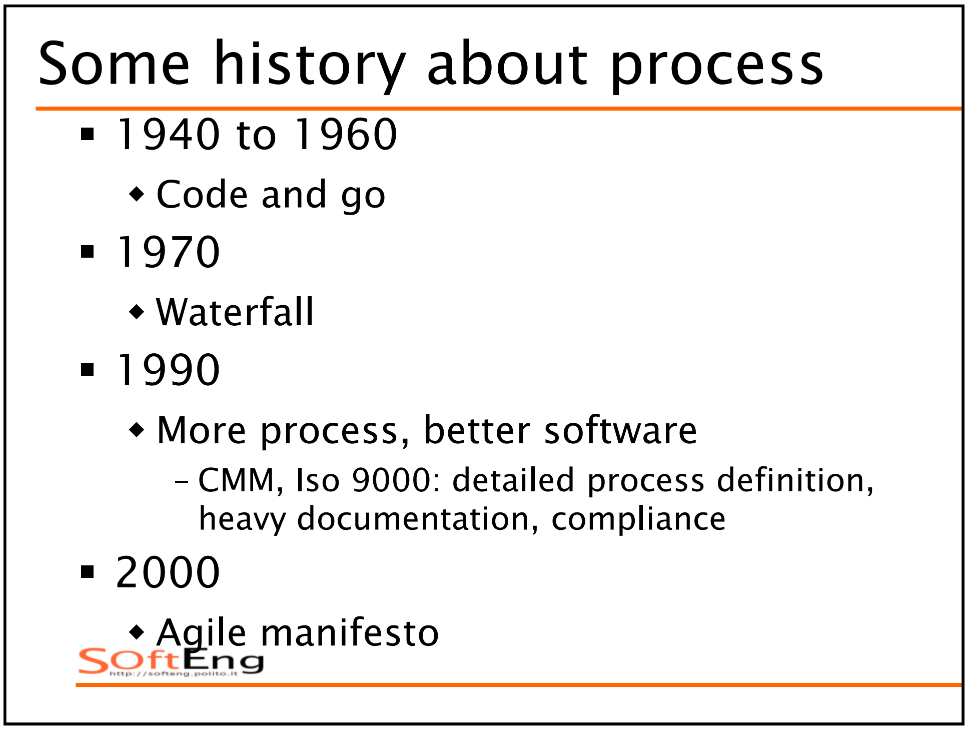
Agile methodologies: Scrum, XP

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

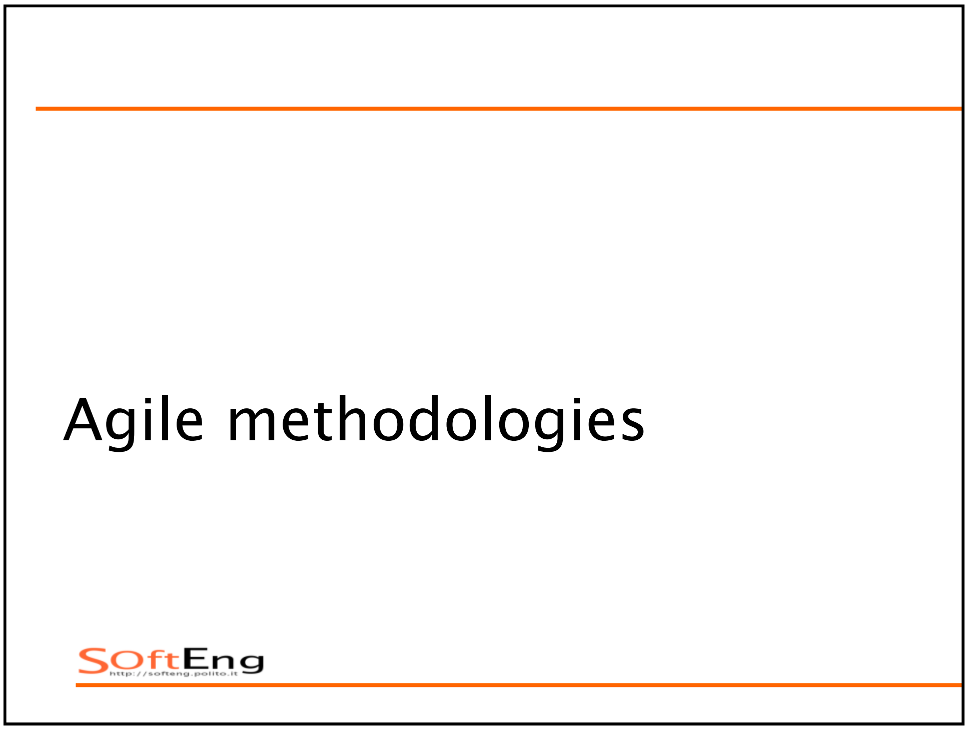
Outline

- Agile methodologies
- Scrum
- XP
- Test Driven Development



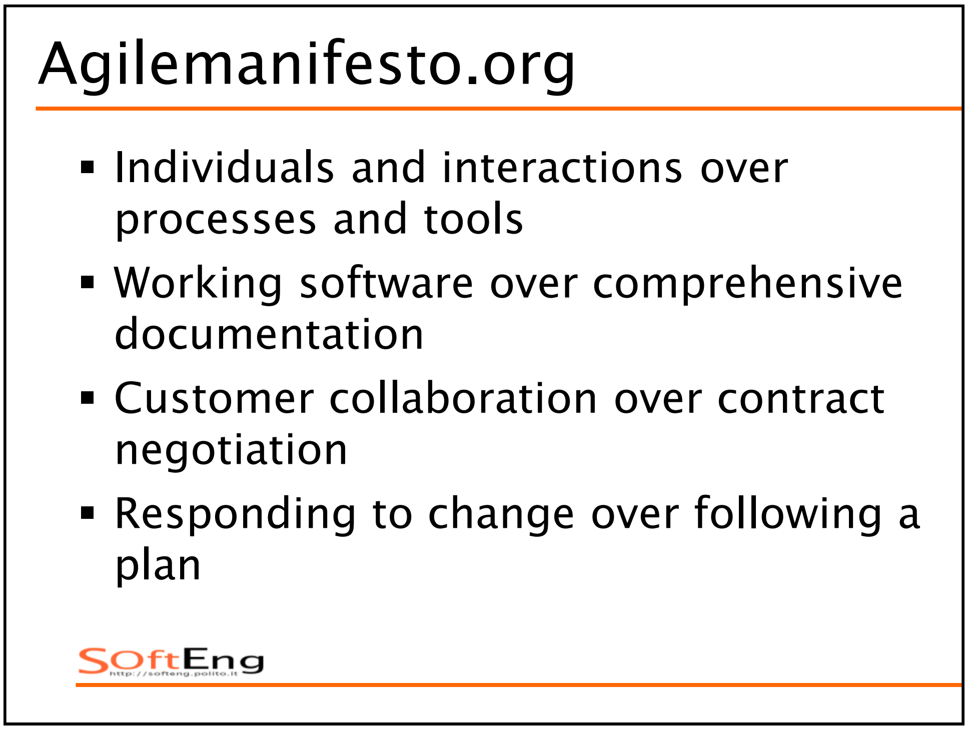
Some history about process

- 1940 to 1960
  - Code and go
- 1970
  - Waterfall
- 1990
  - More process, better software
    - CMM, Iso 9000: detailed process definition, heavy documentation, compliance
- 2000
  - Agile manifesto



