This Lecture

- Introduction to Software Engineering
  - The Software Crisis
  - Methodologies
- Extreme Programming
  - Agile Development Principles
  - XP Practices
- Software Architectures
- Course Summary

July 1, 2006 Object Oriented Design Course 2

Intro to Software Engineering Extreme Programming Course Summary

David Talby

The Software Crisis II

- Small software projects take on average 100% more time and resources than planned

  Based on: Standish Group, '94
- Large projects are one year late on average

  Based on: Jones, '94
- Periods of excessive time pressure occur in 75% of large projects
- 60% of programmers report excessive time pressure

  Based on: Glass, '94 and Jones, '94

July 1, 2006 Object Oriented Design Course 4

The Software Crisis

- 75% of the enterprise software products are considered to be a failure – they are not in use or do not fit customer's requirements

  Based on: Mullet, D. (July, 1999). The Software Crisis, Benchmarks Online - a monthly publication of Academic Computing Services **2**(7).
- Software errors cost US economy $59.5 billion annually (In Q2 of 2003 – investments of $200 billion in software development)

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

July 1, 2006 Object Oriented Design Course 3

What is Software Engineering?

- Repeatability
  - Ability to do a similar project again well
  - Same time, budget, quality are expected

July 1, 2006 Object Oriented Design Course 6

The Software Crisis III

- In Words
  - Most software is totally unusable
  - Most software is buggy, unstable and insecure
  - Most software development plans are unreliable
  - Yet, software runs the world

July 1, 2006 Object Oriented Design Course 5

What is Software Engineering? III

- Legal Liability
  - Both Civil and Criminal
  - Certification required for life-critical issues
  - Methodology & Notation are laws

What is Software Engineering? II

- Methodology
  - Well-defined roles: Architect, Engineer, Tester, Deployer, …
  - Well-defined products: Designs, Specs, Code, Test scenarios, …
  - Standard workflow of how things are done

Development Methodologies

- A methodology describes
  - An entire life cycle of a software product
  - Roles, Products, Workflow
  - Best Practices
- Classic methodologies: Linear
  - Linear/Incremental: Waterfall model
  - Iterative: Rational Unified Process
- Modern methodologies: Agile
  - Adaptive
  - Extreme

State of the Software World

- Large Scale
  - Lack of repeatability, even for small projects
  - Inability to provide quality software
  - No standard definition of roles & products
  - No standard for requirements, design, tests, …
  - It's a "wild west" profession
- Small Scale
  - Developers don't produce working software
  - Developer tools are also far from perfect

Extreme Programming

Slides courtesy of
Yael Dubinsky

Rational Unified Process

- By Rational, see rational.com/rup
- Iterative Development
- Decompose large system to sub-systems
  - A team and development effort per system
  - Architects Team does overall design, sharing
- Five stages of each system's life cycle
  - Business modeling, Requirements, Analysis & Design, Implementation, Test
  - Many artifacts are not code or tests
- Highly managed, highly automated process

Agile Software Development Methods

- Adaptive
- Crystal
- DSDM
- Extreme Programming
- FDD
- Lean Development
- Scrum

All acknowledge that the main issue of software development is people: customers, communication

The Agile Manifesto

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

http://agilemanifesto.org/

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

Reifer, D. (2002). How to get the most out of Extreme Programming/Agile methods? *Proceedings of the XP/Agile Universe 2002*, pp. 185-196.

What is Extreme Programming?

- Extreme Programming emerged by software practitioners
- Differences from traditional methodologies
  - Emphasis on people vs. development activities & schedule
- 12 practices
- 4 values: feedback, simplicity, communication, courage
- The meaning of 'Extreme'

"XP is a lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop software." Beck, 2000

"XP is about social change." Beck, 2nd edition, 2005

Beck, K. (2000). Extreme Programming explained, Addison Wesley.

Business Day

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

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

How XP? - Project Timetable:

1 release (2 months), 4 iterations (2 weeks each)

Week 1, Release 1, Iteration 1	Week 2, Release 1, Iteration 1	Week 3, Release 1, Iteration 2	Week 4, Release 1, Iteration 2
Week 5, Release 1, Iteration 3	Week 6, Release 1, Iteration 3	Week 7, Release 1, Iteration 4	Week 8, Release 1, Iteration 4
Release 2 starts			

Business Day

Business Day

Business Day

Business Day

Business Day

Extreme Programming Practices

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

Development Day

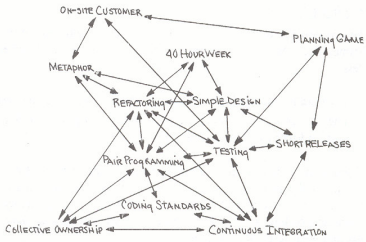
- Stand-up meeting
- The development environment
- Pair programming
- Testing (Test Driven Development)
- Code standards
- Refactoring
- Simple design
- Continuous integration
- Collective ownership
- Sustainable pace

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

Agile/XP in Practice: Conceptual Changes

- **XP encourages changes in culture:**
  - Cooperation (**vs.** knowledge-is-power)
  - Simplicity (**vs.** habit-of-increase-complexity)
  - Change in work habits
  - Change in customer's conception, involvement and relationships with the software team
- **Therefore, XP is not easy to implement**
  - Match to project and team must be verified

XP practices support each other



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

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

Software Architectures and Course Summary

David Talby

Summary

- Writing Software ≠ Delivering Products
  - Requirements, Architecture, Design, Code, Integrate, Test, Deploy, Maintain, Update
- The Software Crisis
- Beware of which process you work in

Selected Topics

- Programmer's Day-to-Day Skills
  - Design Patterns
  - Tools and work habits
  - Language features
- Prepare for the future
  - Framework and components
  - Development methods
- And now to a few subjects we didn't cover…

Why Object Oriented Design?

- Software Today
  - < 20% of existing code is Object-Oriented
  - > 90% of new code is Object-Oriented
  - Reuse: Libraries, Components, Web Services
  - Major Frameworks/Platforms: Java and .NET
  - A lot developed on non-traditional platforms

Platforms II

- Rich Client Web Applications
  - AJAX, Struts, …
- Workflows & Business Process Management
  - Oracle BPEL
  - Microsoft WWF
  - Business Rule Engines
- Office
  - Tools for Office
  - Smart Documents

Platforms

- Cellular Phones
  - J2ME, Windows Mobile
- Enterprise Application Servers
  - J2EE, .NET
- Web Applications
  - JSP + Additions, ASP.NET
- Small Devices
  - Palm Pilot, Windows CE, PlayStation
- Media Centers

Frameworks II

- Security
  - Identity Management (JAAS, LDAP, Biometrics)
  - Digital Rights Management
  - Mal-ware: Classification, Constructions, Detection
- Remote Monitoring & Management
- Google
  - Site Search, AdSense, …
- UI Component Frameworks
  - Delphi vs. Swing vs. .NET
- Web Services (WSE 3.0, …)

Frameworks

- Desktop Search Engines
  - Architecture, API
- Development Platforms
  - Eclipse, Net Beans, Visual Studio
- Database Access
  - JDO, Hibernate, ADO.NET
- Game Development
  - OpenGL, DirectX, Graphic Engines
- Enterprise Portals
  - SharePoint, IBM

Development Tools

- Model Driven Development
  - UML Based (Together, …)
  - DSL Based (Visual Studio, …)
- Code Generation
- Testing
  - Automatic unit and acceptance testing
  - Load testing
- Open Source
- Team Development Tools
  - Configuration Control, Defect Tracking, …

A few final words…

- Use this material
- Keep learning
- Keep in touch

- and Good Luck!