CS3300 Introduction to Software Engineering

Lecture 19: Agile Development Methods

Nimisha Roy ▸ nroy9@gatech.edu

Slides adapted from Alessandro Orso

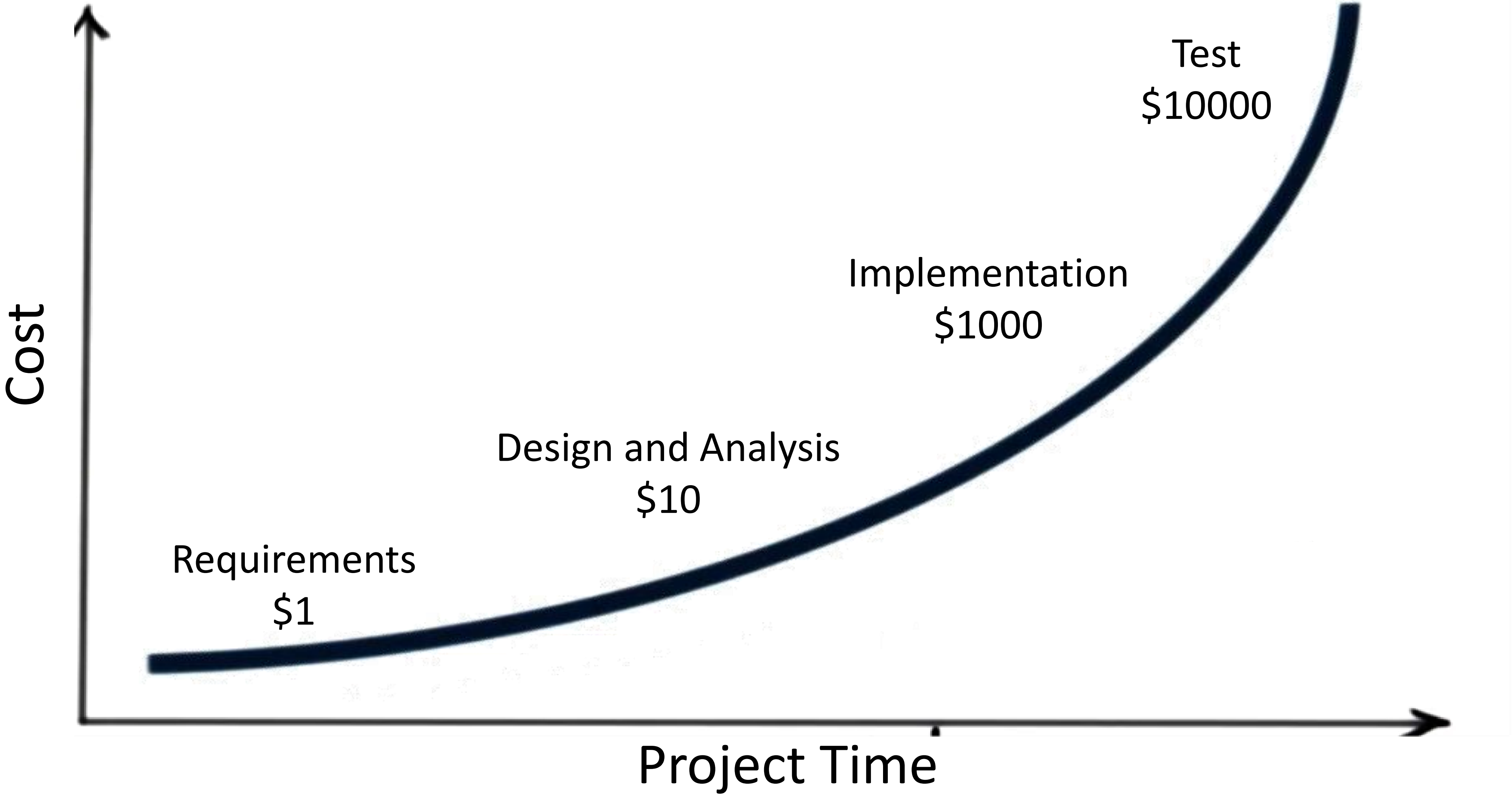
Transition from Waterfall to Agile

From Waterfall….

… To Agile

Slide adapted from Alessandro Orso

As Barry Boehm Said…

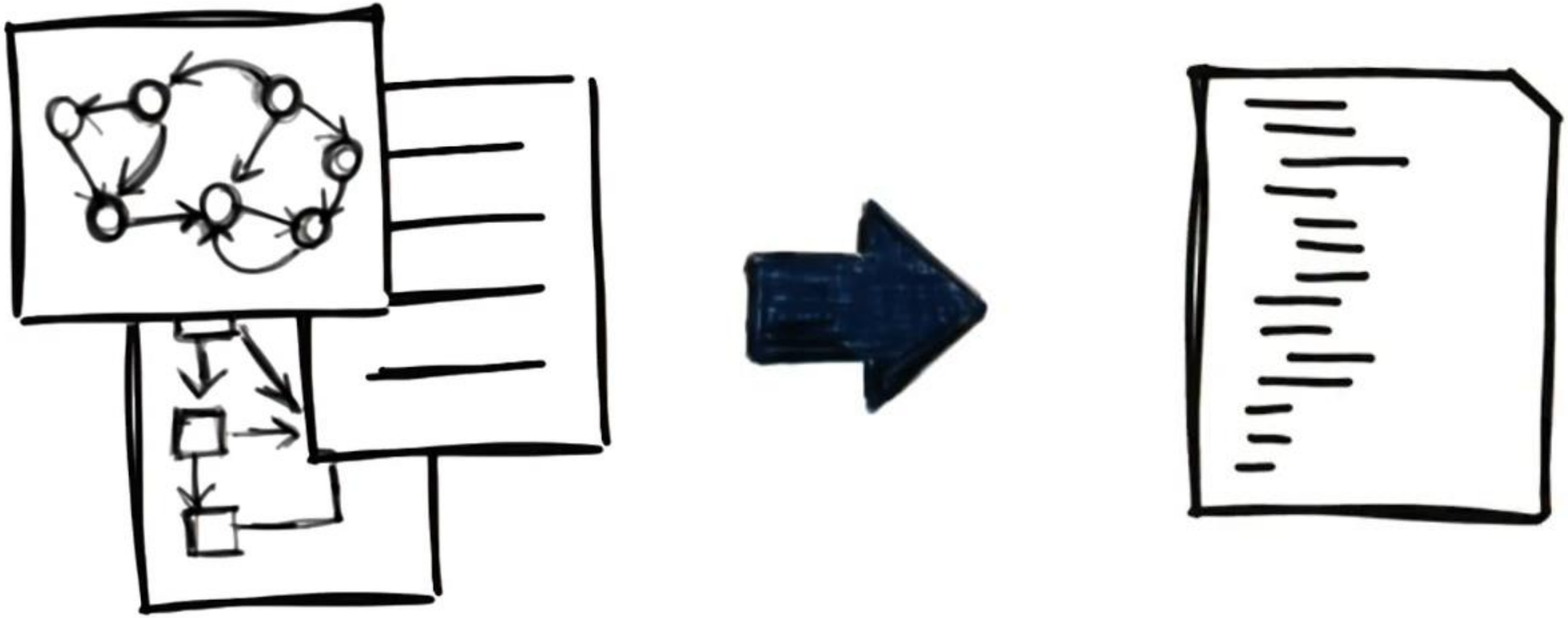


Cost of Change grows exponentially with time

Slide adapted from Alessandro Orso

What to do then?

