



Beware of Agility

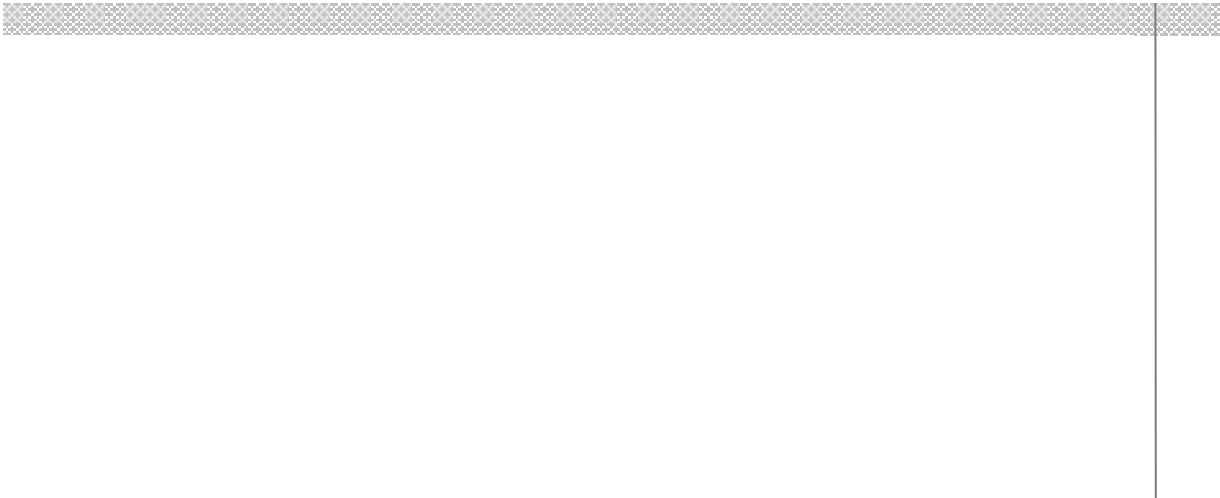
dangers of blind adoption

Sven.Gorts@refactoring.be, Olivier.Costa@AEGISoft.be

Agenda



- Values & Principles
- Agility meets Reality
- Afterthoughts



Values & Principles

Agile Manifesto



Manifesto



Extreme Programming



Values **XP** Principles
Practices

XP Values



simplicity

communication

Values

respect

courage

feedback

XP Principles



rapid feedback

embracing change

Principles

assume simplicity

incremental change

quality work

Choosing For Agility



Why?

- Developer's expectations

We want to be agile because ...

- Benefits promised

We expect agility to improve ...



Agility meets **Reality**

Contradicting Values



XP

- communication
- simplicity
- courage
- feedback
- respect

Company

- documentation
- opportunity
- politics
- ignorance
- heroism

Agile Evangelist



Motivated:

- Brings in a fresh view
- Identifies teams problems

Unfortunately:

- Role is not to change, but to type code

⇒ Promotes agility

⇒ Advocates change

Legacy Guru



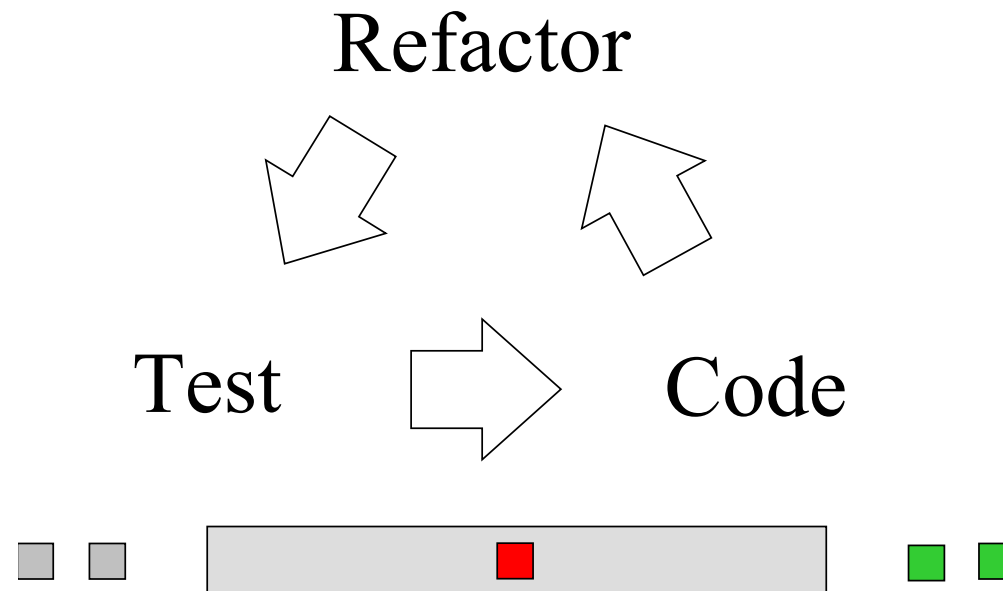
- Knows his way around
- Clever and quick hacker
- Has lots of credit