Agile methodologies

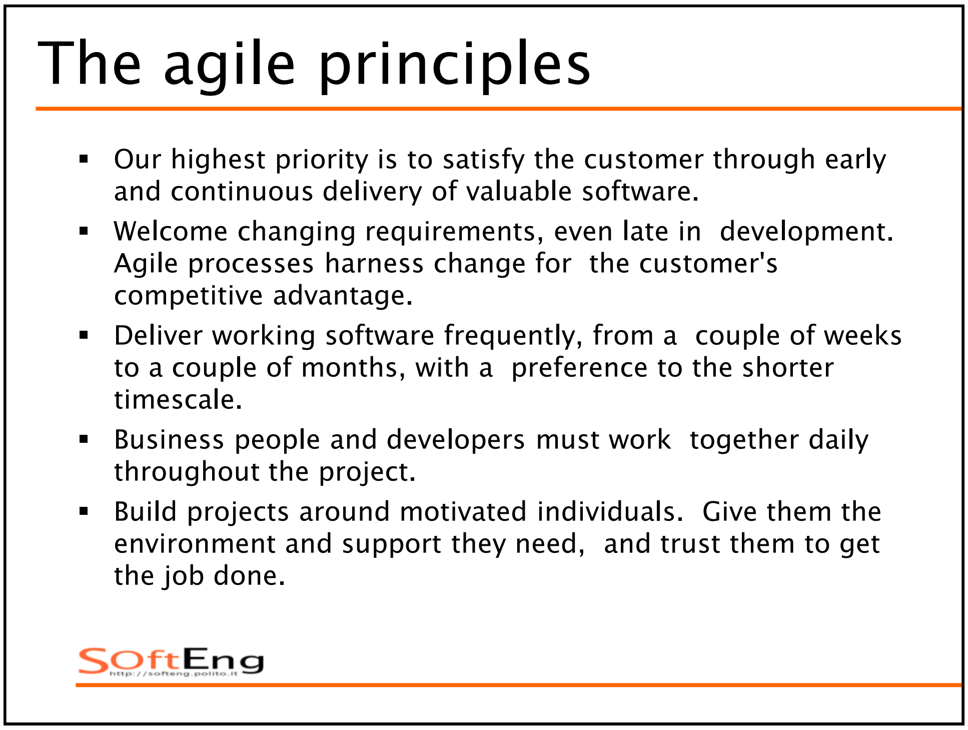
SOftEng
http://softeng.polito.it



Agilemanifesto.org

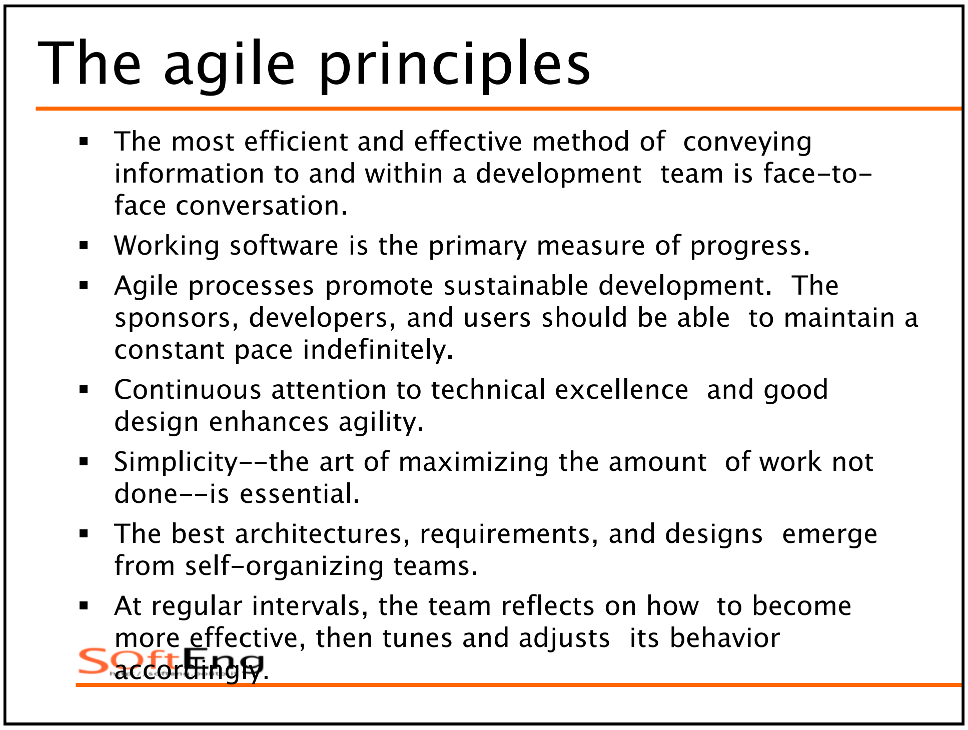
- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

SOftEng
http://softeng.polito.it



The agile principles

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.