Discover errors early => upfront planning

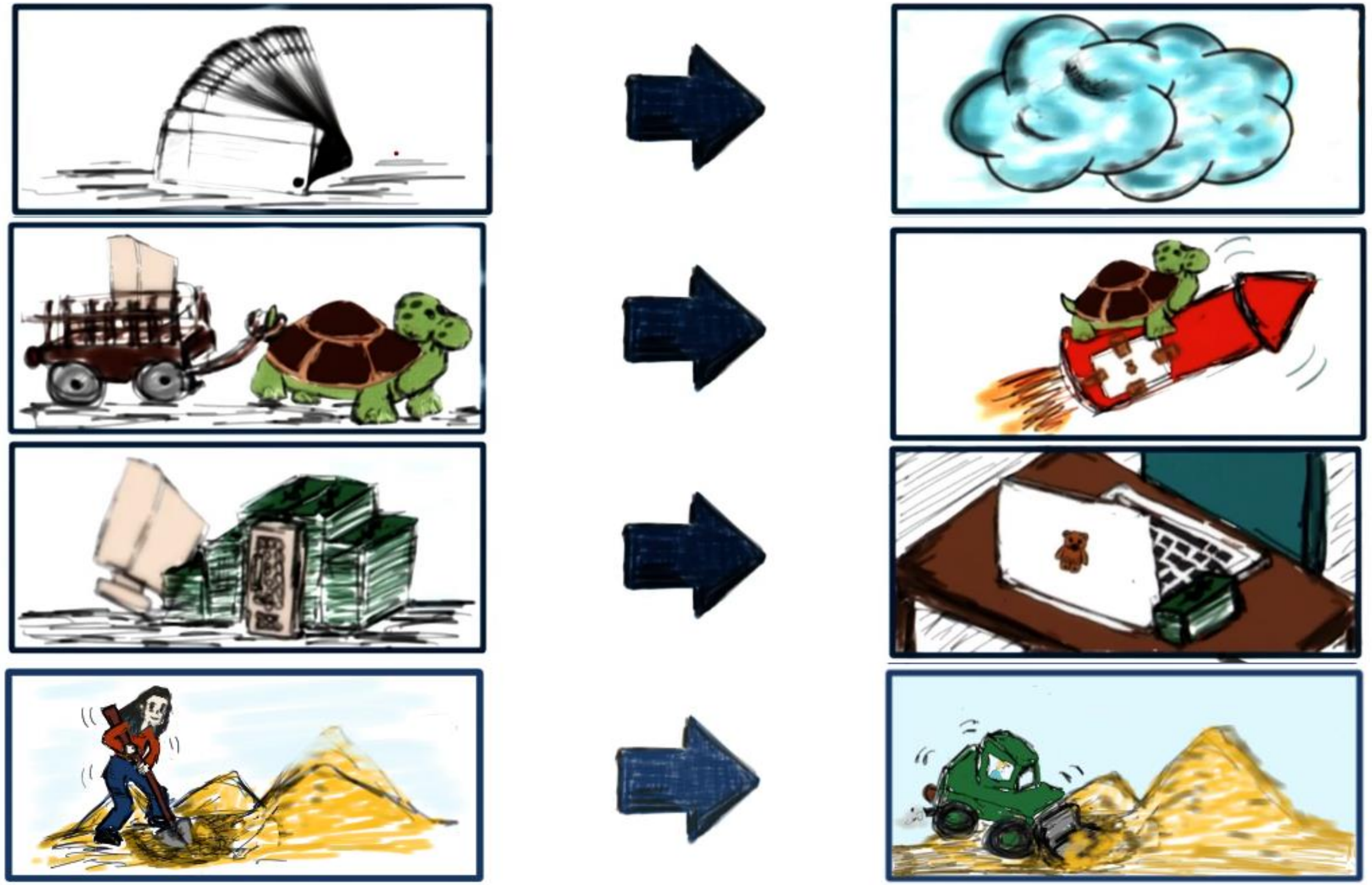


Model Documents

Code

Slide adapted from Alessandro Orso

Something Changed in the last 30 years



Speed, cost, efficiency, automation – high level languages, VCS, smart ideas

Slide adapted from Alessandro Orso

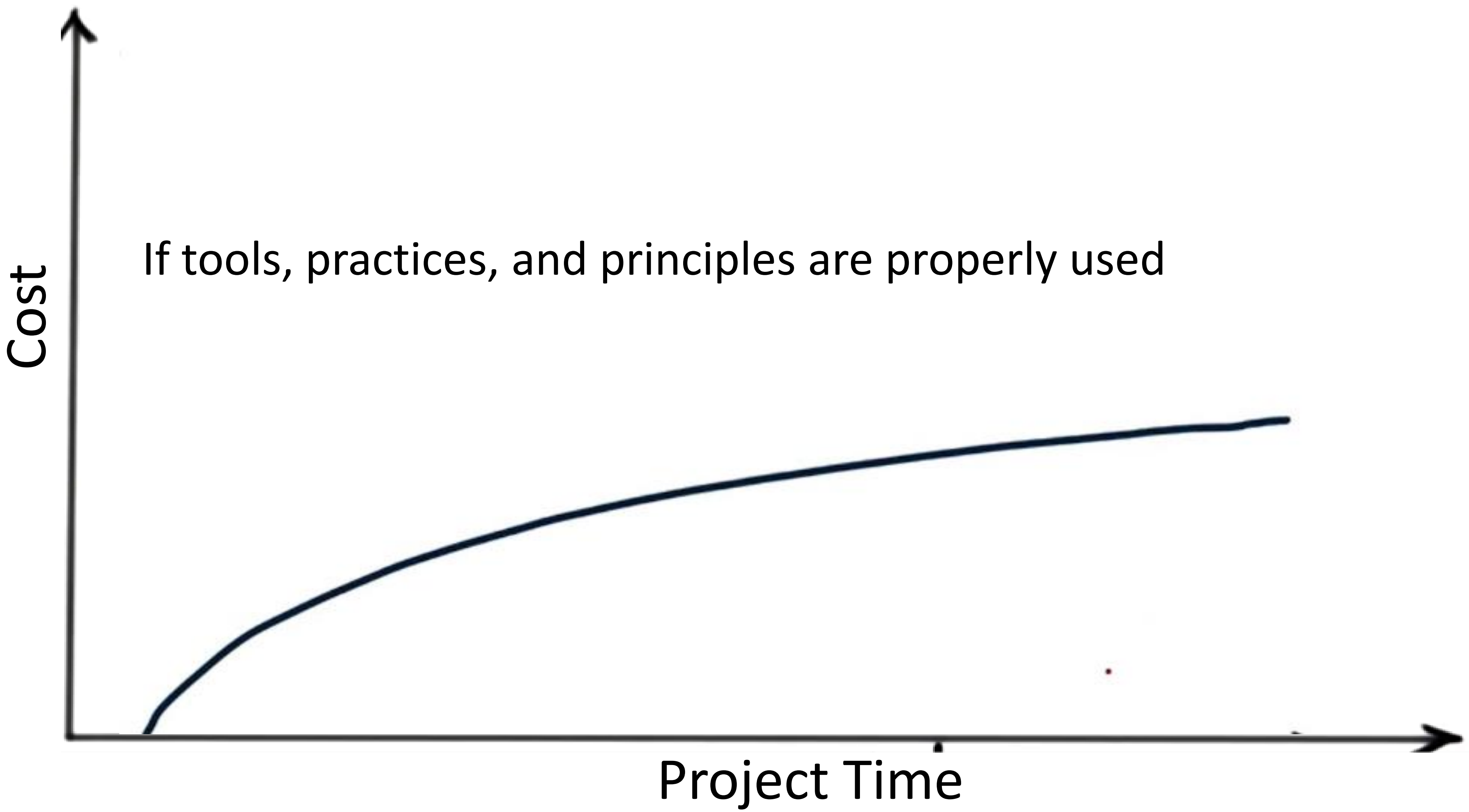
Something Changed in the last 30 years

Easy to change!

Speed, cost, efficiency, automation – high level languages, VCS, smart ideas

Slide adapted from Alessandro Orso

Maybe the cost of change can be flat?



If tools, practices, and principles are properly used

Slide adapted from Alessandro Orso

If cost is flat...

Upfront work == Liability

We pay for speculative work some of which is likely to be wrong.

Ambiguity, Volatility => Good to delay

We don't want to plan and invest resources for something that might never happen

There is value in waiting !!

Time answers questions and removes uncertainty

Slide adapted from Alessandro Orso

Agile Methods Aim at Flat Cost

Feb 2001: 17 Software Developers met to discuss lightweight development methods and published….

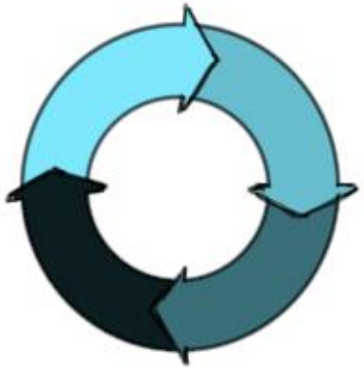
Some companies that use this SW development process: IBM, Cisco, Microsoft, AT&T

