Agile methodologies, XP and TDD

maurizio.morisio@polito.it
http://softeng.polito.it

SoftEng

Outline

- Agile methodologies
- XP
- Test Driven Development
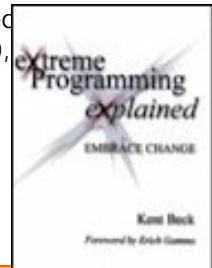- TDD Experiment
- XP + CMM case study

SoftEng

eXtreme Programming

SoftEng

Extreme programming

- Kent Beck: Extreme Programming Explained Addison-Wesley, 2000, ISBN 0-201-61641-6

SoftEng

Fundamentals of XP

- Distinguish between decisions made by business stakeholders and developers
- Simplistic – keep design as simple as possible
    - "design for today not for tomorrow"
- Write automated test code before writing production code and keep all tests running
- Pair programming
- Very short iterations with fast delivery

SoftEng

Why is XP controversial?

- No specialists – every programmer participates in architecture, design, test, integration
- No up-front detailed analysis and design
- No up-front development of infrastructure
- Not much writing of design & implementation documentation beside tests and code

SoftEng

Some basic facts

- Producing code is required to deliver a system
- Dollars spent on analysis and design are wasted if the system is never used
- Business requirements have to be the drivers for software development
- Requirements change

SoftEng

Back to the basics

- Coding
- Testing
- Listening
- Designing

SoftEng

Four values

- Communication
  - "problems with projects can invariably be traced to somebody not talking to somebody else about something important" p 29
- Simplicity
  - "what is the simplest thing that could possibly work?"
- Feedback
  - Put system in production ASAP
  - "Have you written a test case for that yet?"
- Courage
  - Hill climbing (simple, complex, simpler,..)
  - Big jumps take courage

SoftEng

The key practices

SoftEng

12 practices

- Customer satisfaction
  - On-site customer
  - Small releases
- Software quality
  - Metaphor
  - Testing
  - Simple design
  - Refactoring
  - Pair programming
- Project management
  - Planning game
  - Sustainable development
  - Collective code ownership
  - Continuous integration
  - Coding standards

SoftEng

On-site customer

- Many software projects fail because they do not deliver software that meets business needs
- Real customer has to be part of the team
  - Defines business needs
  - Answers questions and resolves issues
  - Prioritizes features

SoftEng

maurizio.morisio@polito.it

Small releases

- Put system into production ASAP
  - Fast feedback
- Deliver valuable features first
- Short cycle time
  - Planning 1-2 months is easier than planning 6-12 months

SoftEng

Metaphor/Architecture

- How does the whole system work?
- What is the overall idea of the system?
- Initially: Architectural spike

SoftEng

Simple design

- The "right" design
  - Runs all tests
  - No code duplication
  - Fewest possible classes and methods
  - Fulfills all *current* business requirements

Design for today not the future

SoftEng

Refactoring

- Restructure system without changing the functionality
- Goal: Keep design simple
  - Change bad design when you find it
  - Remove dead code

SoftEng

Pair programming

- "All production code is written with two people looking at one machine"
  - Person 1: Implements the method
  - Person 2: Thinks strategically about potential improvements, test cases, issues
- Pairs change all the time
- Advantages
  - No single expert on any part of the system
  - Training on the job
  - Permanent inspections
- Problems:
  - Wasted development time?
  - Pairs need to function

SoftEng

Pair programming - effects

- More quality and
- Less productivity?

