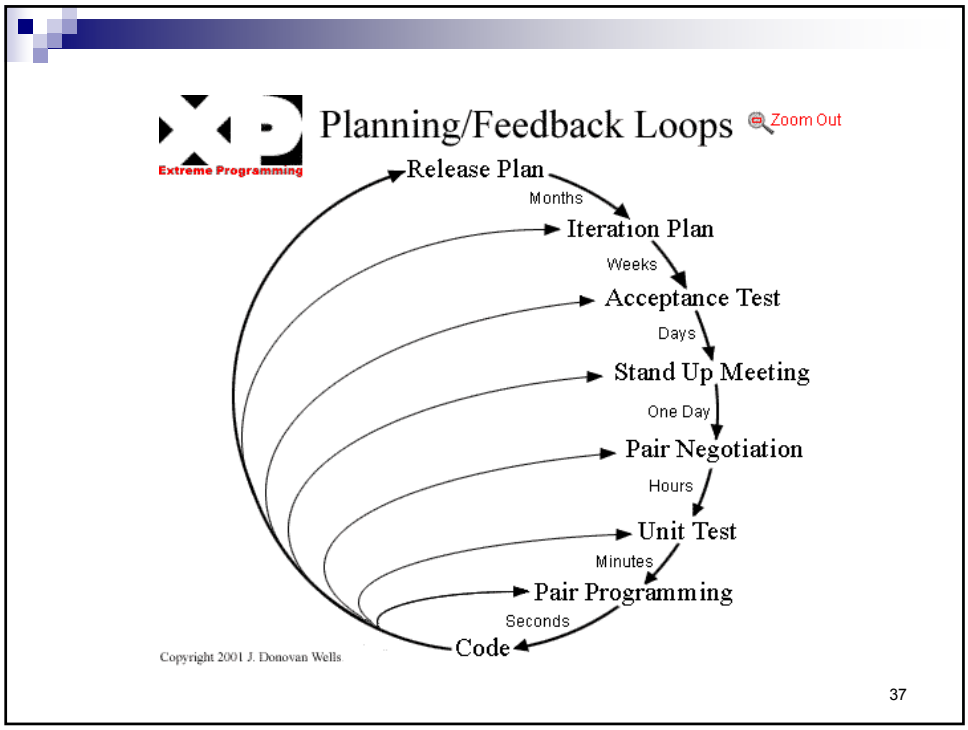
XP in Practice: Conceptual Changes

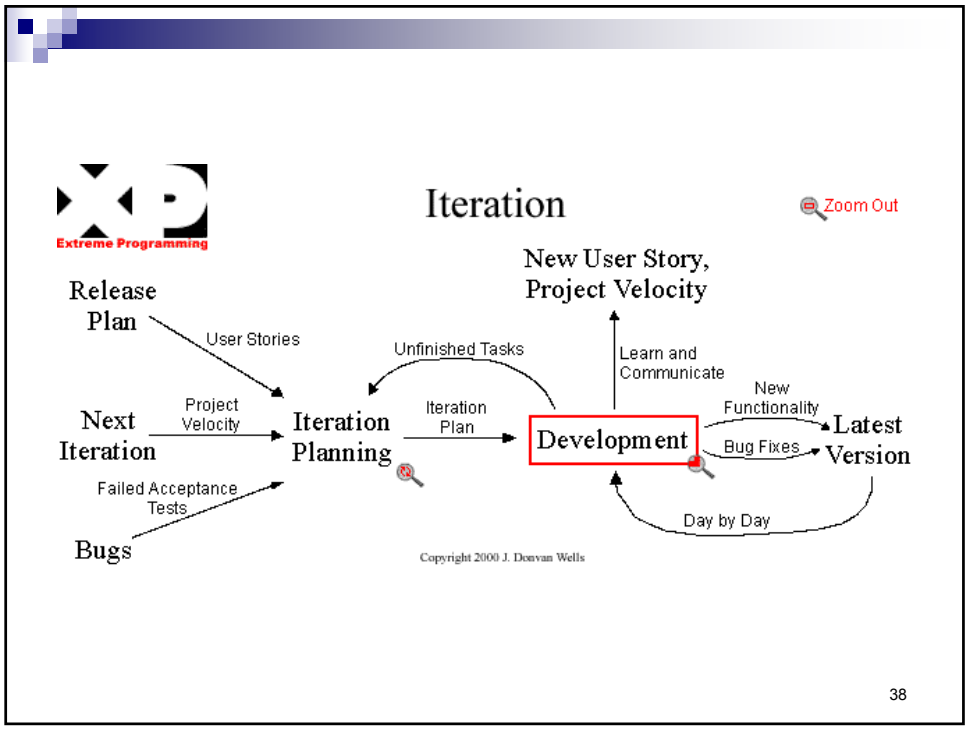
- **XP encourages:**
  - Cooperation (**vs.** knowledge-is-power)
  - Simplicity (**vs.** habit-of-high-complexity)
  - Change in work habits