Slide adapted from Alessandro Orso

Agile Methods: Principles

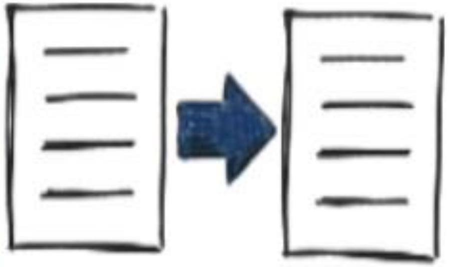
Focus On the Code rather than design

People over Process



Iterative Approach- deliver working SW quickly, evolve it quickly

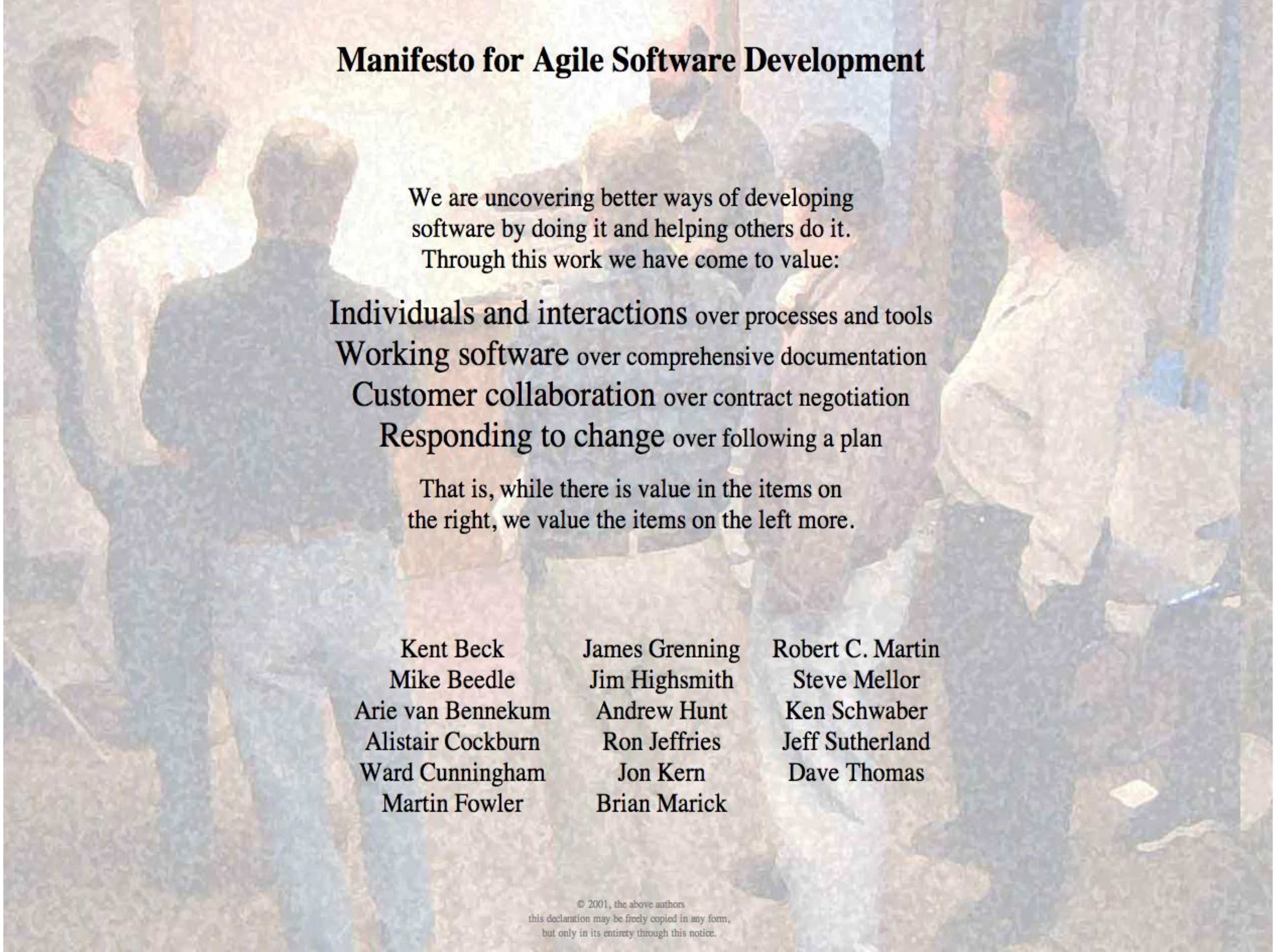
Customer Involvement



Expectation that requirements will change

Simplicity – not inadequacy

Slide adapted from Alessandro Orso



Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

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.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

XP

“ XP is a lightweight methodology for small to medium sized teams developing software in the face of vague or rapidly changing requirements “

Kent Beck

Slide adapted from Alessandro Orso

What is XP?

Lightweight- doesn't overburden developers with an invasive process

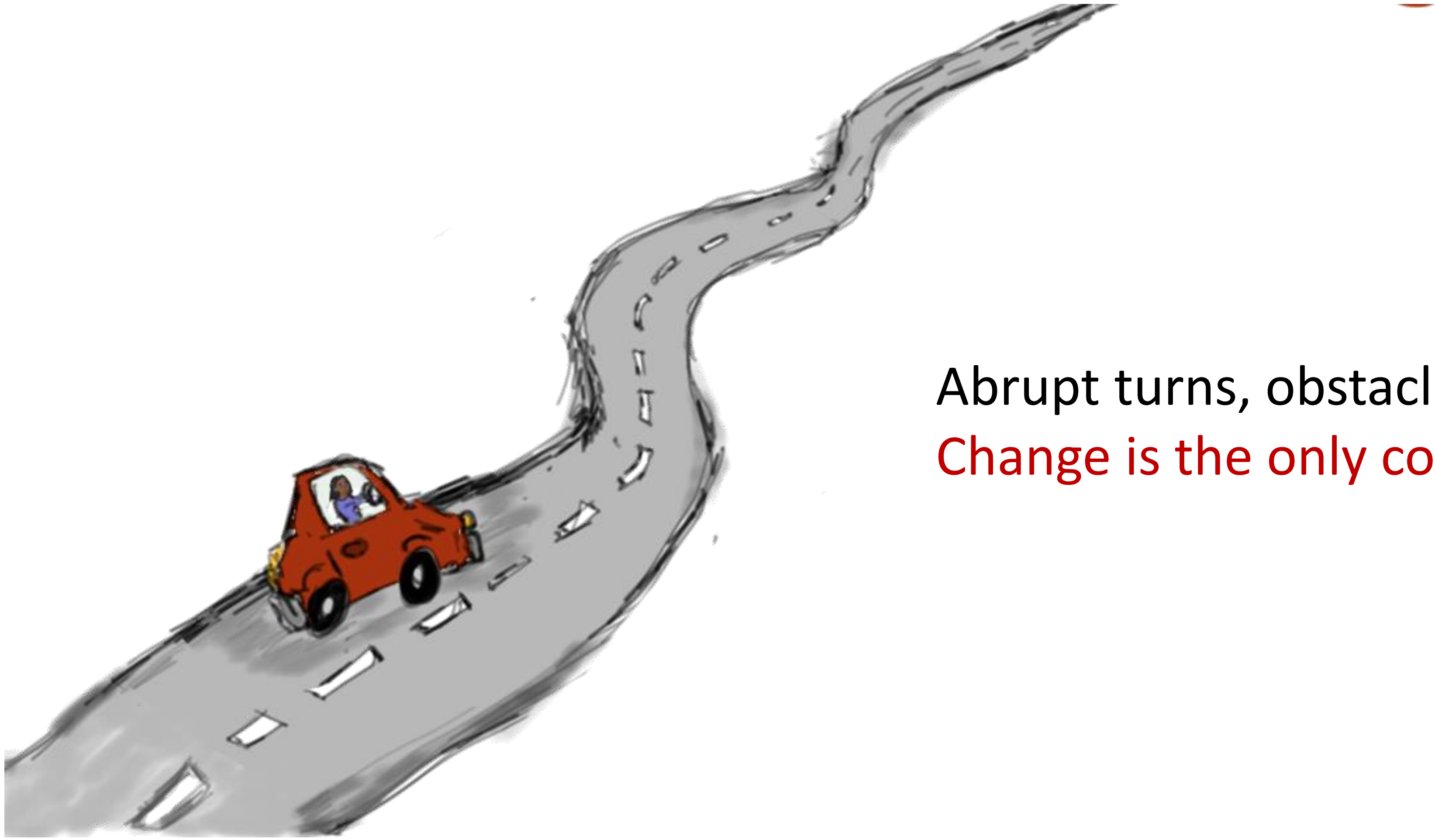
Discipline - Set practices to follow

Humanistic – Developers/customers center of process

Software Development

Slide adapted from Alessandro Orso

Developing is like driving



Abrupt turns, obstacles

Change is the only constant

Slide adapted from Alessandro Orso

Mentality of Sufficiency

How would you program if you had all the time in the world?

- Write tests
- Restructure Often
- Talk with fellow programmers and with customers often

Slide adapted from Alessandro Orso

XP’s values and principles

Communication

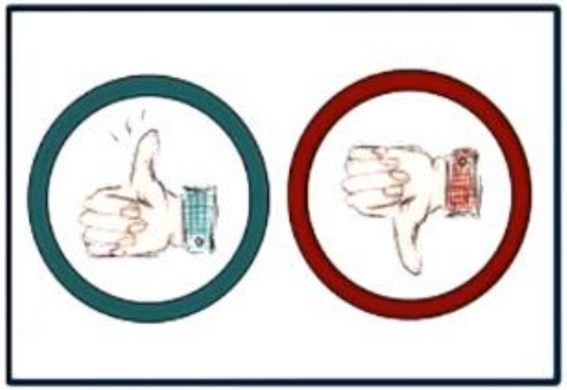
(right communication flowing, user stories, customer involvement)

Simplicity

Look for the simplest thing that works

Feedback

From test cases and customers



Courage

To change, improve, discard, try new things, build and test quickly

Slide adapted from Alessandro Orso

XP’s practices

1) Incremental Planning
2) Small releases
3) Simple Design
4) Test First
5) Refactoring
6) Pair Programming
7) Continuous integration
8) On-site customer