SoftEng

Williams

- Williams, Laurie, Kessler, Robert R., Cunningham, Ward, and Jeffries, Ron, *Strengthening the Case for Pair-Programming*, *IEEE Software,* July/Aug 2000 .
  - University study with 41 students
  - Higher quality code
    - Test cases passed individuals: 73.4%-78.1%
    - Test cases passed pairs: 86.4%-94.4%
  - Pairs completed assignments 40-50% faster (average 15% higher costs)
  - Pair programming preferred by students (85%)

SoftEng

Long (on 5 studies)

- Quality
  - Better quality PP than solo programmers
  - Meaningful effect
  - Both with students and professionals
  - Improves quality of program without impacting quality of programmer
    - Average programmers benefit from it
- Not difficult to learn
  - May be more difficult for more skilled programmers

SoftEng

Long (on 5 studies)

- Productivity
  - PP lower productivity than solo programmer
  - Meaningful effect

  - One study suggests that PP may have same productivity in context of difficult algorithms – changing requirements

SoftEng

TDD

- *Automatic* test drivers
- Write tests before production code
  - Unit tests → developer
  - Feature/acceptance tests → customer
- Strong emphasis on regression testing
  - Unit tests need to execute all the time
  - Tests for completed features need to execute all the time
- Unit tests pass 100%
- Acceptance tests show progress on user stories

SoftEng

The planning game

- Business decisions
  - Scope: which "stories" should be developed
  - Priority of stories
  - Composition of releases
  - Release dates
- Technical decisions
  - Time estimates for features/stories
  - Elaborate consequences of business decisions
  - Team organization and process
  - Scheduling

SoftEng

Sustainable development

- Developing full speed only works with fresh people
- Working overtime for two weeks in a row indicates problem

SoftEng

Collective ownership

- All code can be changed by anybody on the team
- Everybody is required to improve any portion of bad code s/he sees
- Individual code ownership tends to create experts

SoftEng

Continuous integration

- Integration happens after a few hours of development
  1. Code is released into current baseline on integration machine
  2. All tests are run
  3. In case of errors:
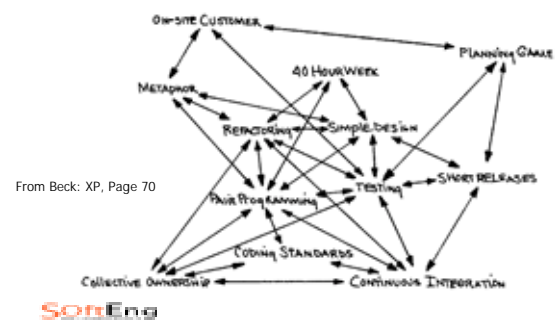   - Reverse to old version
     - Fix problems
     - Goto (1)

SoftEng

Coding standards

- Team has to adopt a coding standard
  - Makes it easier to understand other people's code
  - Avoids code changes because of syntactic preferences

SoftEng

How everything fits together

From Beck: XP, Page 70

SoftEng

Issues in XP adoption

SoftEng

All techniques?

- Proposers state that combination of all techniques provide highest benefit
- Stepwise adoption
  - Pick your worst problem and apply corresponding XP technique

SoftEng

Business contracts

- Fixed scope/fixed price contracts problematic – why?
- Fixed cost and fixed programmer hours

SonEng

Colocation and project size

- Co-location of team members required
- Scalability of the process:
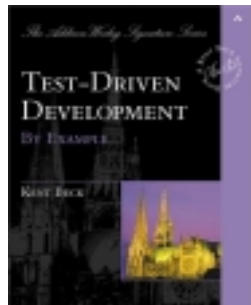 Small teams → small projects

SonEng

When can XP be used?

- Small projects:
  - 2-10 developers, maybe 20
- Developer and customer representative are co-located
- Problems:
  - Point-and-go culture
  - Testing takes hours to execute
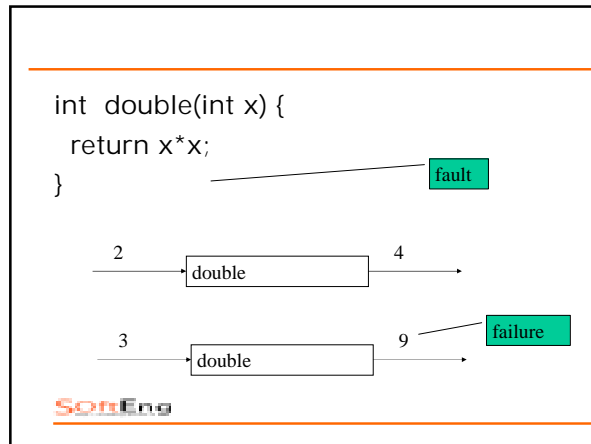
SonEng

