SOFTWARE TESTING

TEST-DRIVEN DEVELOPMENT

Mahdi Roozbahani

Slides are based on Alex Orso.

FROM WATERFALL ...

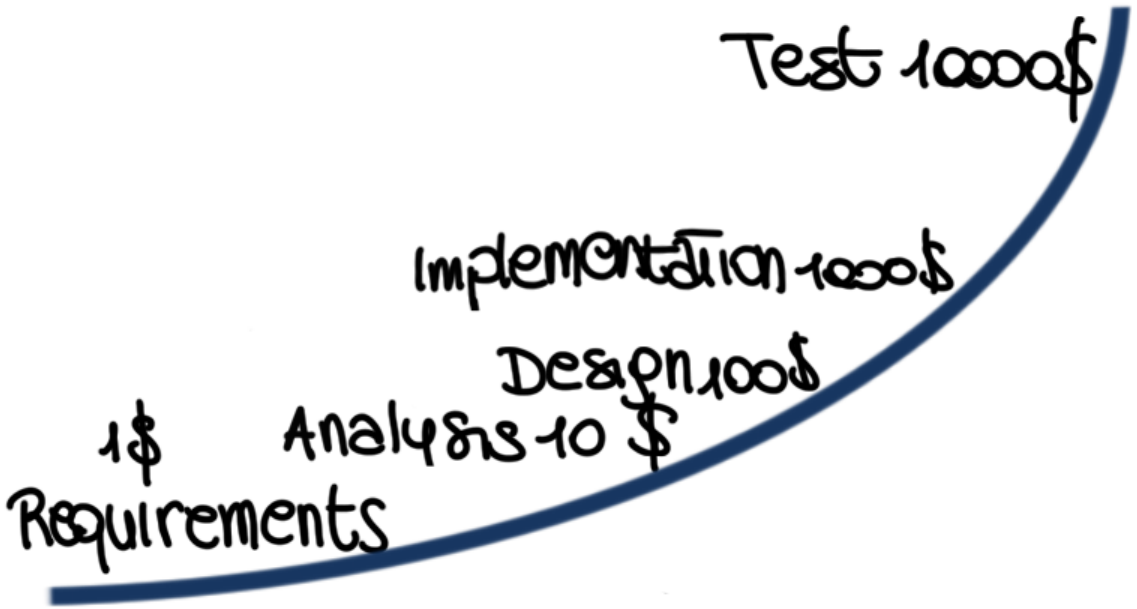
FROM WATERFALL ...

... TO AGILE

AS BOEHM SAID...

Cost of change grows exponentially with time

AS BOEHM SAID...



Cost of change grows exponentially with time

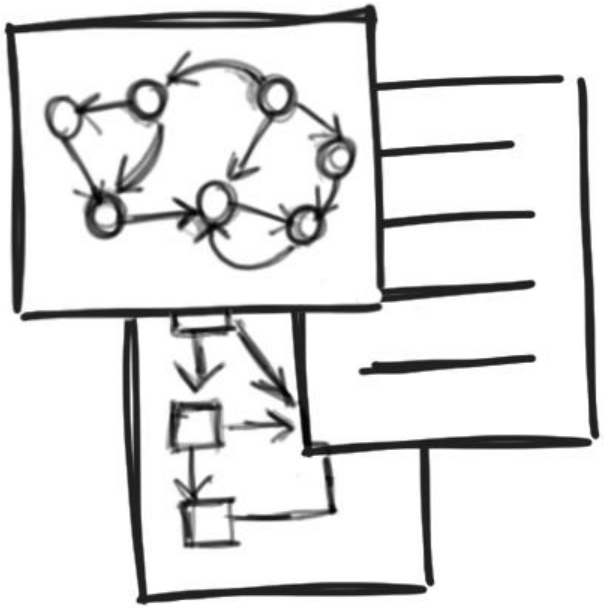
WHAT TO DO, THEN ?

WHAT TO DO, THEN ?

Discover errors early ⇒ upfront planning

WHAT TO DO, THEN ?