…

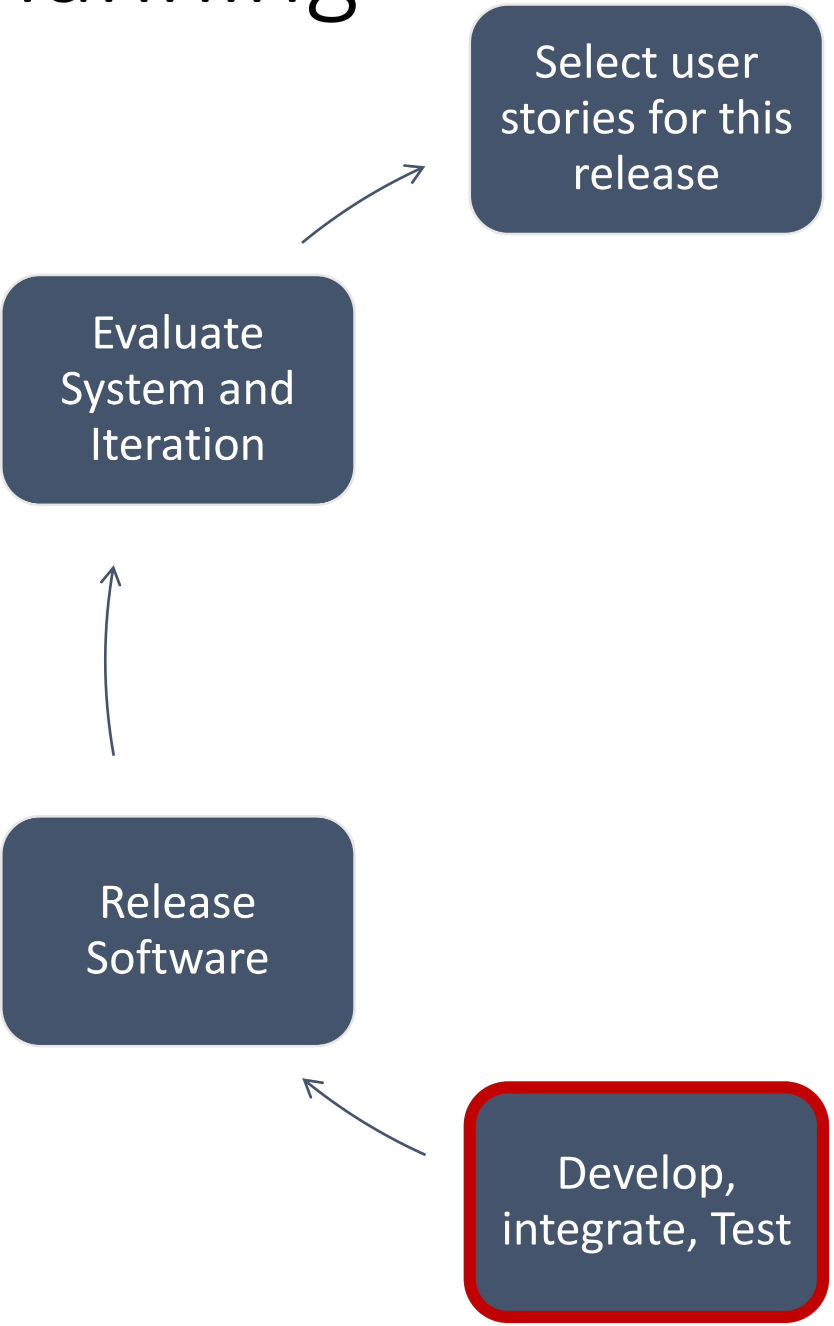
Slide adapted from Alessandro Orso

Attendance Time!