The agile principles

- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile methods

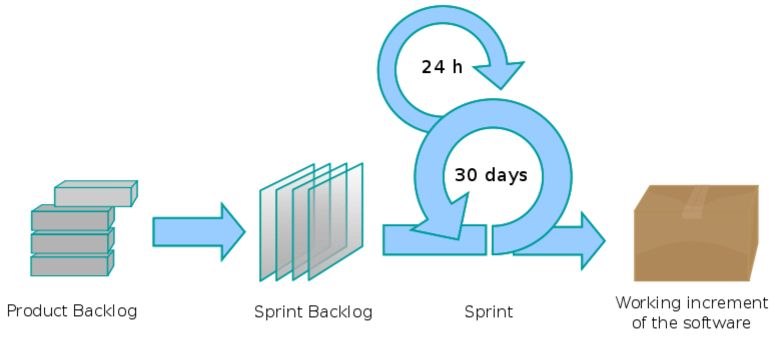
- XP
- Cristal
- Scrum

SOftEng http://softeng.polito.it

Scrum

SOftEng http://softeng.polito.it

Scrum process

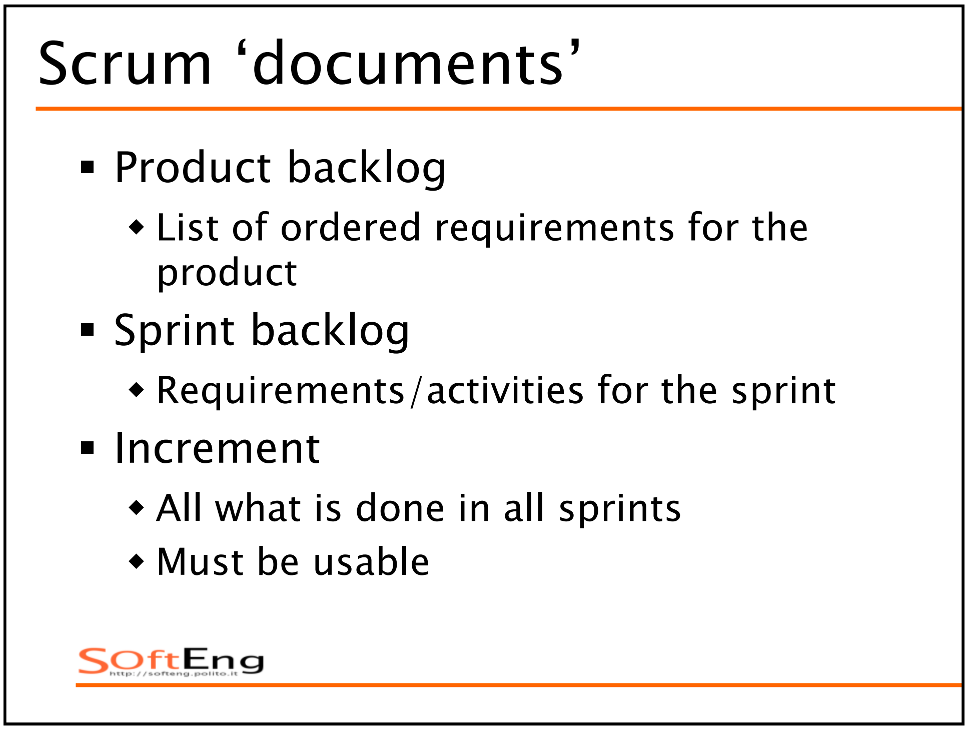


SoftEng http://softeng.polito.it

Scrum roles

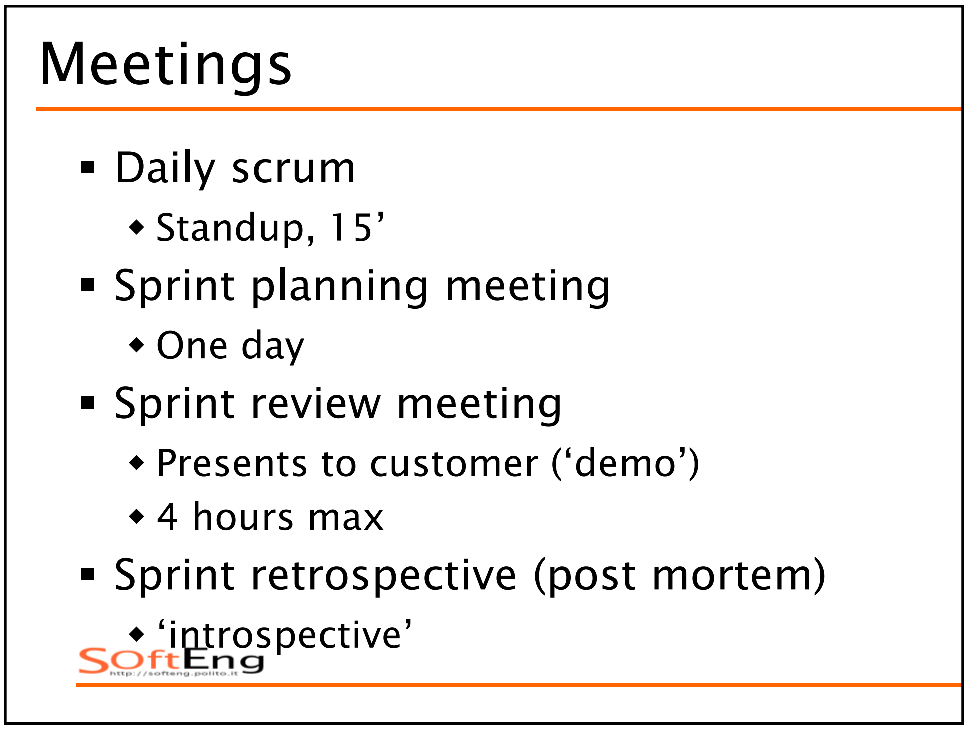
- Scrum master
  - (team leader)
- Product owner
  - Represents stakeholder and business
- Development team
  - All activities, self organizing

SoftEng http://softeng.polito.it



Scrum ‘documents’

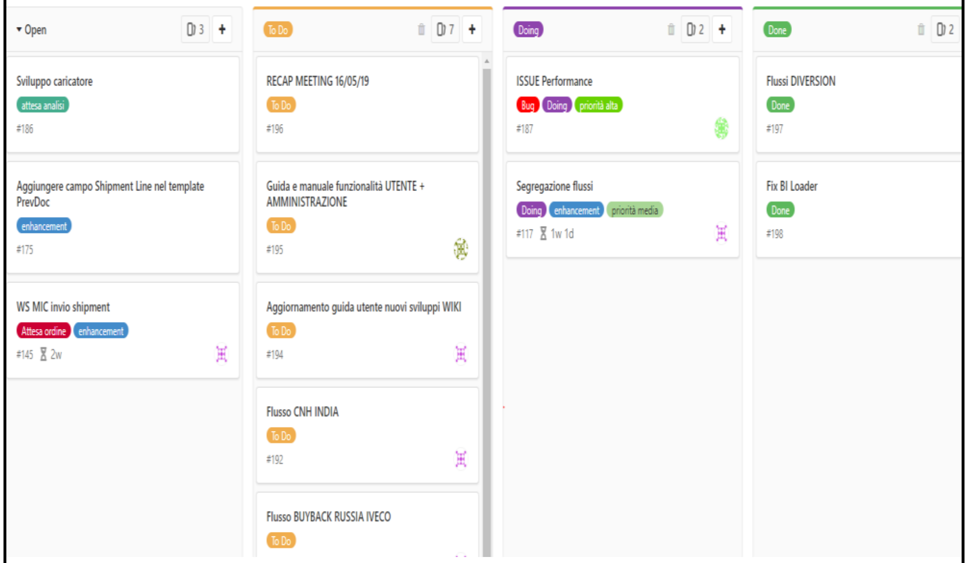
- Product backlog
  - List of ordered requirements for the product
- Sprint backlog
  - Requirements/activities for the sprint
- Increment
  - All what is done in all sprints
  - Must be usable



Meetings

- Daily scrum
  - Standup, 15’
- Sprint planning meeting
  - One day
- Sprint review meeting
  - Presents to customer (‘demo’)
  - 4 hours max
- Sprint retrospective (post mortem)
  - ‘introspective’

Scrum using GitLab issues


Open 3
Sviluppo caricatore
attesa analisi
#186
Aggiungere campo Shipment Line nel template PrevDoc
enhancement
#175
WS MIC invio shipment
Attesa ordine enhancement
#145 2w

To Do 7
RECAP MEETING 16/05/19
To Do
#196
Guida e manuale funzionalità UTENTE + AMMINISTRAZIONE
To Do
#195
Aggiornamento guida utente nuovi sviluppi WIKI
To Do
#194
Flusso CNH INDIA
To Do
#192
Flusso BUYBACK RUSSIA IVECO
To Do

Doing 2
ISSUE Performance
Bug Doing priorità alta
#187
Segregazione flussi
Doing enhancement priorità media
#117 1w 1d

Done 2
Flussi DIVERSION
Done
#197
Fix BI Loader
Done
#198

Issues labels

- Open issues
  - Backlog
    - One issue one user story, or else
- Todo, doing issues
  - Current Sprint
- Done issues
    - Delivered or ready to be delivered to end user

SoftEng http://softeng.polito.it

Agile vs waterfall, results

- Given the same starting point, results (delivered functionality) in case of agile or waterfall can be deeply different

SoftEng http://softeng.polito.it

- Starting point
  - Waterfall: F1 to F6, detailed definition
  - Agile: F1 to F6, high level definition

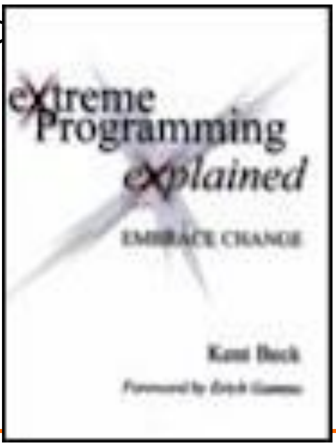
Waterfall	Agile
	Iteration 1 Rank: F3, F5, F1, F2, F6, F4 Delivered: F3, F5
	Iteration 2 Rank: F1, ~~F6~~, F6', F2, ~~F4~~, F7 Delivered F3, F5, F1, F6'
Delivered: F1, F2, F3, F4, F5, F6	Iteration 3 Rank F2 F7 Delivered F3, F5, F1, F6', F2, F7