Test Driven Development

SonEng

SonEng

Process
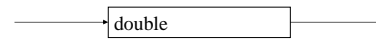
- Given a story (requirement)
  - Write a test so that it fails
  - Write production code to have test succeed
    – Minimize time to do so
  - Iterate
  - If something smells, refactor

- Tools:
  - JUnit
  - Todo list

SonEng

```
int  double(int x) {
  return x*x;
}
```

fault

2 → double → 4

3 → double → 9   failure

SoftEng

Test cases

- In 2, out 4
- in 3, out 6

→ double →

SoftEng

Test cases as code

```
testDouble2(){
  assertEquals(4, double(2));
}      // PASS
```

```
testDouble3(){
 assertEquals(6, double(3));
}     // FAIL
```

SoftEng

JUnit

SoftEng

Junit

- Test framework
  - plug-in for Eclipse

▪ Test code
testDouble2(){
 	
}
	▪ Production code
	int double(){
	
	}

SoftEng

Framework elements

- assert*()

- TestCase
  - class, contains tests (using asserts)
- TestSuite
  - class contains TestCases

SoftEng

Assert*()

- condition
  - assertTrue("message when test fails", condition);
- Return values (object, int, longs, byte)
  - assertEquals(expected_value, expression);
- Return values (float, double):
  - assertEquals(expected_value, expression, error);
- If condition tested
  - Is true, executes next statement
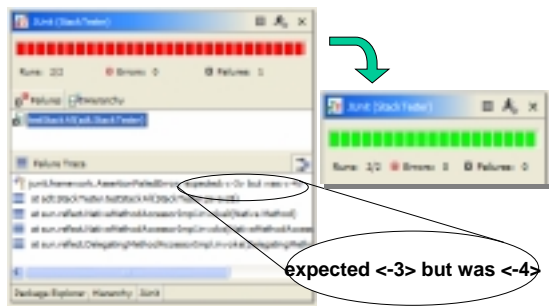 - Is false, break to end of method

SoftEng

Run as JUnit Test

- Run
- Run As..
- Junit Test
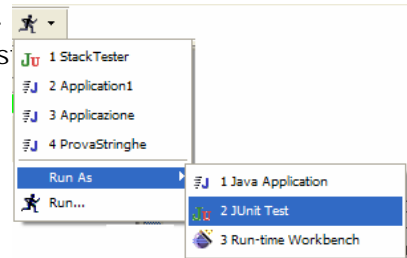
SoftEng

Red / Green Bar

expected <-3> but was <-4>

Test Driven Development

Ex. Supermarket Checkout

SoftEng

Checkout

- Software part of checkout system in supermarket, grocery store
- No GUI, no bar code reader

SoftEng

Story 1 – Price list

- Read bar code and print price
  - When a customer arrives, items codes are read (by simulated bar code reader)
  - Price is printed

```
> P001
3.0
```

```
> P001
3.0
> P002
1.5
```

SoftEng

Story 2 – Total

- On Close command print total and close session

```
> P001
3.0
> P002
1.5
> P001
3.0
> CLOSE
Total: 7.5
```

SoftEng

Story 3 – Items description

- Print item description and price
  - When item code is read, printout price and description

```
> P001
Anchovy 3.0
> P002
Garlic 1.5
> P001
Anchovy 3.0
> CLOSE
Total: 7.5
```

SoftEng

Story 4 – taxes

- At end of session print total with and without TVA
- TVA is the same for all items

```
> P001
Anchovy 3.0
> P002
Garlic 1.5
> P001
Anchovy 3.0
> CLOSE
Before TVA: 6.25
  TVA 20%: 1.25
    Total: 7.5
```

SoftEng

…

- Story 5 – Discount total
- Story 6 -Print list of items and total
- Story 7 – items with discount
- Story 8 – items 3X2
- Story 9 – TVA rate varies per item

SoftEng