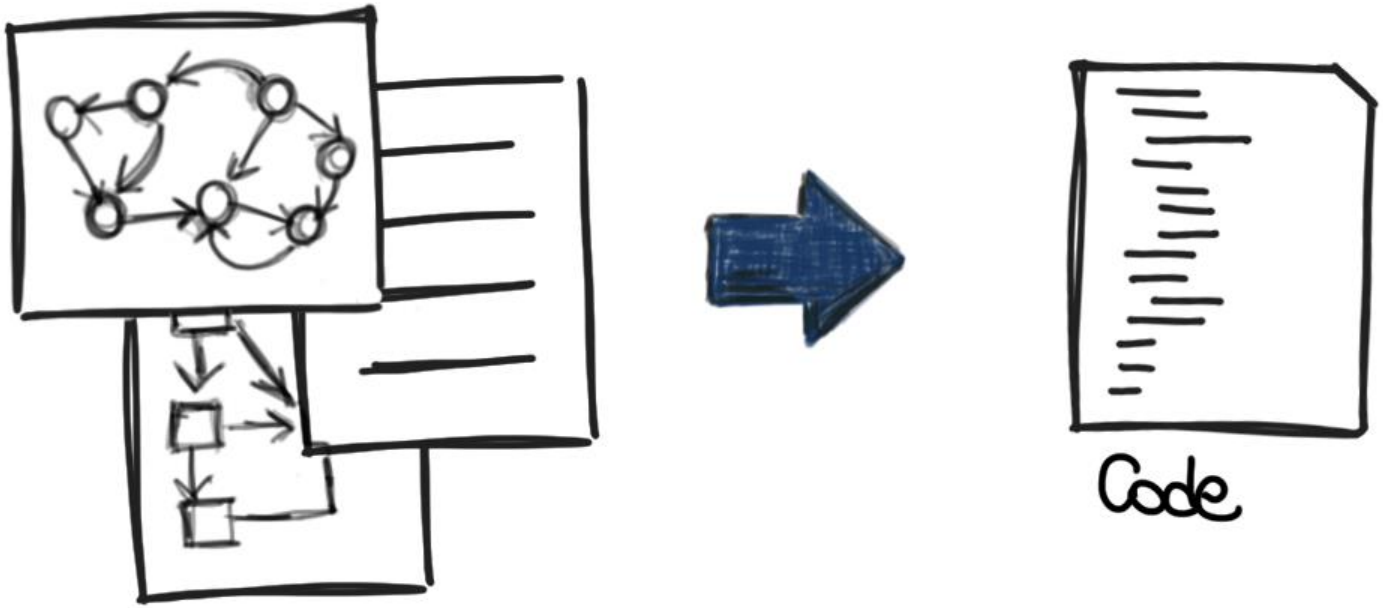
Discover errors early ⇒ upfront planning



Model documents

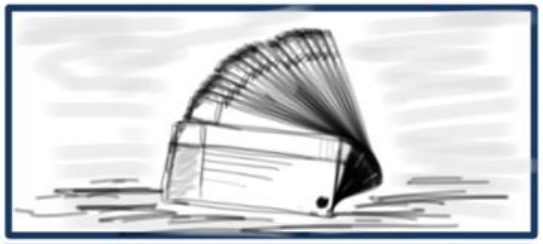
WHAT TO DO, THEN ?