https://bit.ly/3BtxZXT

Slide adapted from Alessandro Orso

Incremental Planning



Select user stories for this release

Break Stories into tasks

Plan release

Develop, integrate, Test

Release Software

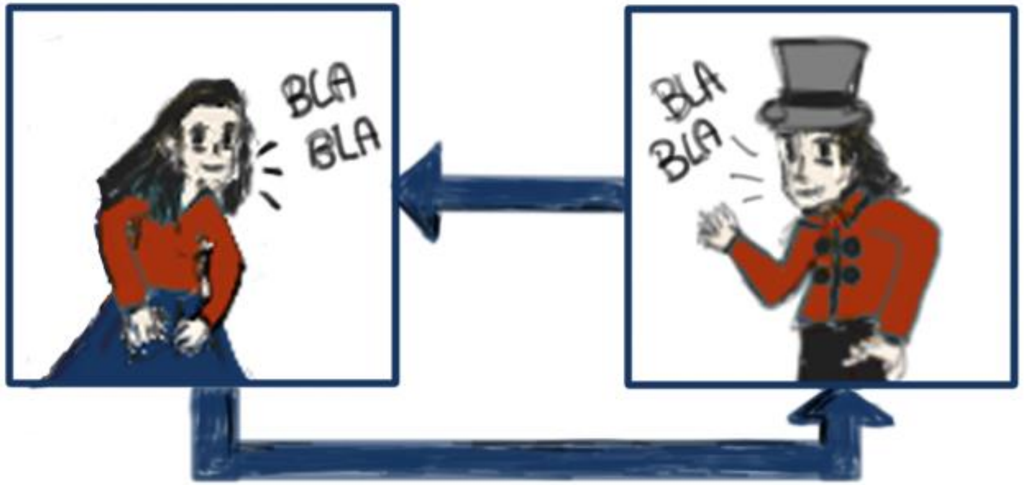
Evaluate System and Iteration

Slide adapted from Alessandro Orso

Small Releases

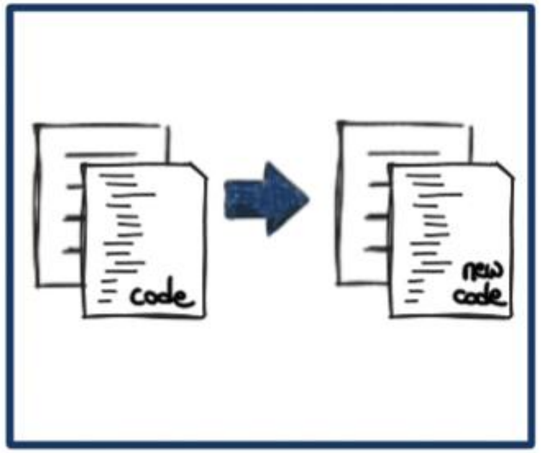
Deliver real business value

Rapid Feedback, Quick Changes



Sense of achievement for developer

Reduce Risks

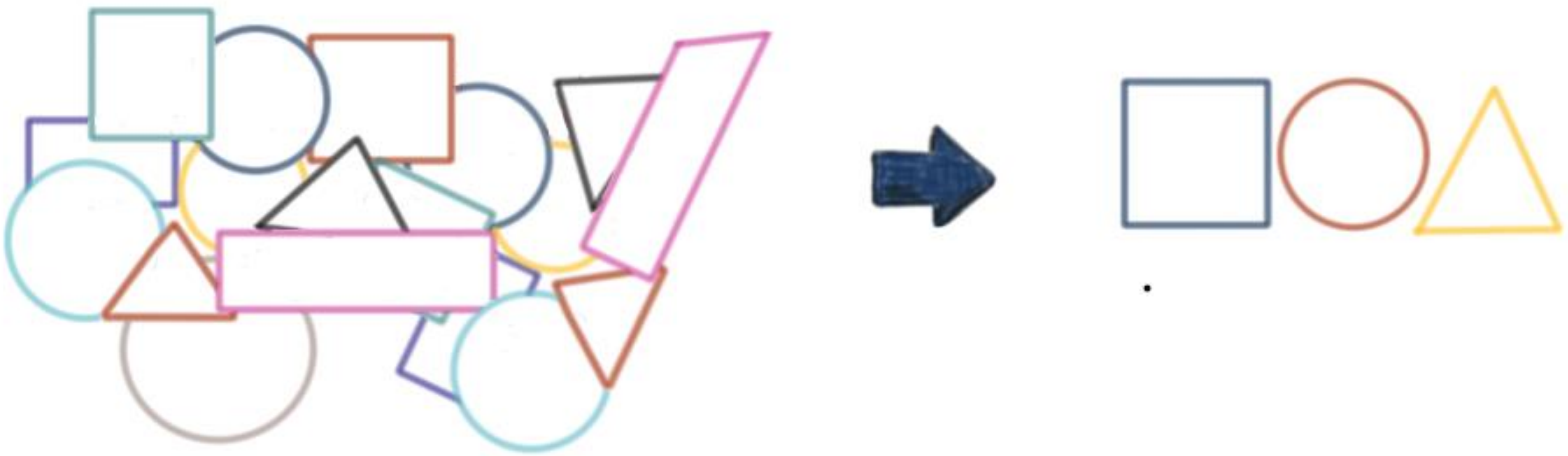


Quickly Adapt

Slide adapted from Alessandro Orso

Simple Design

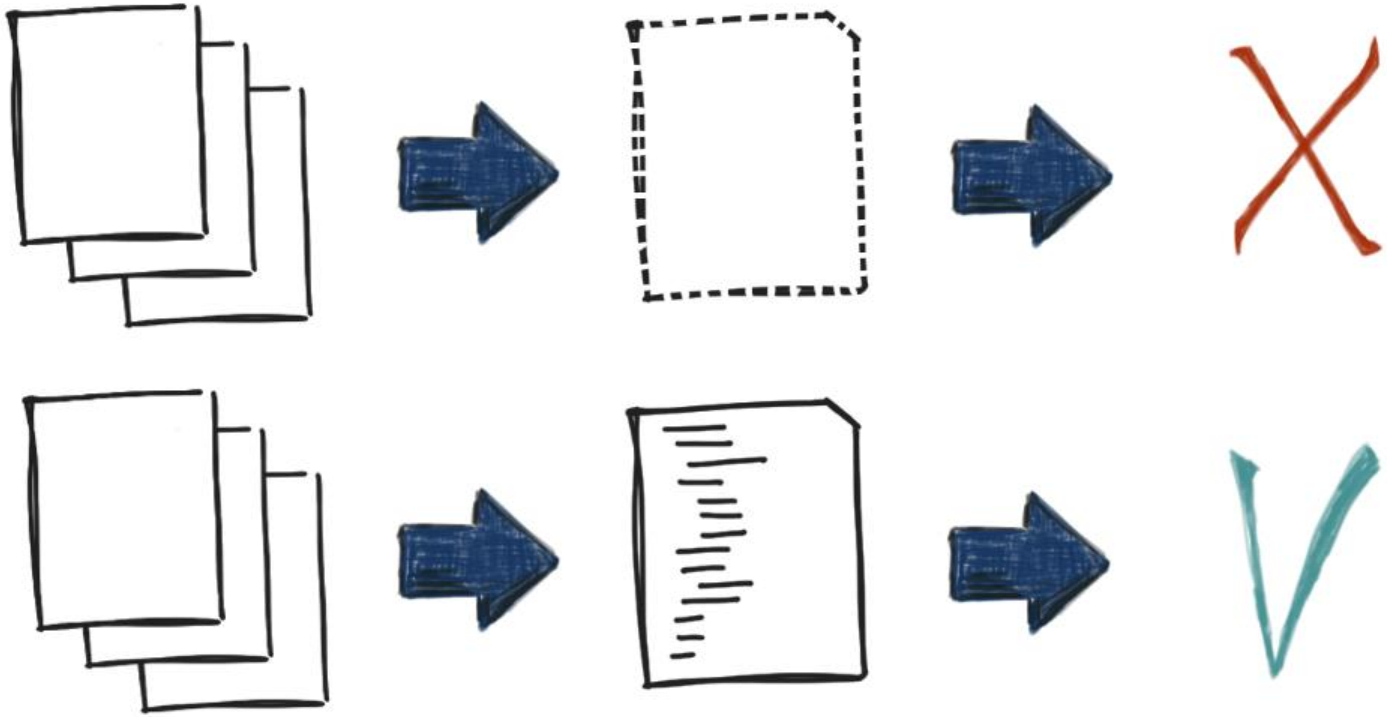
- No complicated design right away
- Simple design enough to meet requirements
- No duplicate functionality
- Fewest possible Classes and Methods



Slide adapted from Alessandro Orso

Test- First Development

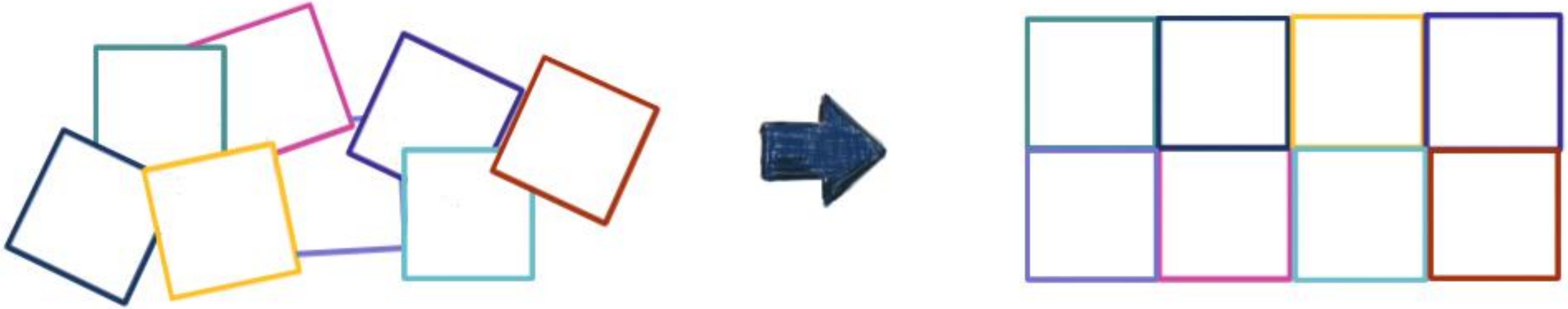
- Any program feature that doesn't have an automatic test doesn't exist
- Develop unit tests for each piece of functionality before implementing the functionality
- Immediate feedback on the implementation



Slide adapted from Alessandro Orso

Refactoring

- Suboptimal design (because of evolving, adding features in a certain way) => Restructure it
- Make the code simple and maintainable
- As soon as opportunities for improvement, before or after changes
- Refactor on demand, on the system and the process needed



Slide adapted from Alessandro Orso

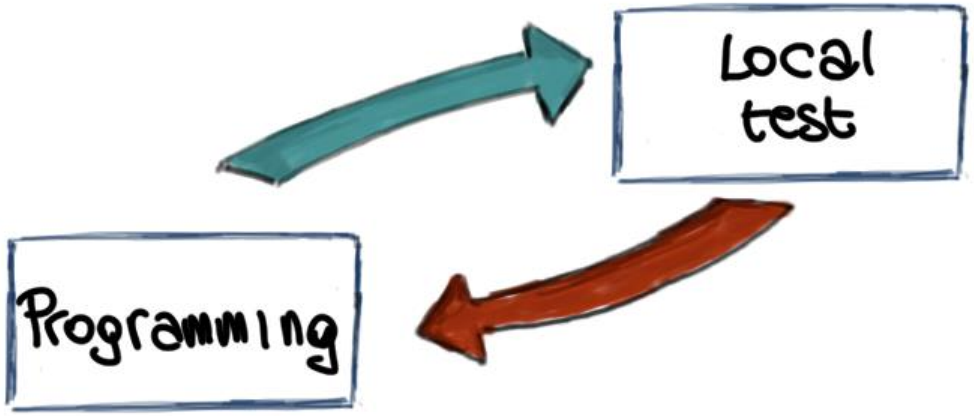
Pair Programming

- All production code with 2 people on 1 machine
- Different roles- Programming ⇔ Strategizing (what tests might work? Can code be refactored?)
- Studies suggest development productivity with pair programming is similar to that of 2 people working independently

Slide adapted from Alessandro Orso

Continuous Integration

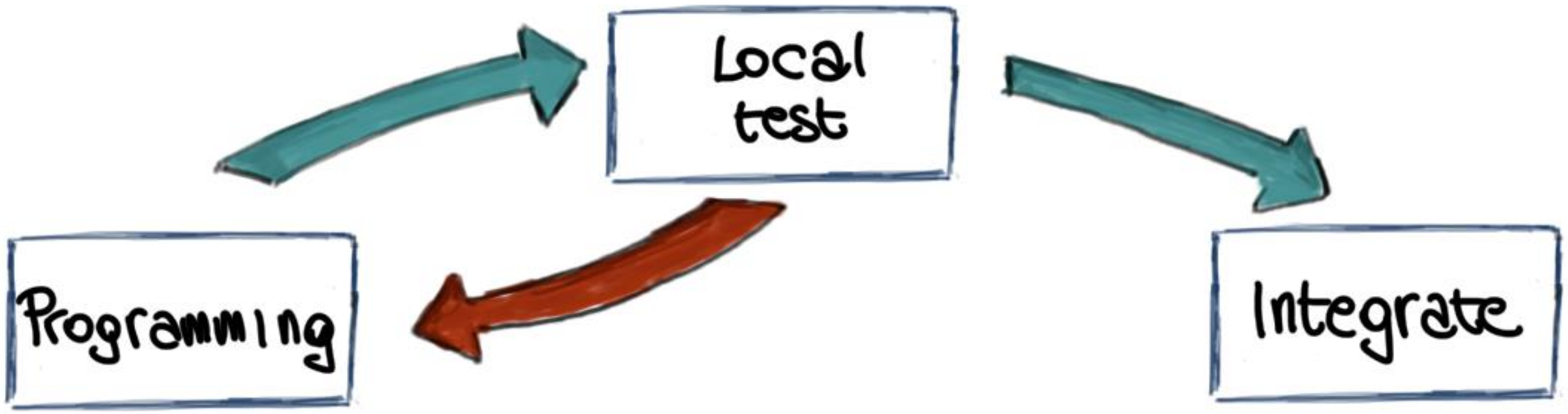
- Integrate and test every few hours or everyday
- No integration nightmare