⇒ Gets the job done



Theory and Practice

Test Driven Development



- Red: Write A Failing Unit Test
- Green: Write Enough Code To Make It Pass
- Refactor: Deal With Design / Code Smells

Test Driven Development



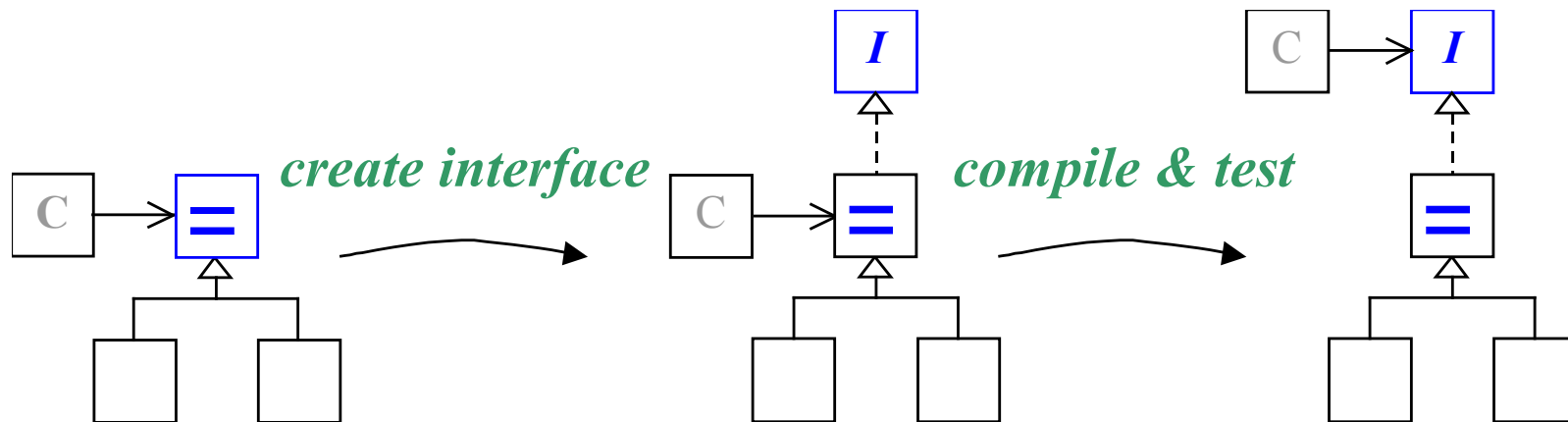
However:

- Unit tests take time to write
- Unit tests need to be maintained
- Unit tests don't test the system
- Writing good unit tests is a skill

⇒ Bottleneck for development !

⇒ We have dedicated testers !