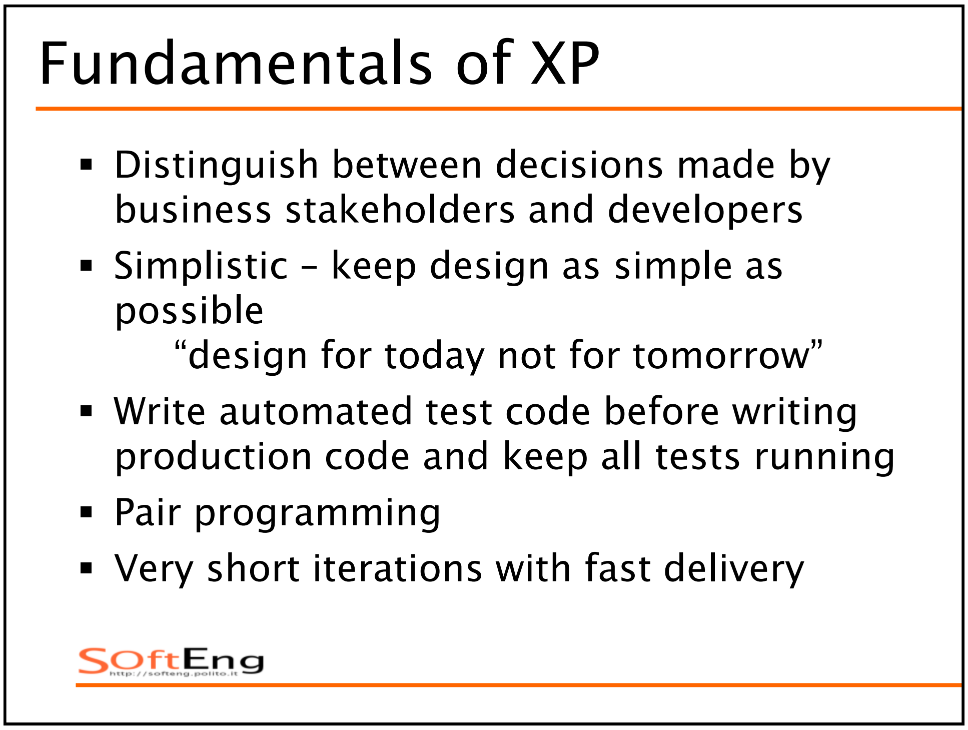
SoftEng http://softeng.polito.it

eXtreme Programming

Extreme programming

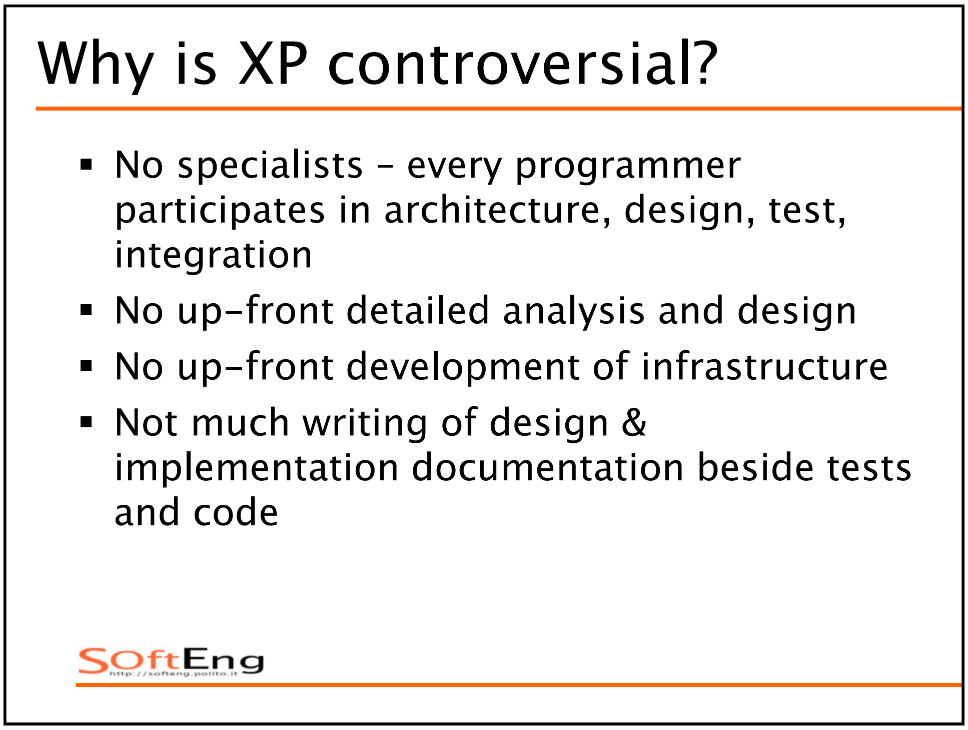
- Kent Beck: Extreme Programming Explained Addison-Wesley, 2000, ISBN 0-201-61641-6