Slide adapted from Alessandro Orso

Continuous Integration

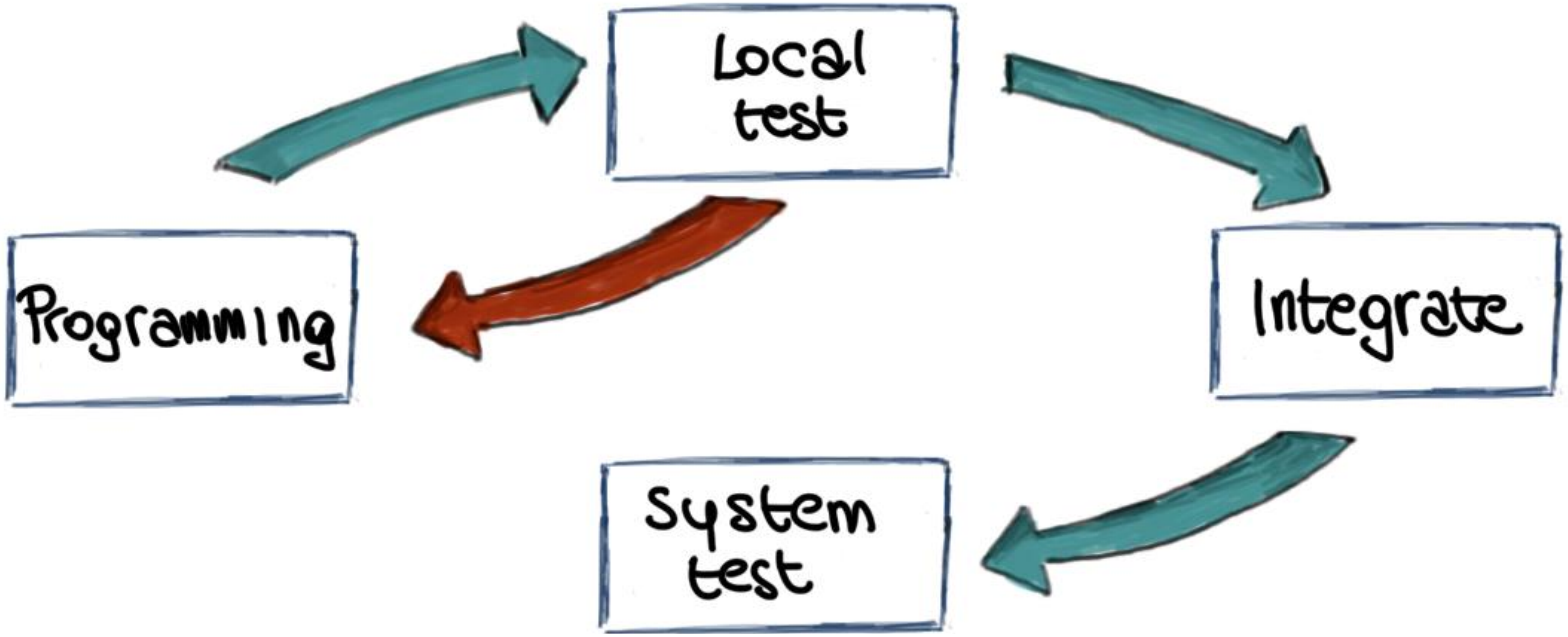
- Integrate and test every few hours or everyday
- No integration nightmare



Slide adapted from Alessandro Orso

Continuous Integration

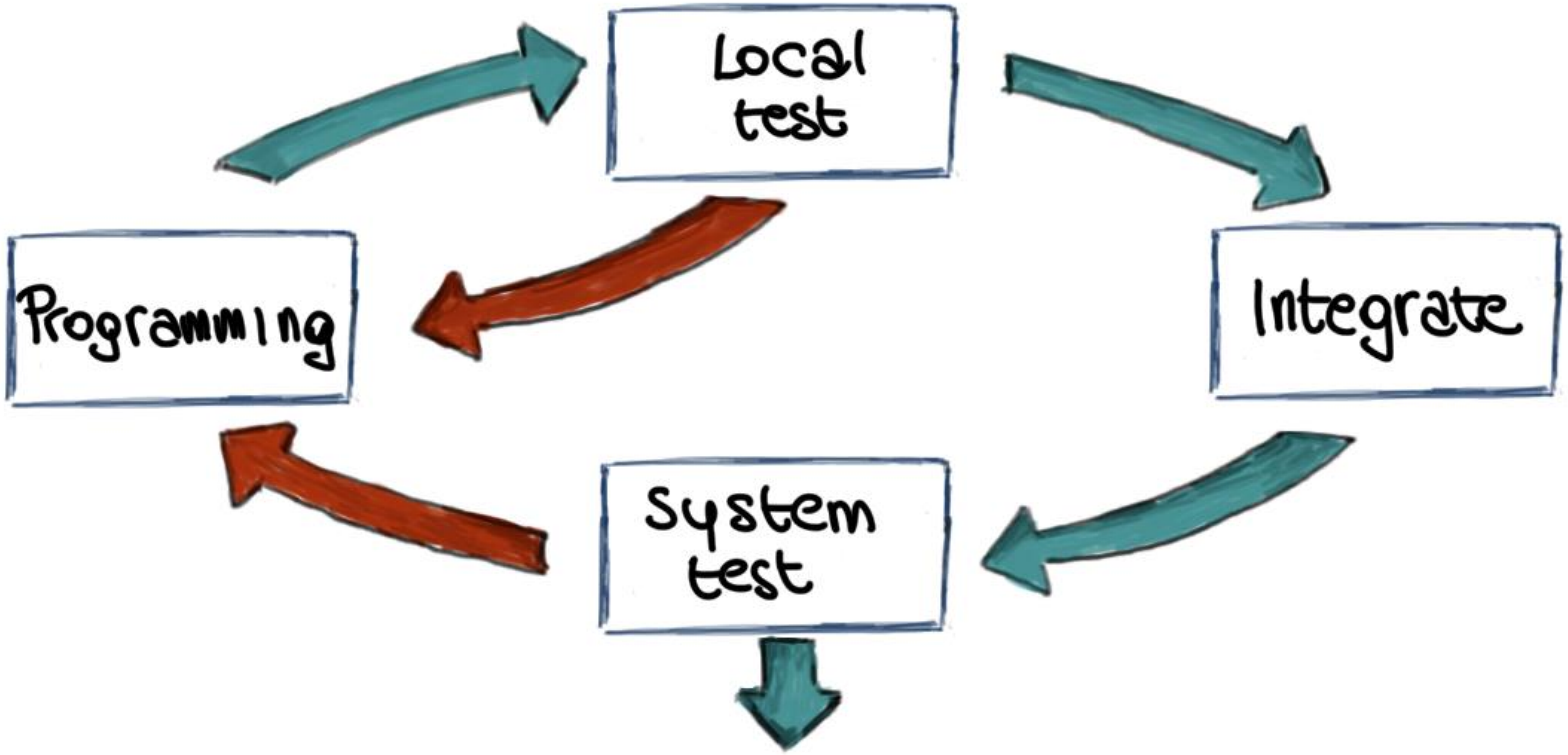
- Integrate and test every few hours or everyday
- No integration nightmare



Slide adapted from Alessandro Orso

Continuous Integration

- Integrate and test every few hours or everyday
- No integration nightmare



Slide adapted from Alessandro Orso

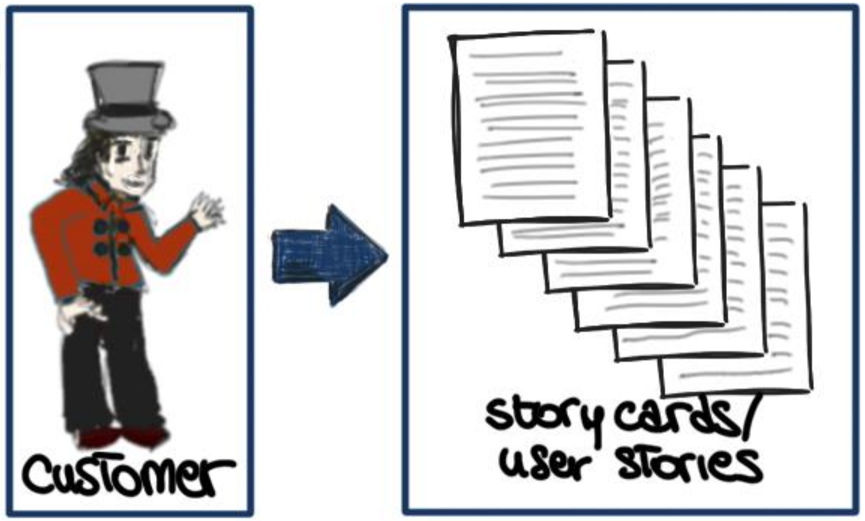
On-Site Customer

- The customer is an actual member of the team
  - Sits with the team
  - Brings requirements
- System should be worth involving 1 customer at all times