Refactoring



- Improvement of code in small steps
- Safe changes that preserve behavior

Refactoring



However:

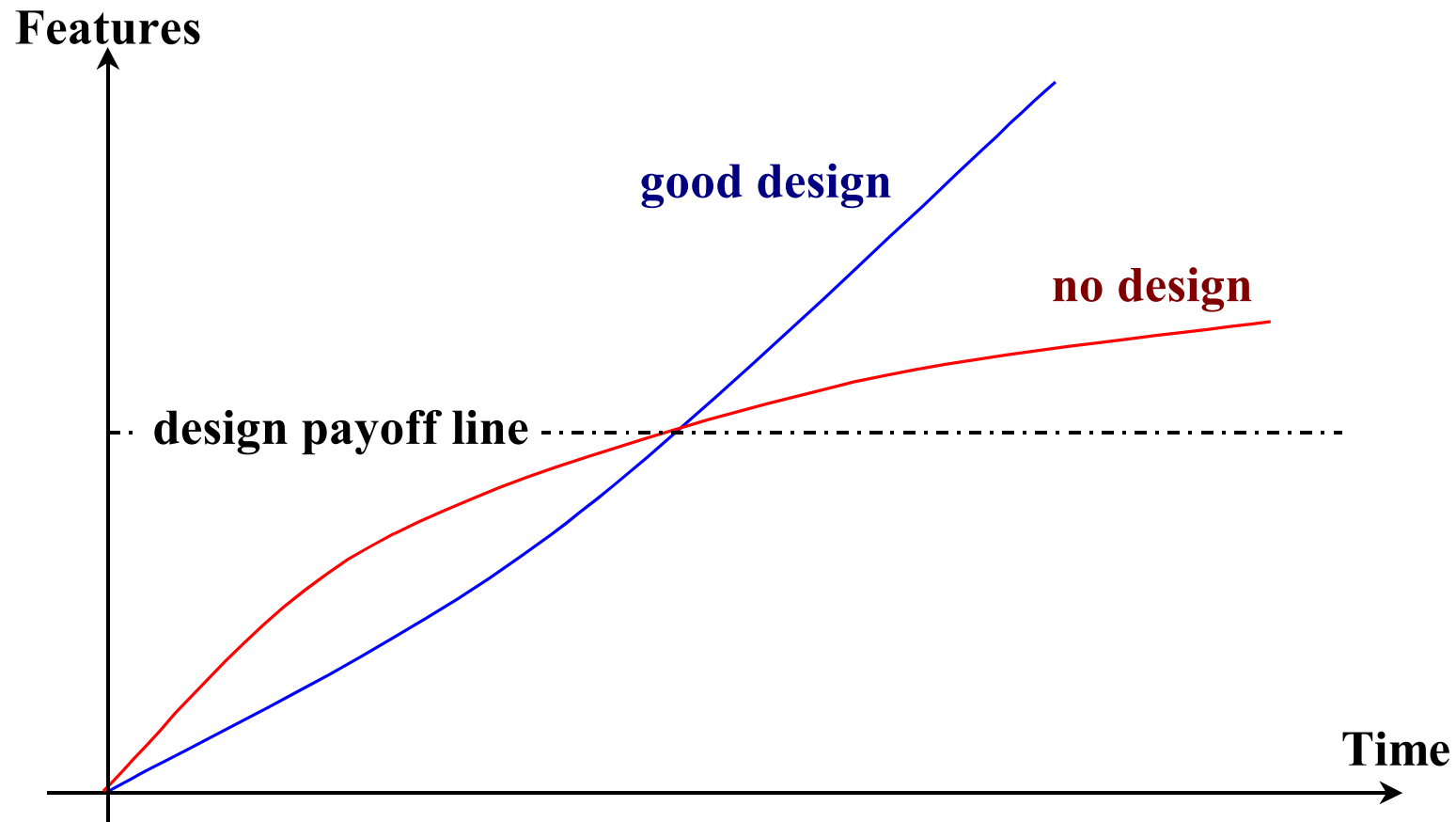
- Refactoring stalls development
- Design changes not understood
- Regressions happen, often
- Efficient unit tests required

Why fix what ain't broken ?