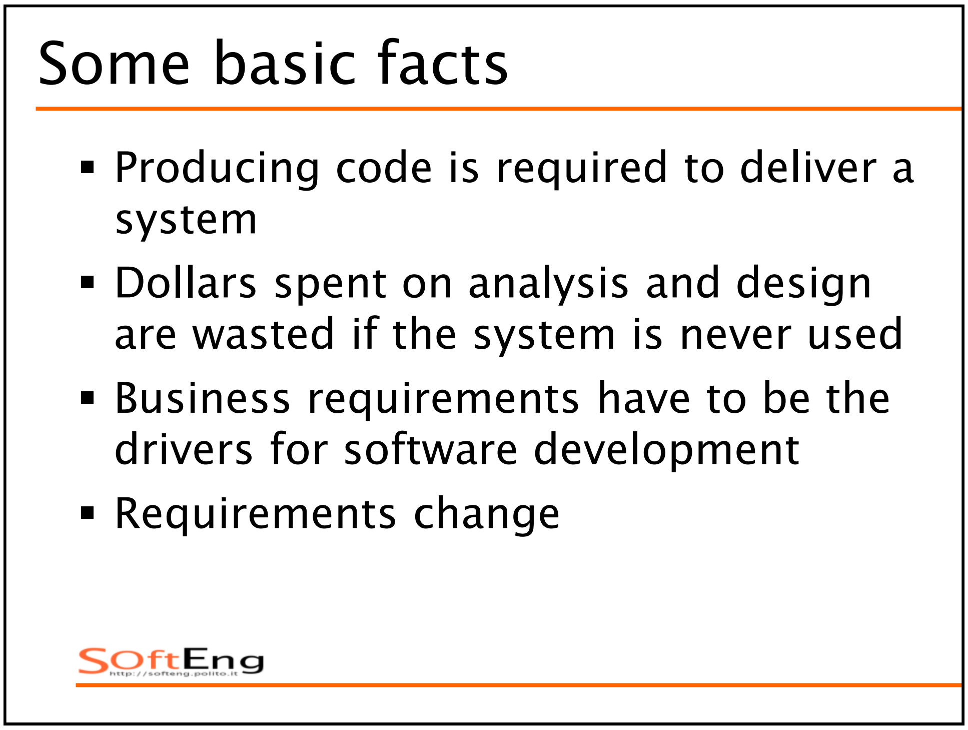
Fundamentals of XP

- 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



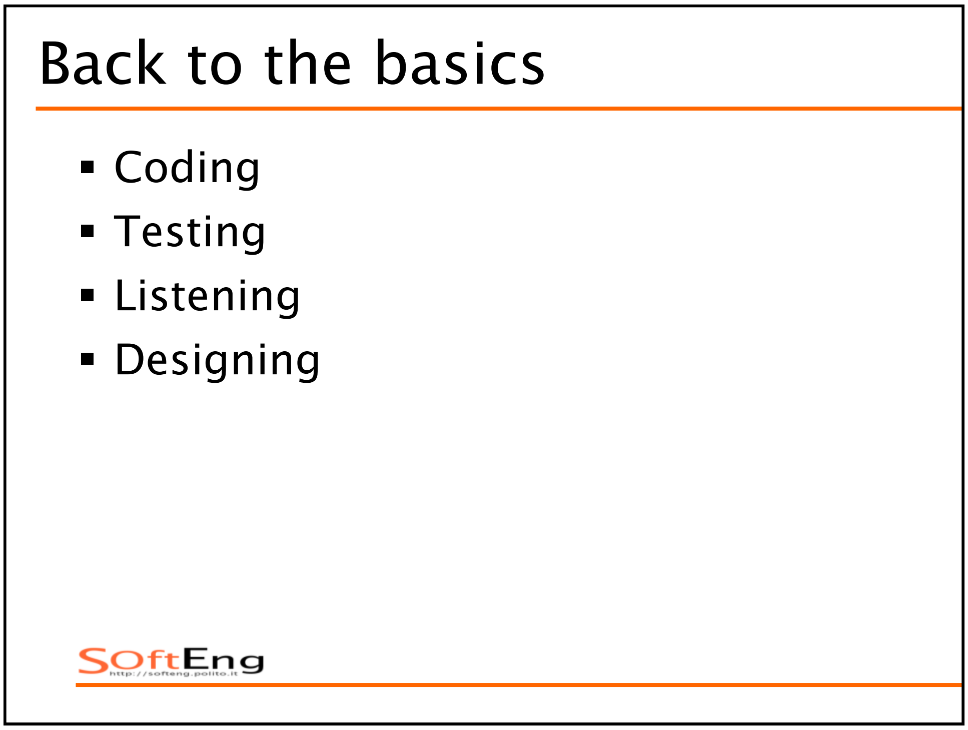
Why is XP controversial?