Discover errors early ⇒ upfront planning

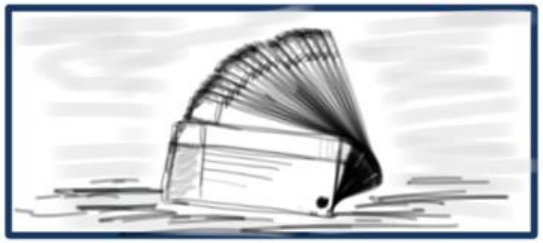


SOMETHING CHANGED IN THE LAST 30 YEARS

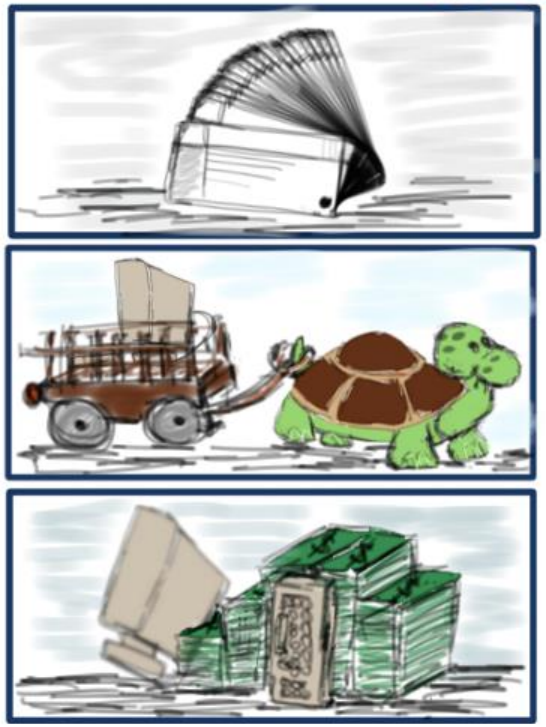