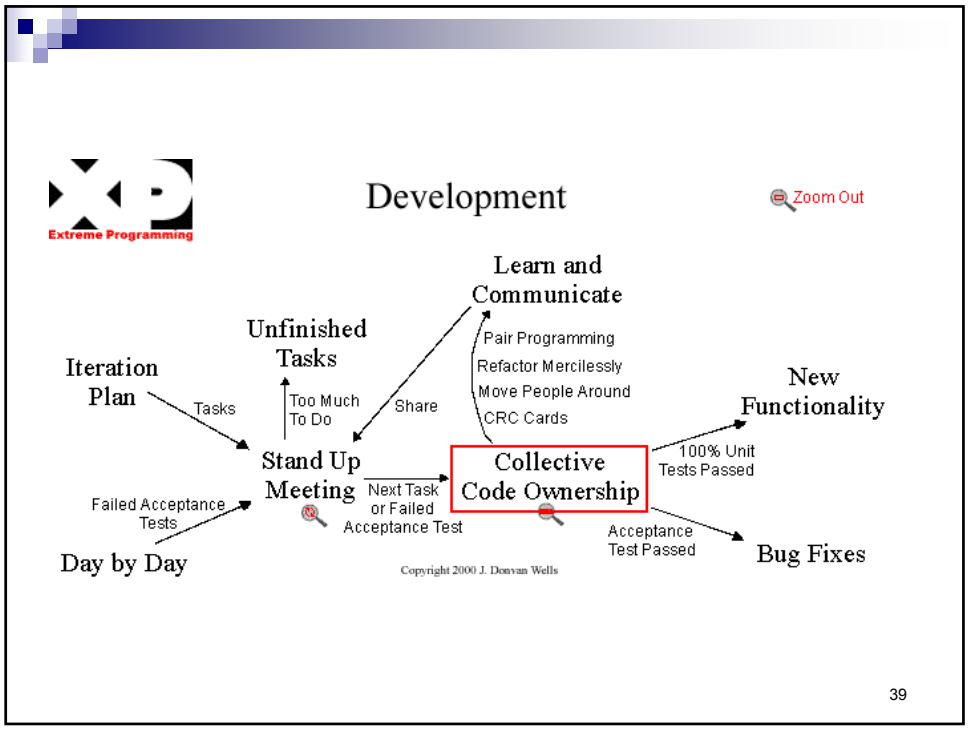
35



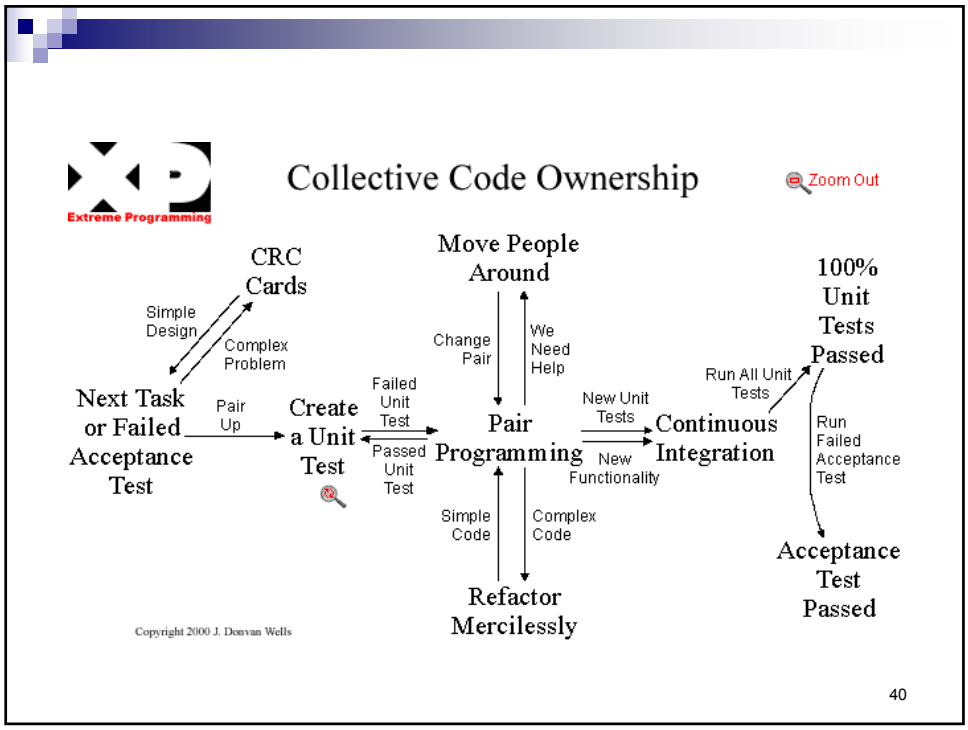
36


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
37


Extreme Programming
Iteration
Zoom Out
New User Story,
Project Velocity
Release
Plan
User Stories
Unfinished Tasks
Learn and
Communicate
New
Functionality
Next
Iteration
Project
Velocity
Iteration
Planning
Iteration
Plan
Development
Bug Fixes
Latest
Version
Failed Acceptance
Tests
Bugs
Day by Day
Copyright 2000 J. Donvan Wells
38

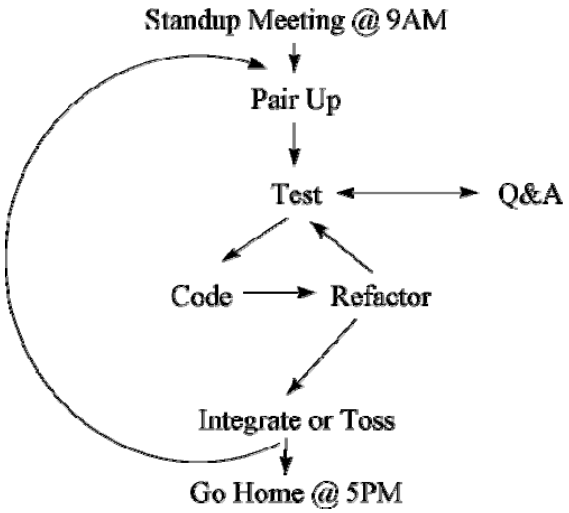

Extreme Programming
Development
Zoom Out
Learn and Communicate
Unfinished Tasks
Pair Programming
Refactor Mercilessly
Move People Around
CRC Cards
Iteration Plan
Tasks
Too Much To Do
Share
New Functionality
Stand Up Meeting
Next Task or Failed Acceptance Test
Collective Code Ownership
100% Unit Tests Passed
Failed Acceptance Tests
Acceptance Test Passed
Day by Day
Bug Fixes
Copyright 2000 J. Donvan Wells

39


Extreme Programming
Collective Code Ownership
Zoom Out
CRC Cards
Move People Around
100% Unit Tests Passed
Simple Design
Complex Problem
Change Pair
We Need Help
Run All Unit Tests
Next Task or Failed Acceptance Test
Pair Up
Create a Unit Test
Failed Unit Test
Passed Unit Test
Pair Programming
New Unit Tests
New Functionality
Continuous Integration
Run Failed Acceptance Test
Simple Code
Complex Code
Acceptance Test Passed
Refactor Mercilessly
Copyright 2000 J. Donvan Wells

40

A Day in the Life…



41

Why XP? – Cognitive and Social Analysis

- ניתוח XP מנקודת מבט קוגניטיבית וחברתית.
  - Game Theory: Prisoner's Dilemma
  - Learning Theory: Constructivism

42