- 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



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



Back to the basics

- Coding
- Testing
- Listening
- Designing

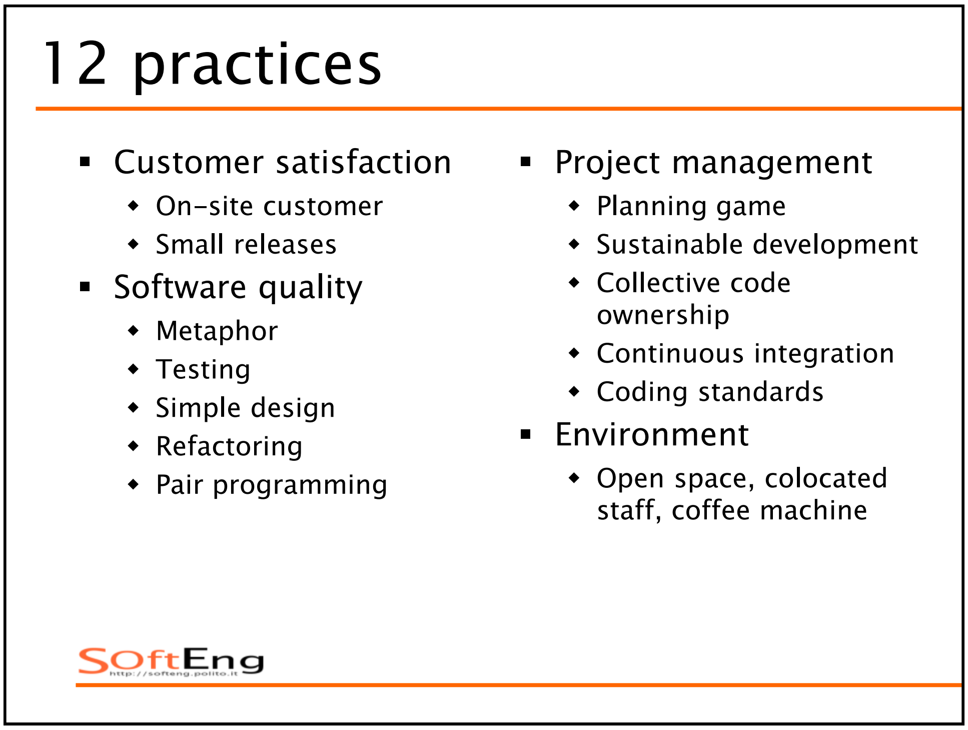
Four values

- 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 http://softeng.polito.it

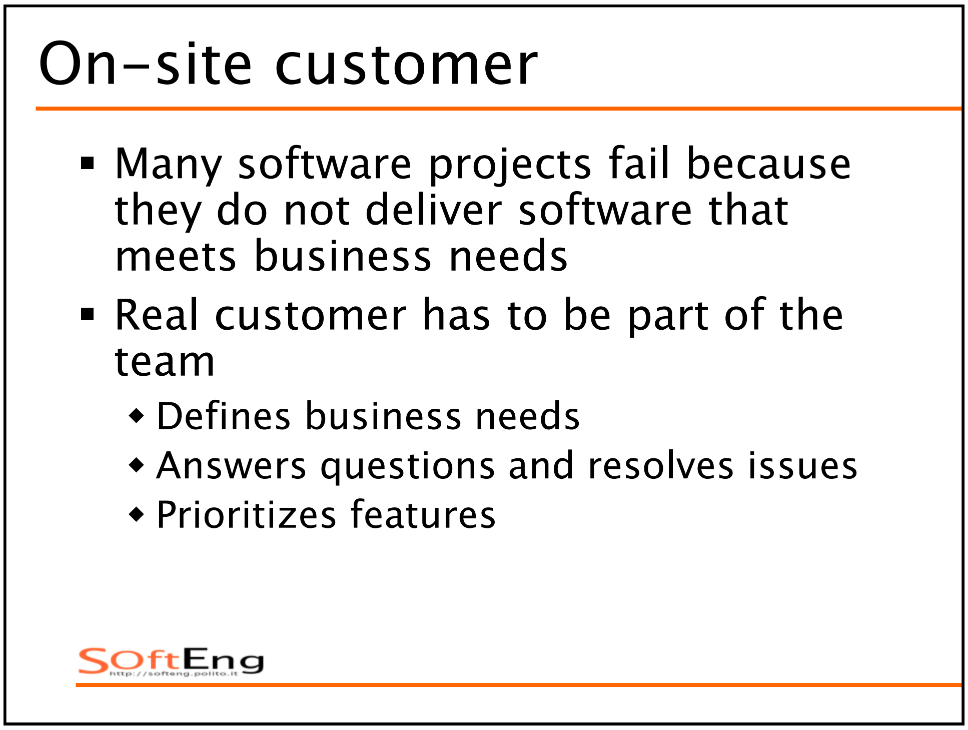
The key practices

SOftEng http://softeng.polito.it



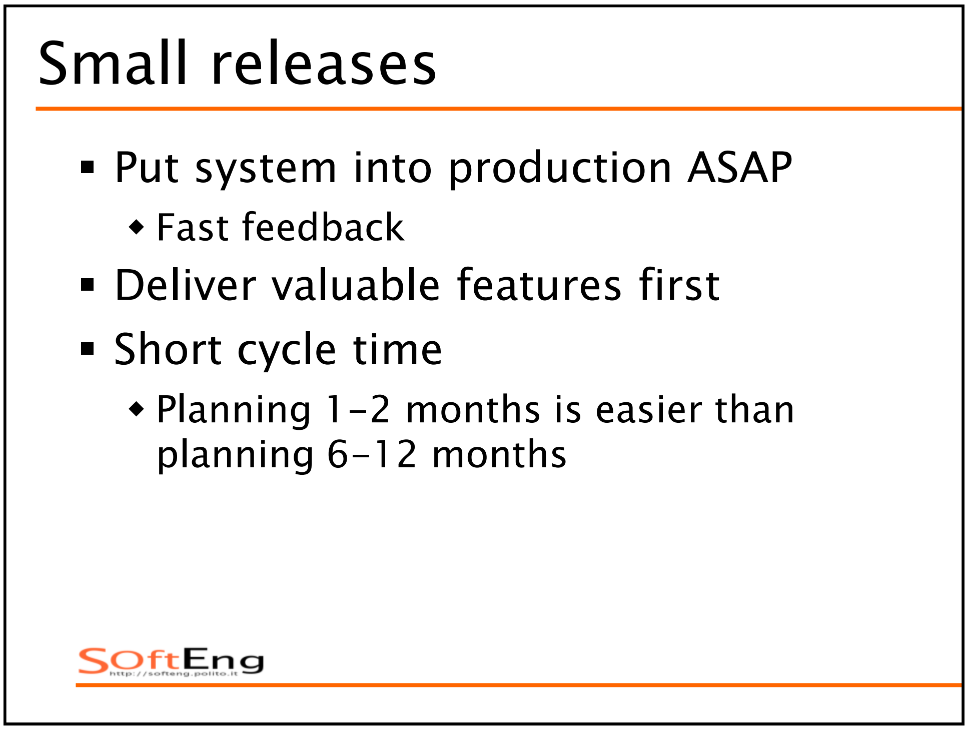
12 practices

- 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
- Environment
  - Open space, colocated staff, coffee machine



On-site customer

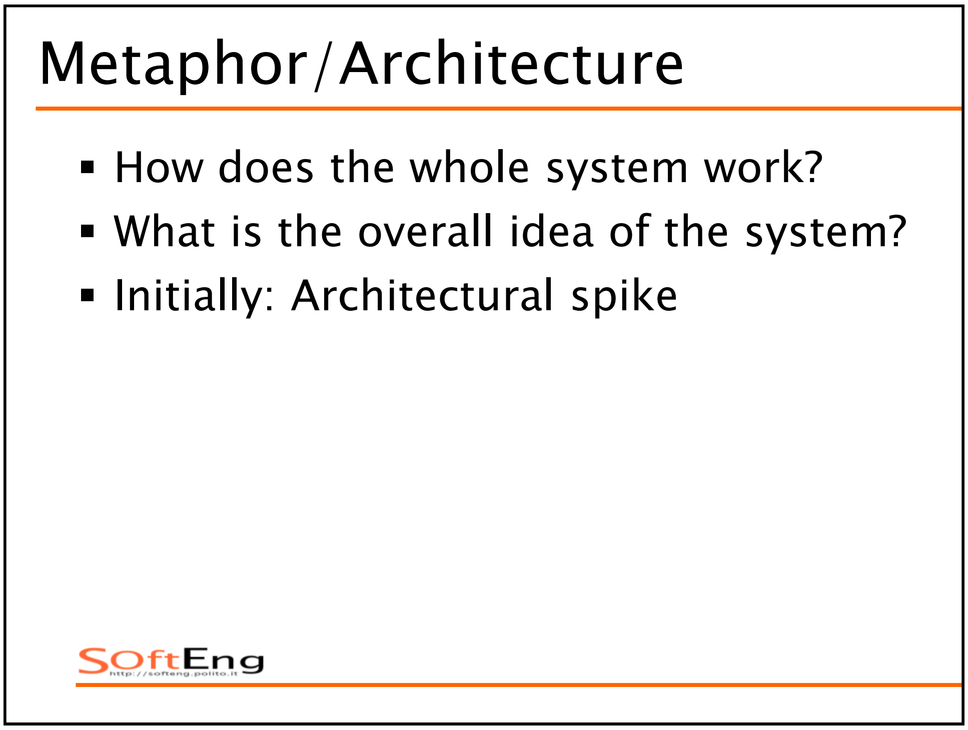
- Many software projects fail because they do not deliver software that meets business needs
- Real customer has to be part of the team
  - Defines business needs
  - Answers questions and resolves issues
  - Prioritizes features