Slide adapted from Alessandro Orso

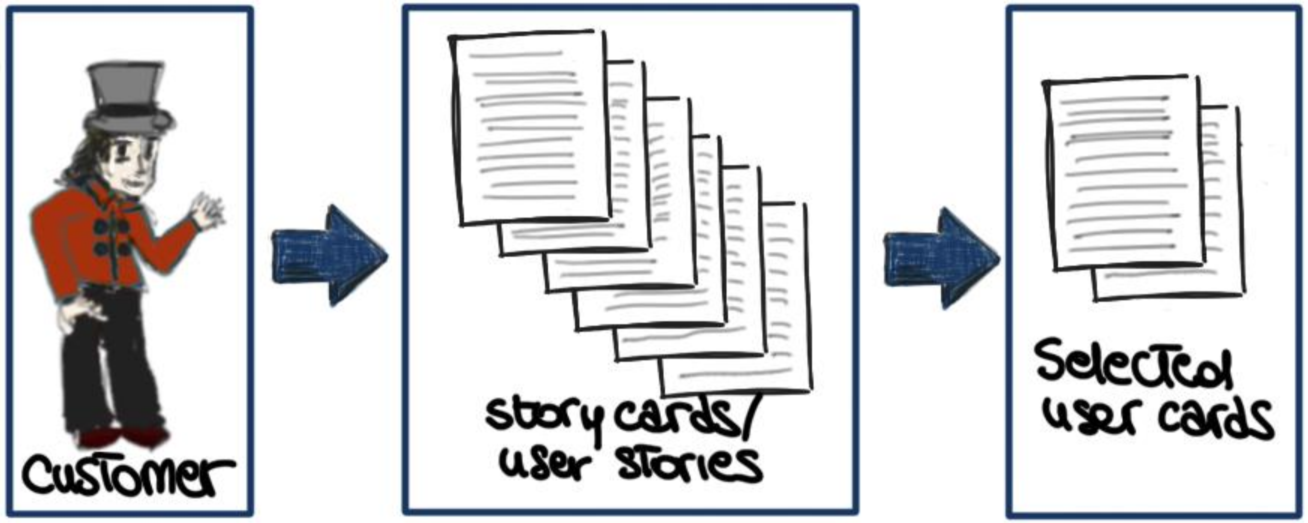
Requirements Engineering in XP

Requirements Engineering in XP

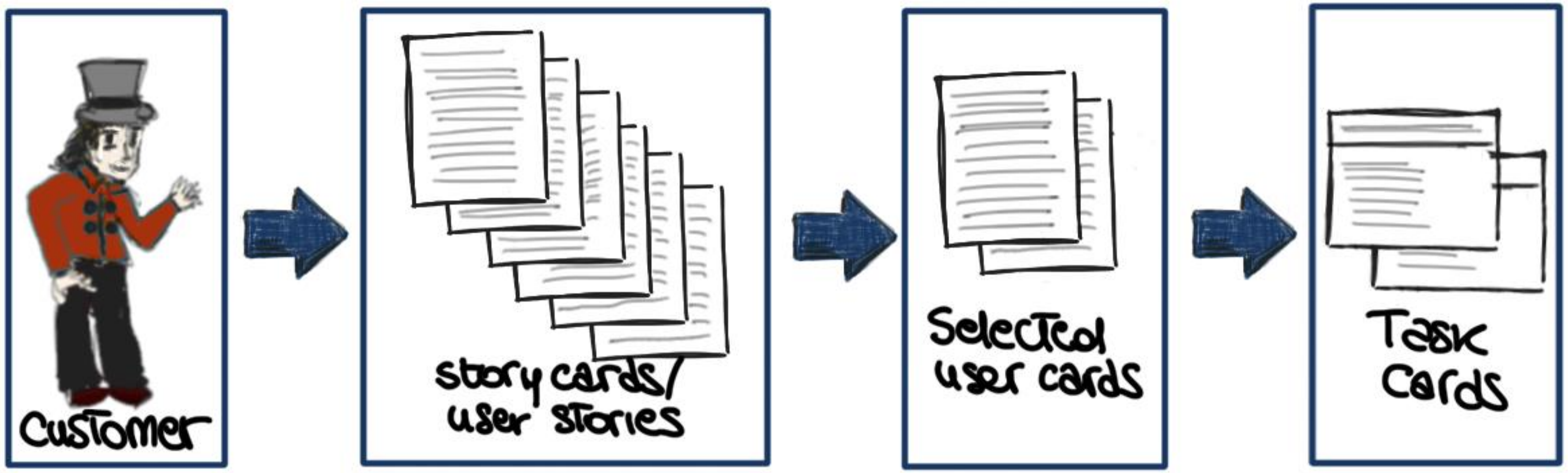


Scenarios of user stories => Implementation tasks => Scheduling/ cost estimates

Requirements Engineering in XP

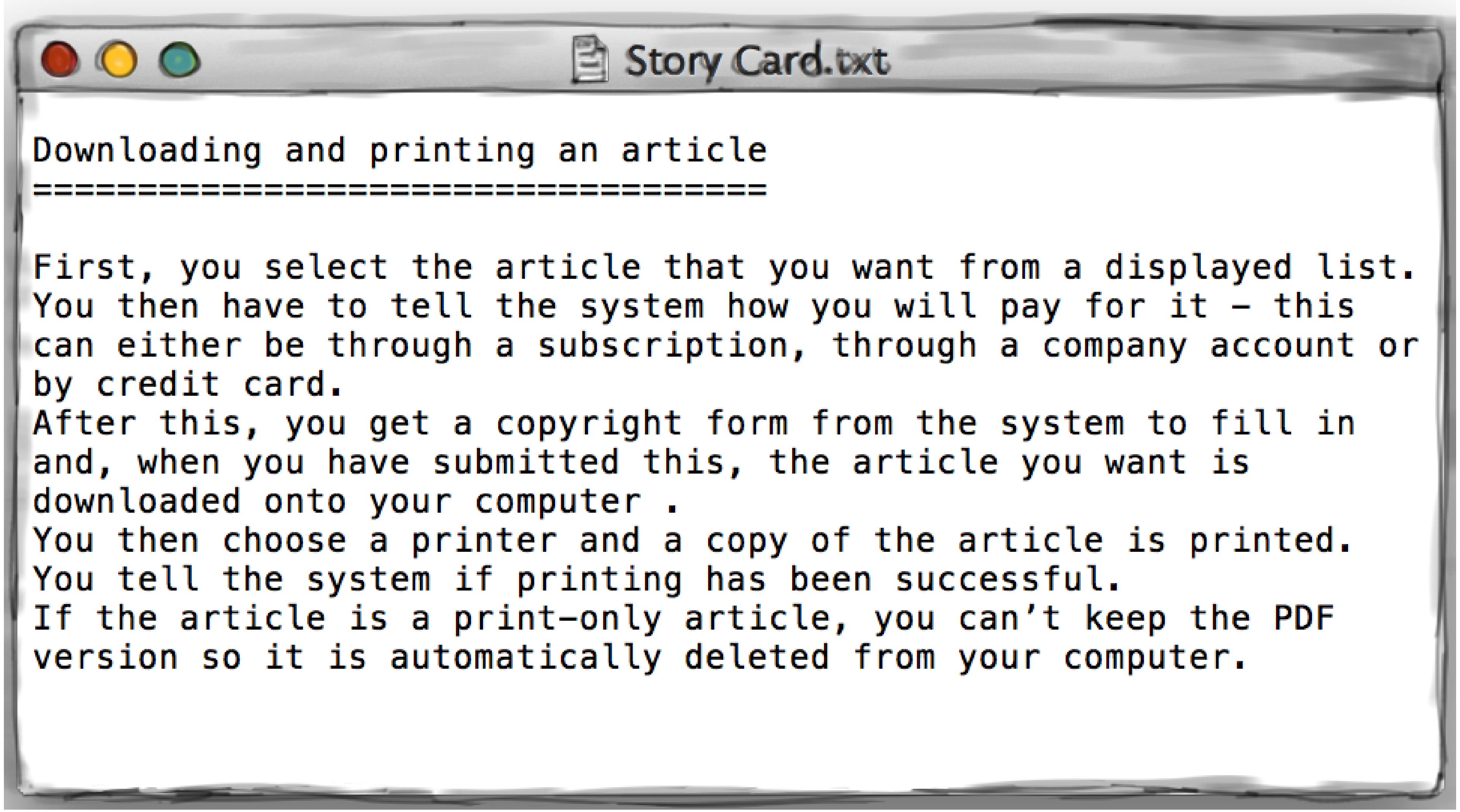


Requirements Engineering in XP



⇒ For a few months' projects, there can be around 50 to 100 user stories

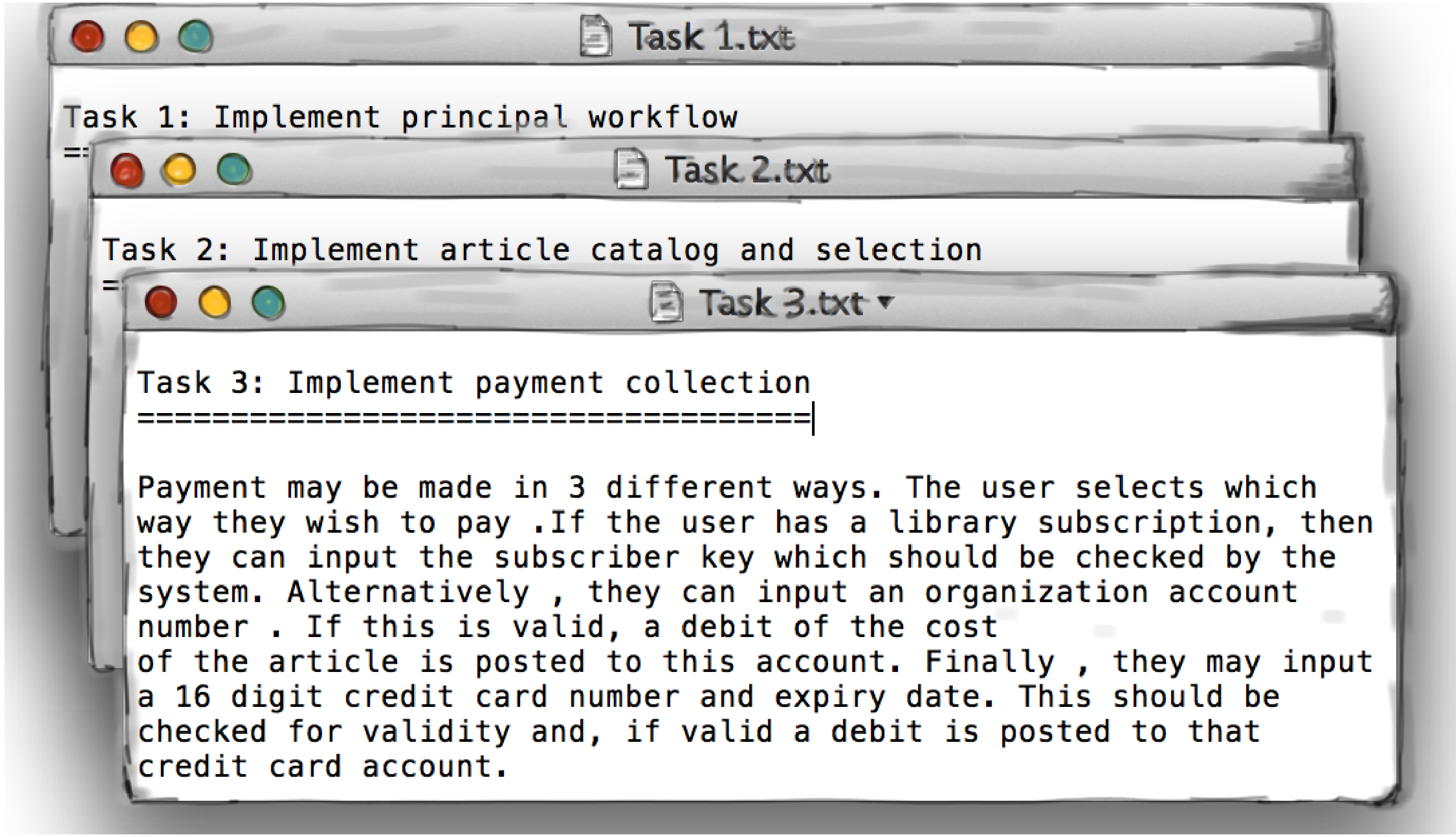
Story Card For Document Downloading



```
Downloading and printing an article
===================================

First, you select the article that you want from a displayed list.
You then have to tell the system how you will pay for it - this
can either be through a subscription, through a company account or
by credit card.
After this, you get a copyright form from the system to fill in
and, when you have submitted this, the article you want is
downloaded onto your computer .
You then choose a printer and a copy of the article is printed.
You tell the system if printing has been successful.
If the article is a print-only article, you can’t keep the PDF
version so it is automatically deleted from your computer.
```