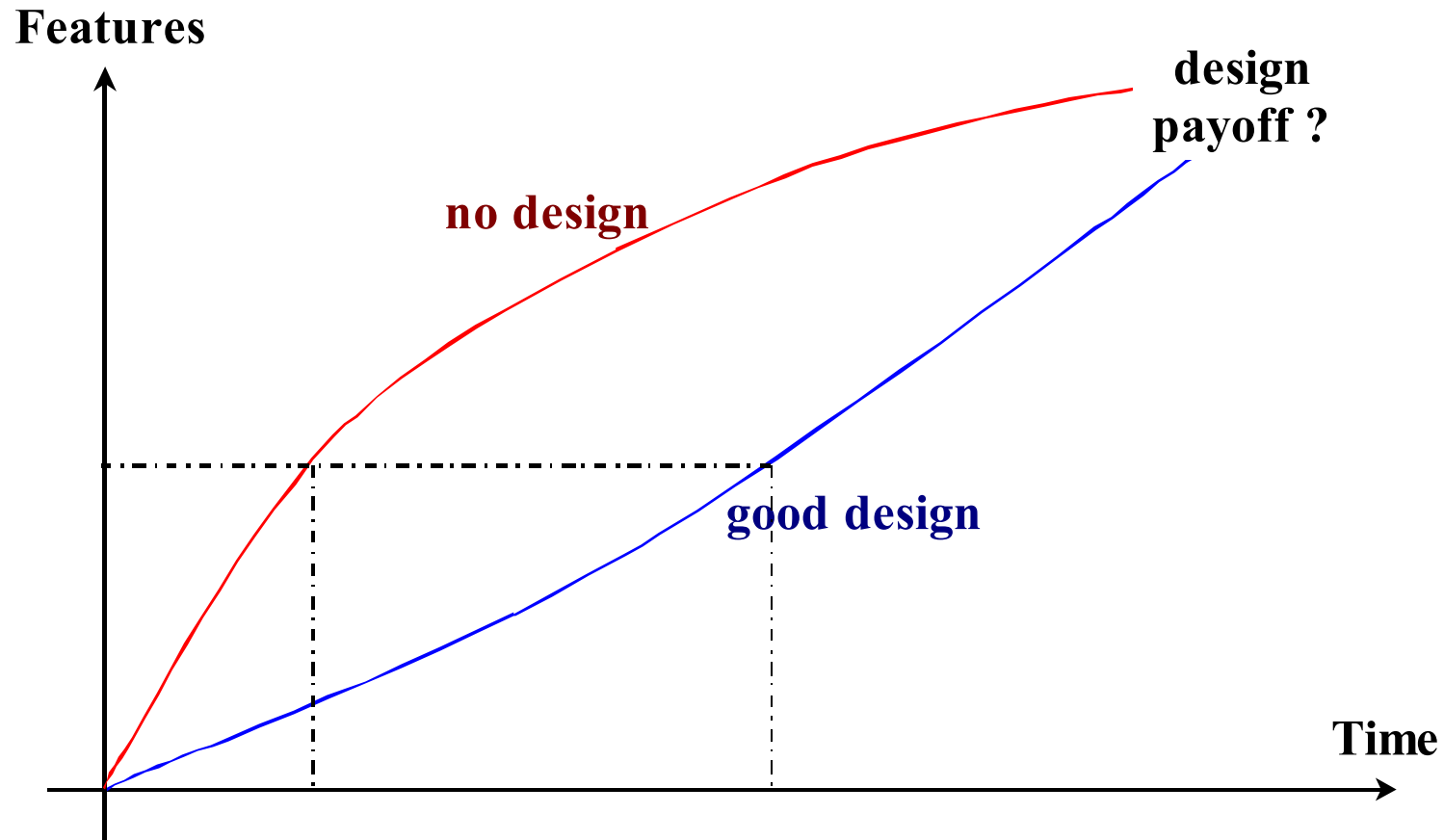
⇒ Does code have to be clean ?