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 http://softeng.polito.it



Metaphor/Architecture

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

SoftEng http://softeng.polito.it

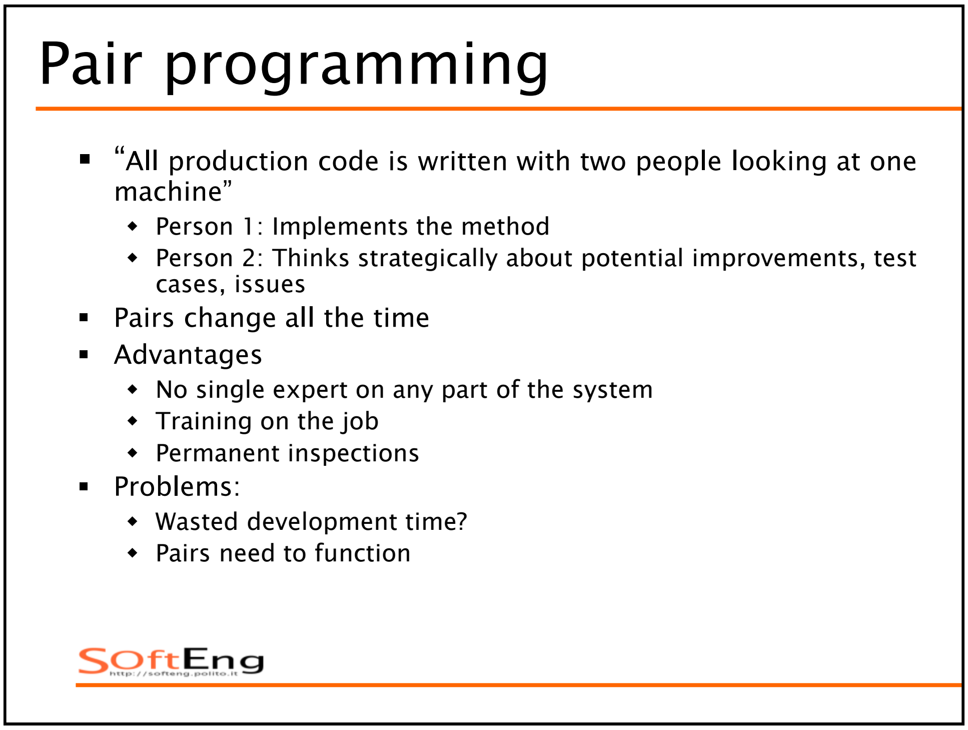
Simple design

- 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

Refactoring

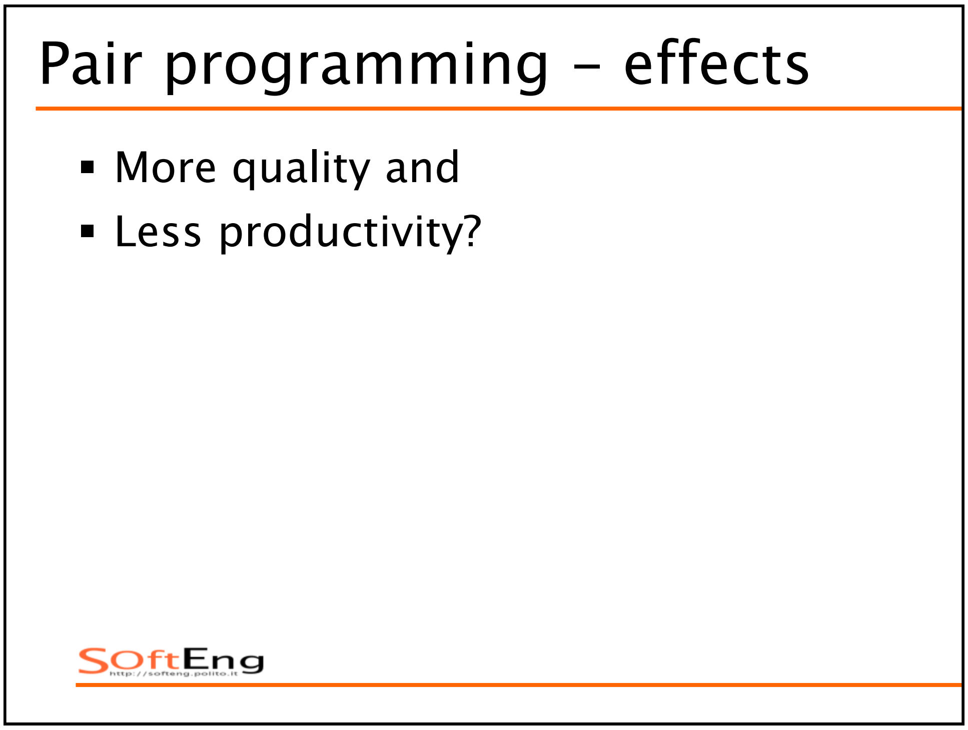
- Restructure system without changing the functionality
- Goal: Keep design simple
  - Change bad design when you find it
  - Remove dead code



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

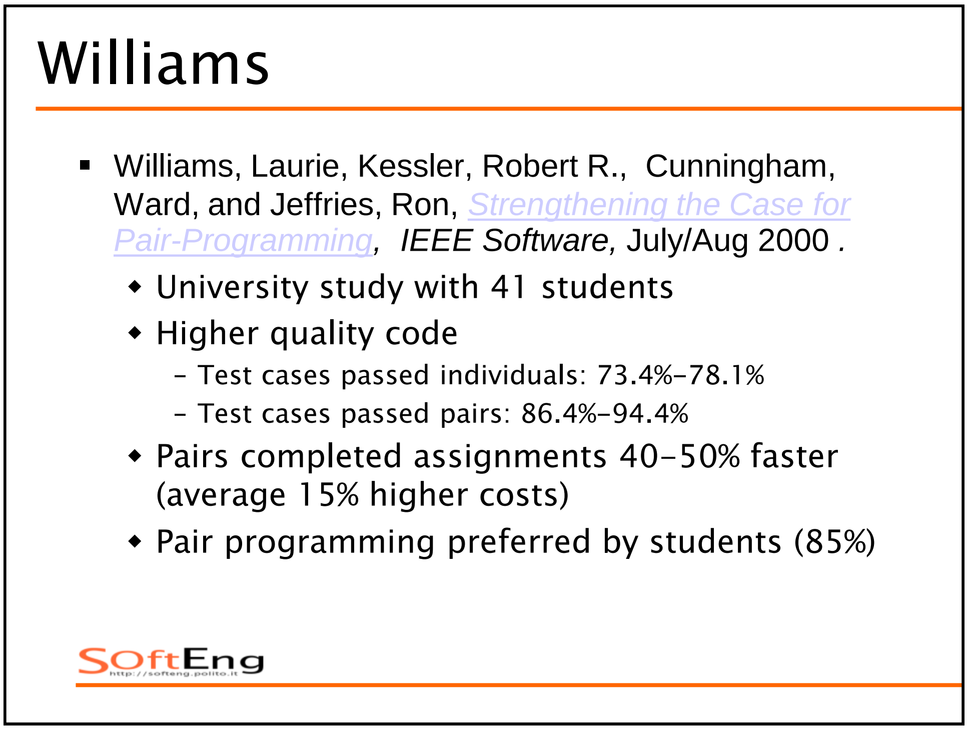
SoftEng http://softeng.polito.it



Pair programming – effects

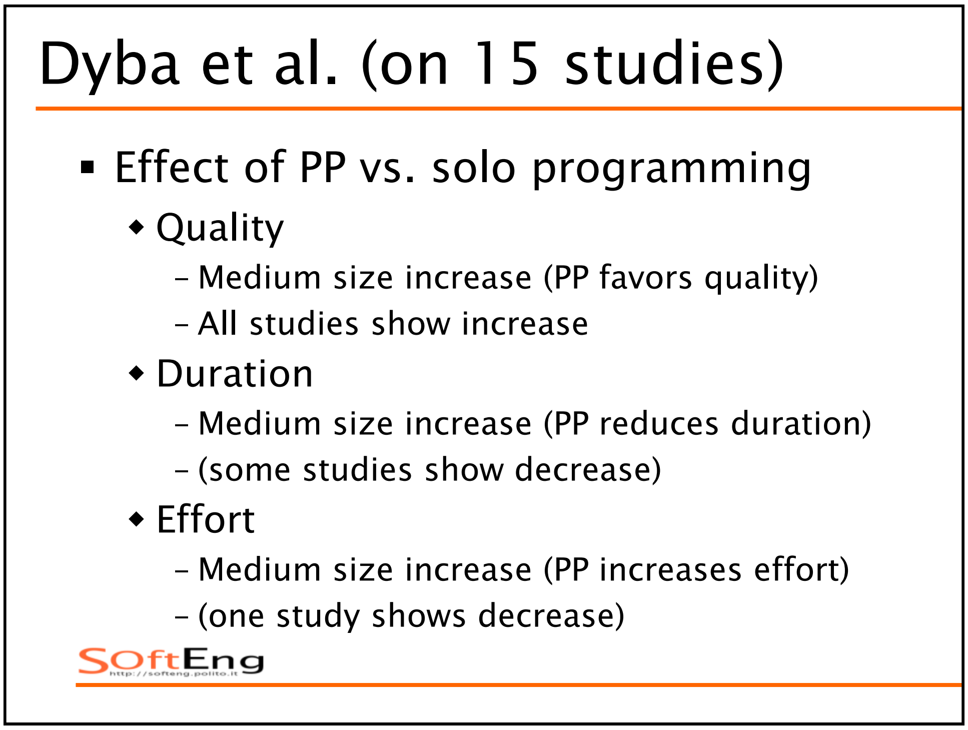
- More quality and
- Less productivity?

SoftEng http://softeng.polito.it



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%)

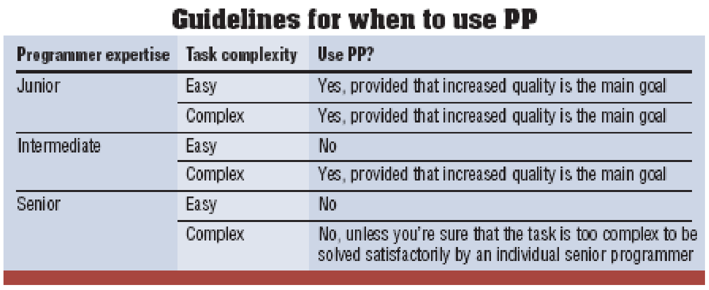


Dyba et al. (on 15 studies)

- Effect of PP vs. solo programming
  - Quality
    - Medium size increase (PP favors quality)
    - All studies show increase
  - Duration
    - Medium size increase (PP reduces duration)
    - (some studies show decrease)
  - Effort
    - Medium size increase (PP increases effort)
    - (one study shows decrease)

Dyba et al.

