# חלק 17
# eXtreme Programming
## מתודולוגיה לפיתוח פרויקטי תוכנה

מצגת – עפ"י ד"ר אורית חזן, הטכניון

# 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

# 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, ...**

# 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 $

# What is eXtreme Programming

- eXtreme Programming originated in industry.
- 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
- 12 practices
- 4 values: feedback, simplicity, communication, courage
- The meaning of ‘eXtreme’
- Optimum: teams up to 12 developers
  - can be adjusted to bigger teams.

# 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

# 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

# Project Timetable

- Short release times - each one 9 weeks.
- A release has three iterations.
- An iteration lasts 3 weeks.
- Each iteration starts with a business day.
- Rest of the days are development days

# Project Timetable: one release

# Business Day

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

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

# 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

# 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

# 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

# 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)

# 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

# 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

# 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 |

# The 12 XP practices

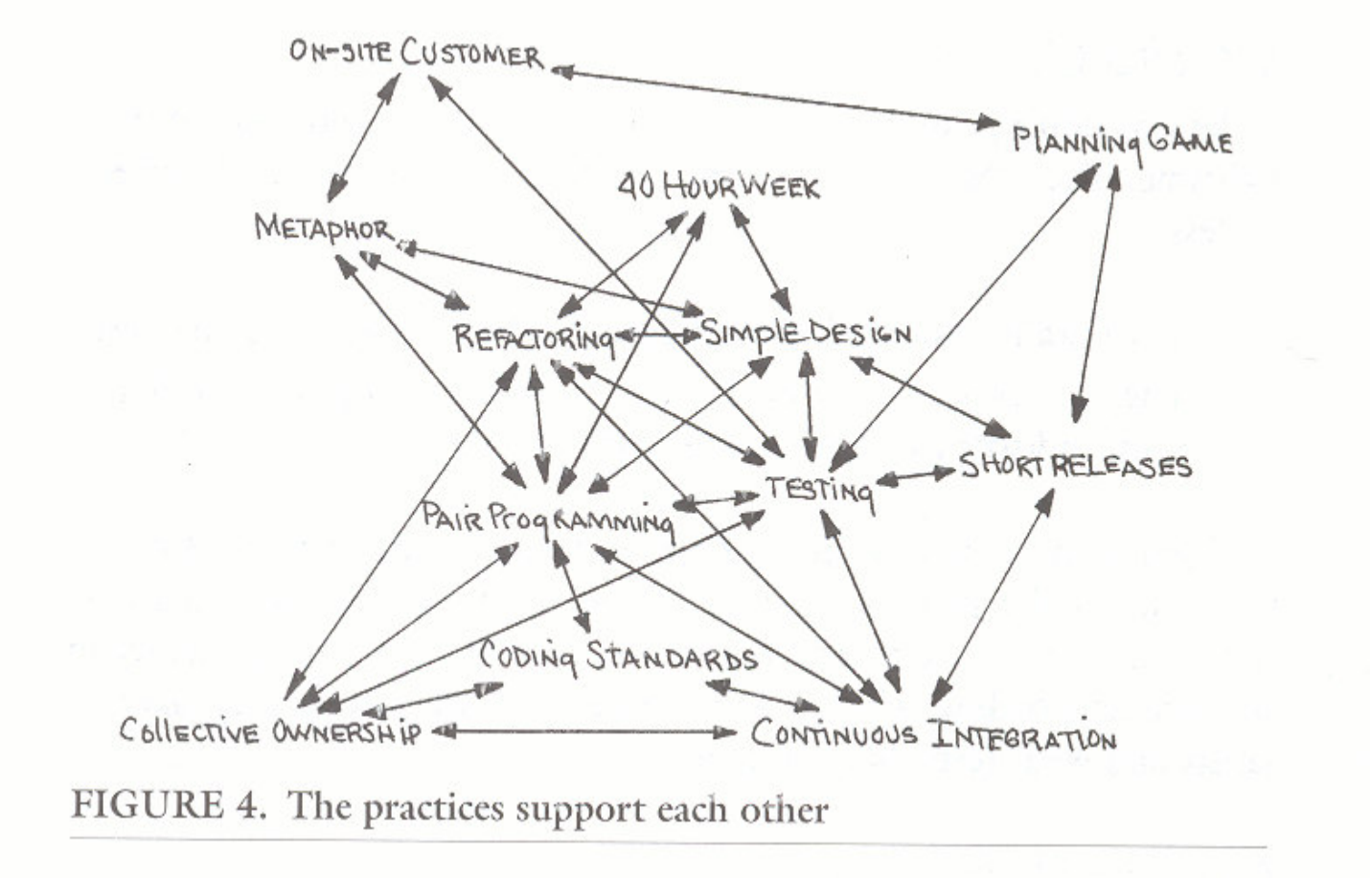


FIGURE 4. The practices support each other

**Note:**

nothing is new; gathering the practices together is XP uniqueness

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

# 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

# Why XP?

- You do not do XP to save money; However,

  XP shortens time to market

- XP is a mature software development

  method

# 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

# 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)

# 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 |

# 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.

# XP in Practice: Conceptual Changes

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

# References

Beck, K. (2000). *Extreme Programming Explained: Embrace Change*, Addison Wesley.

Ron Jeffries, What is Extreme Programming?: http://www.xprogramming.com/xpmag/whatisxp.htm

eXtreme Programming at the Technion

RoleModel:http://www.rolemodelsoftware.com/process/whatIsXp.php

# XP in practice:
# Success and failure

3 Sep, 2002: XP - An interview with Kent Beck

**Q:** What are the issues you see your clients struggling with?

**KB:** One of the issues is **redefining failure or redefining success**. For example, you think that you have a great idea for a project, and it's going to take you nine months to really have it ready for the market. You [may] discover after four weeks that you are going one-tenth the speed that you thought you would, and you cancel the project. Is that a failure or success? In many organizations, this is perceived as a failure.

# XP in practice:
# Success and failure

3 Sep, 2002: XP: An interview with Kent Beck

**KB (cont'):** In the XP world, providing information that allows you to constantly make that decision after four or eight weeks (out of a nine-month development cycle) is what you're there for. In the XP world, you call that a dramatic success. Now you have this cultural mismatch between how outsiders are going to view your outcome and how you view it inside the team. A lot of people [clients] struggle with that. They think that canceling a project is a bad thing, but I think that canceling a project is a good thing -- **as long as you cancel the right one.**