⇒ We can live with a little dirt around

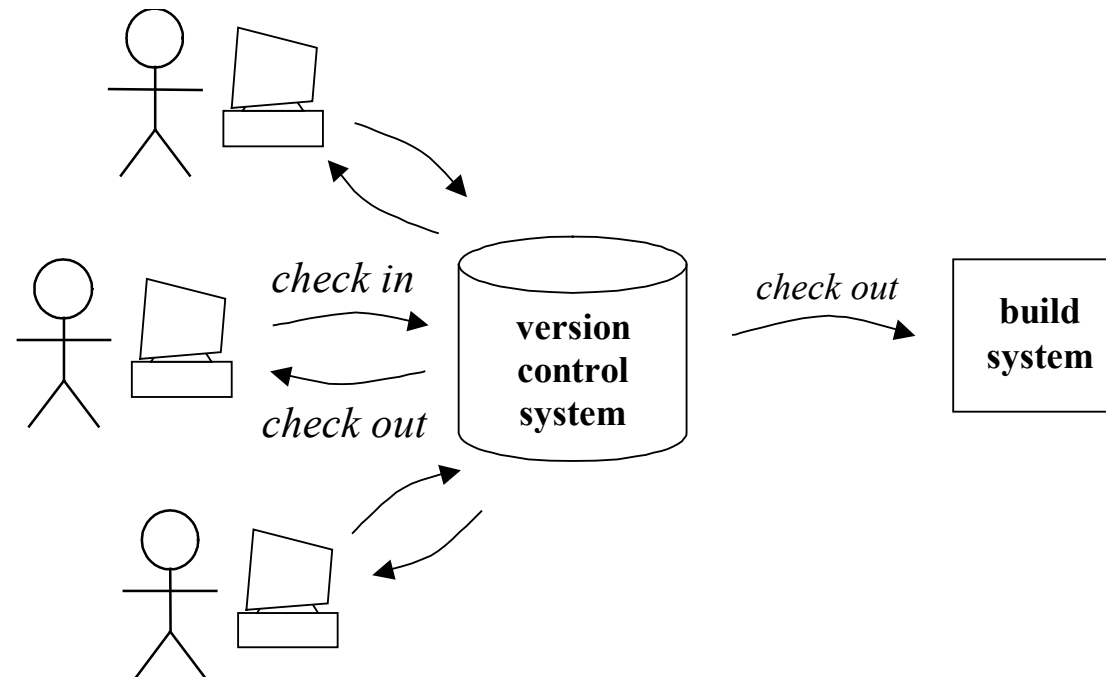
The Value Of Design



The Cost Of Design



Continuous Integration



- Developers integrate frequently
- Build system compiles & runs tests

Continuous Integration



However:

- Broken code gets checked in
- Slow tests are left to the build system
- Unclear who 's to fix the build
- Build broken for several weeks

⇒ Automated build system

⇒ Green build ? (month / week)

⇒ Red build ? (week / month)

Collective Ownership



- Every team member
 - is free to improve
 - any part of the code
 - Code owned by the team
 - accepted code standard
 - shared vocabulary
 - Requires automated tests
 - Consider Pair programming as well
- ⇒ Code has a uniform style and looks as if written by a competent individual

Collective Ownership



However:

- Lack of discipline

- Quick hacks, Broken unit tests, Design degradation, No cleanup done, Broken Windows

- No watch dog

- Converting “Hero & minions” team ?

⇒ No Code Ownership



Afterthoughts

Blind Adoption of Practices

Team practices:

- Test Driven Development
- Refactoring
- Continuous Integration
- Collective Ownership

⇒ Where are the results ?

Agile Evangelisation



Converting the team:

- Bad for team

- Developers feel threatened
- Divides team

- Bad for evangelist

- Tired of fighting the uphill battle
- Code still looks ugly

⇒ What did the evangelist achieve ?

Other Mistakes We Made



- Timing
- Experimenting

Your New Job



- Here are some documents
 - Hey, it works doesn't it ?
 - Jim can do that in only 1 hour
 - We don't work like that here
 - If it ain't broke, don't fix it
- ⇒ No chance agility will rescue you

How To Survive ?



- Agile fundamentalism ?
- Look for another job ?
- Accepting the status quo ?

Questioning Agility



- Is cost of change acceptable ?
- Can ongoing project be converted ?
- Is current team ready for agility ?

Note To Programmers



Even programmers can be whole people in the real world.

XP is an opportunity to test yourself, to be yourself,
to realize that maybe you've been fine all along
and just hanging with the wrong crowd.

(Kent Beck, Extreme Programming Explained 2nd Edition)

References



[<www.refactoring.be>](http://www.refactoring.be)

- Extreme Programming Explained 2nd Edition
(Kent Beck with Cynthia Andres)
- Agile Software Development
(Alan S. Koch)
- Extreme Programming – pocket guide
(Cromatic – [O'Reilly](#))

Session Retrospective