- Effect of group dynamics and skill of pairs must be considered. One one study does it and suggests:



Guidelines for when to use PP

Programmer expertise	Task complexity	Use PP?
Junior	Easy	Yes, provided that increased quality is the main goal
	Complex	Yes, provided that increased quality is the main goal
Intermediate	Easy	No
	Complex	Yes, provided that increased quality is the main goal
Senior	Easy	No
	Complex	No, unless you're sure that the task is too complex to be solved satisfactorily by an individual senior programmer

SOftEng http://softeng.polito.it

Test Driven Development

- *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 http://softeng.polito.it

The planning game

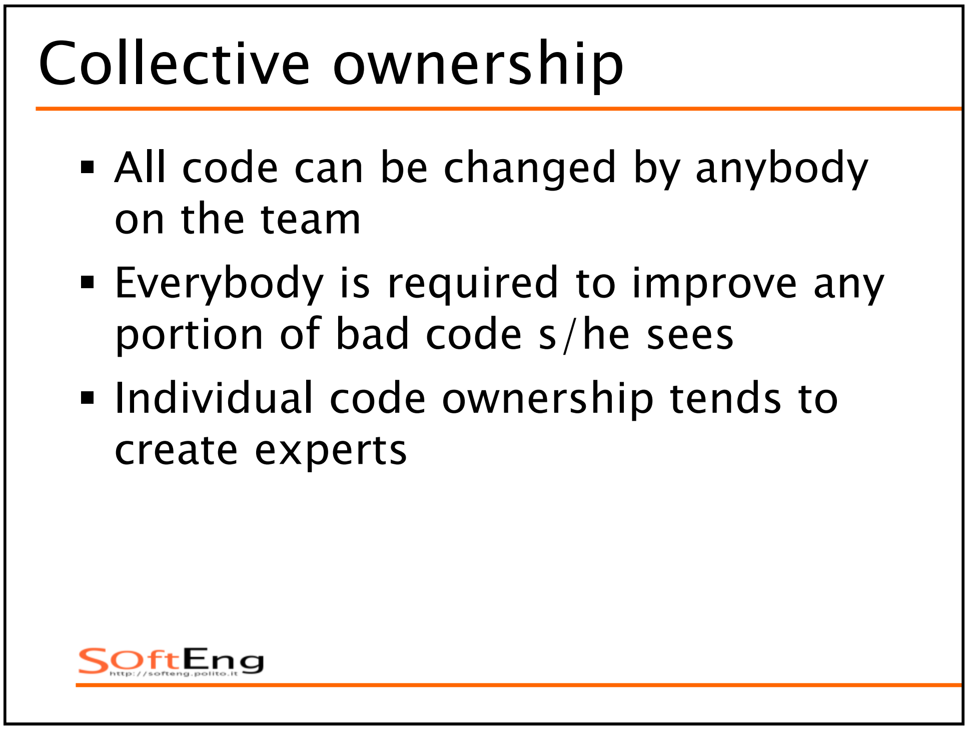
- Business decisions
  - Scope: which “stories” should be developed
  - Priority of stories
  - Composition of releases
  - Release dates
- Technical decisions
  - Time estimates for features/stories
  - Elaborate consequences of business decisions
  - Team organization and process
  - Scheduling

SoftEng http://softeng.polito.it

Sustainable development

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

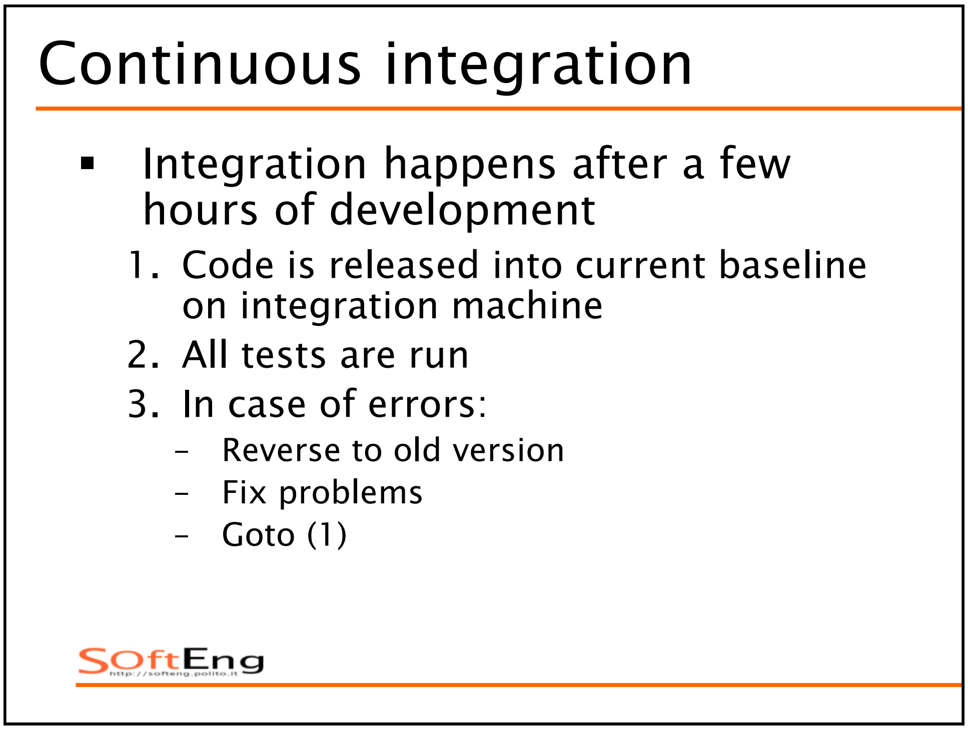
SoftEng http://softeng.polito.it



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 http://softeng.polito.it



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 http://softeng.polito.it

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 http://softeng.polito.it

Environment

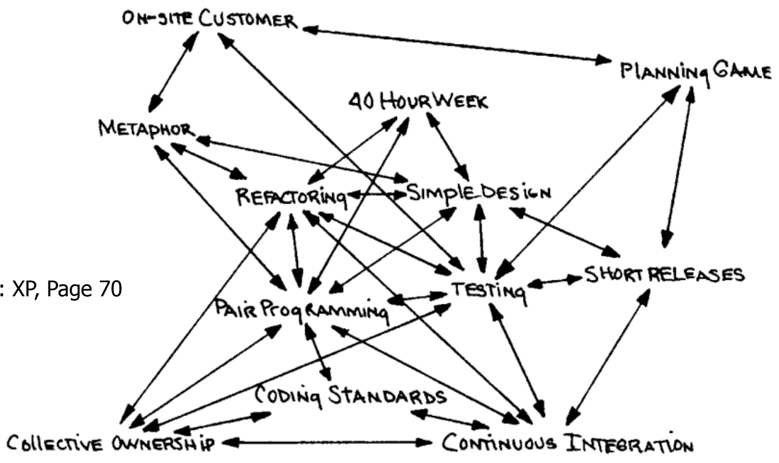
- Support communication among developers
  - Open space (no closed offices)
  - Common space (coffee machine, food, blackboard)

SoftEng http://softeng.polito.it

Notes on working environment

- Evidence that the working environment has great influence on quality and quantity of work
  - Negative factors:
    - Interruptions (colleagues, email, meetings)
    - Lack of confort
      - Noise
      - Light

How everything fits together



From Beck: XP, Page 70

Issues in XP adoption

All techniques?

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

Business contracts

- Time and material model
- Fixed price model
  - Not suitable

SOftEng http://softeng.polito.it

Colocation and project size

- Co-location of team members required
- Scalability of the process:
  Small teams → small projects
  (2-10 developers ideal size)

SOftEng http://softeng.polito.it