Task Cards For Document Downloading



```
Task 1: Implement principal workflow
==
```

```
Task 2: Implement article catalog and selection
=
```

```
Task 3: Implement payment collection
====================================

Payment may be made in 3 different ways. The user selects which
way they wish to pay .If the user has a library subscription, then
they can input the subscriber key which should be checked by the
system. Alternatively , they can input an organization account
number . If this is valid, a debit of the cost
of the article is posted to this account. Finally , they may input
a 16 digit credit card number and expiry date. This should be
checked for validity and, if valid a debit is posted to that
credit card account.
```

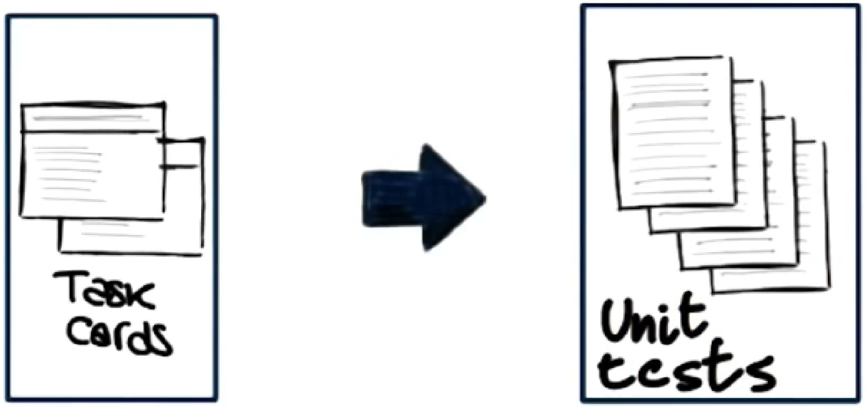
Can be more specific and talk about development tasks

Testing Strategy

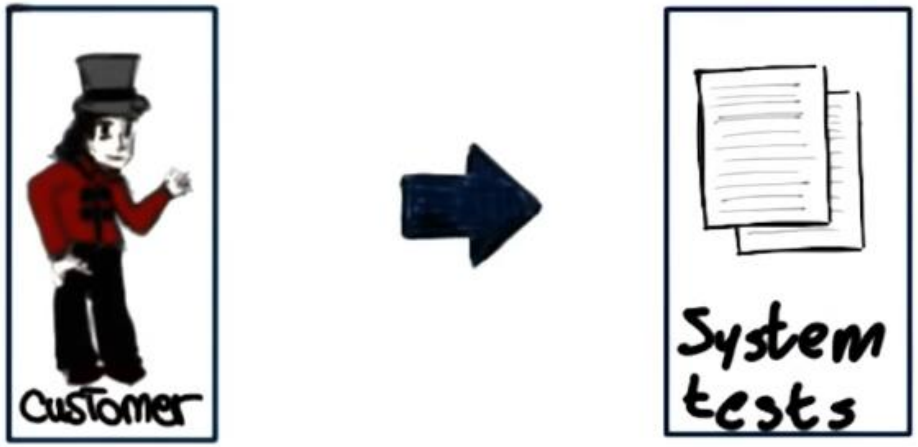
TESTING IS CODED CONFIDENCE

Tests should be isolated and automated

Testing Strategy – Two types



Test every meaningful feature
Special cases/ Specific problems in the task cards
May include refactoring



Customer provides test cases for their stories
Developer converts them to automated tests
Run longer and less frequent
Run every time system is integrated

XP Testing Quiz

Which of the following statements about XP are true?

[ ] Because of pair programming, XP requires twice the number of developers

[ ] In XP, code is rarely changed after being written

[✓] XP follows the test-driven development (TDD) paradigm

[ ] The customer does not need to provide any requirements in XP

[✓] XP is an iterative software development process

Scrum

Another agile development Process. Most popular in industry.

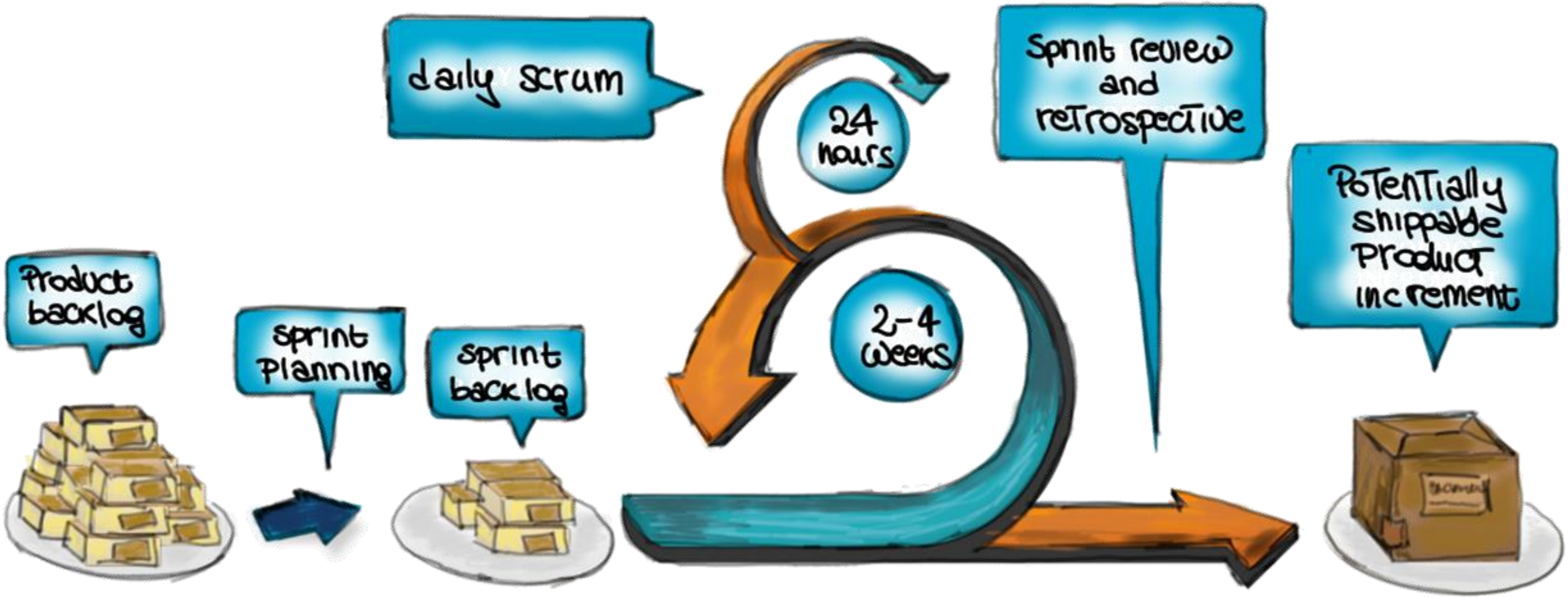
Scrum Actors

Product Owner/Customer- Backlog is the list of things that need to be done (user stories in XP). Clearly express backlog item and order them by value.

Team - responsible for delivering shippable increments and estimating backlog items.

Scrum Master - responsible for managing overall scrum process, remove obstacles, facilitate events, help communication

Scrum- High Level Process



- Living list of requirements
- Ordered by value
- From customer/product owner

Backlog items to be completed in the next sprint

Sprint Is iteration of scrum process. Main part- 2-4 weeks

- 4-hour meeting
- Product owner assesses accomplishments/Issues
- Demo
- Backlogs for next sprint
- Retrospective - Process improvements