SOMETHING CHANGED IN THE LAST 30 YEARS



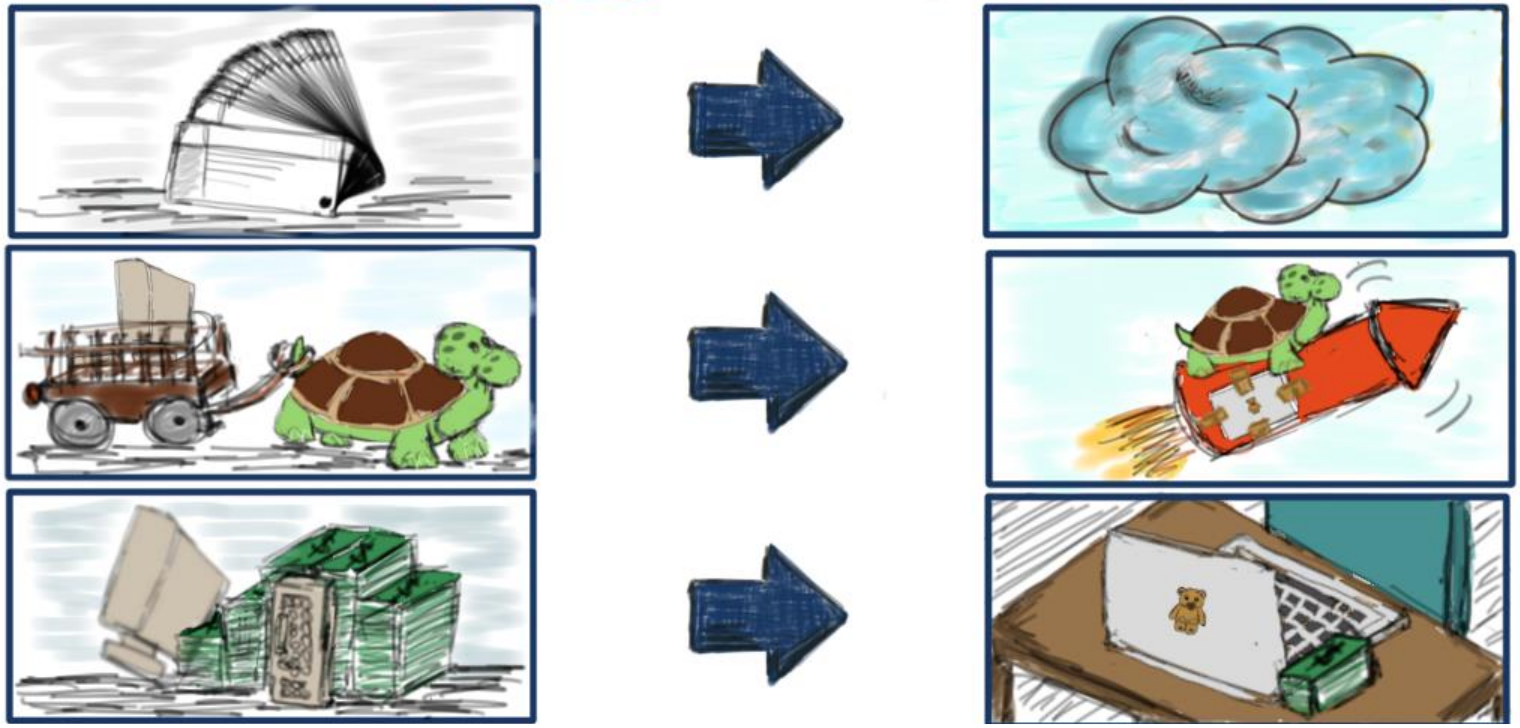
SOMETHING CHANGED IN THE LAST 30 YEARS



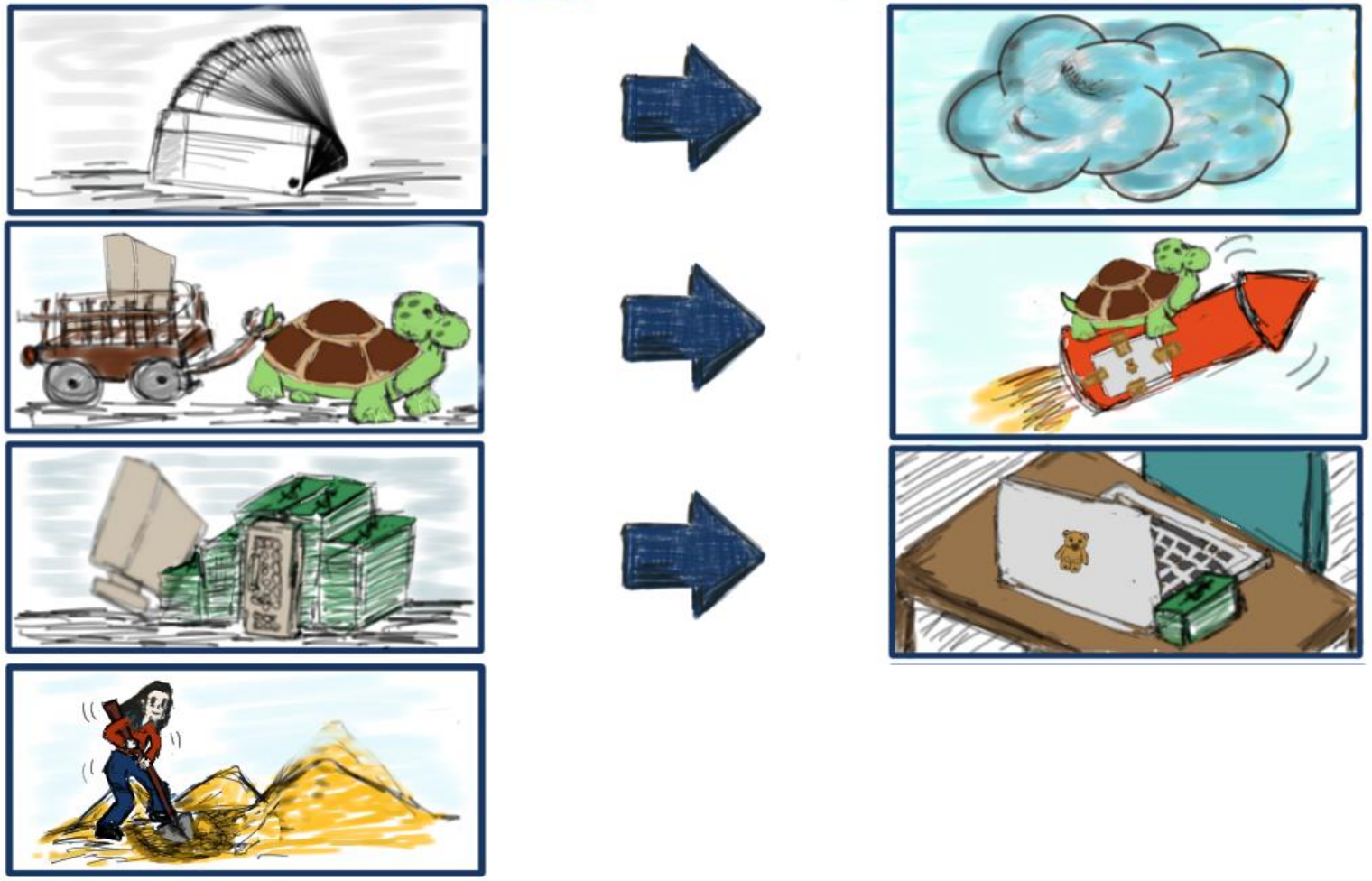
SOMETHING CHANGED IN THE LAST 30 YEARS



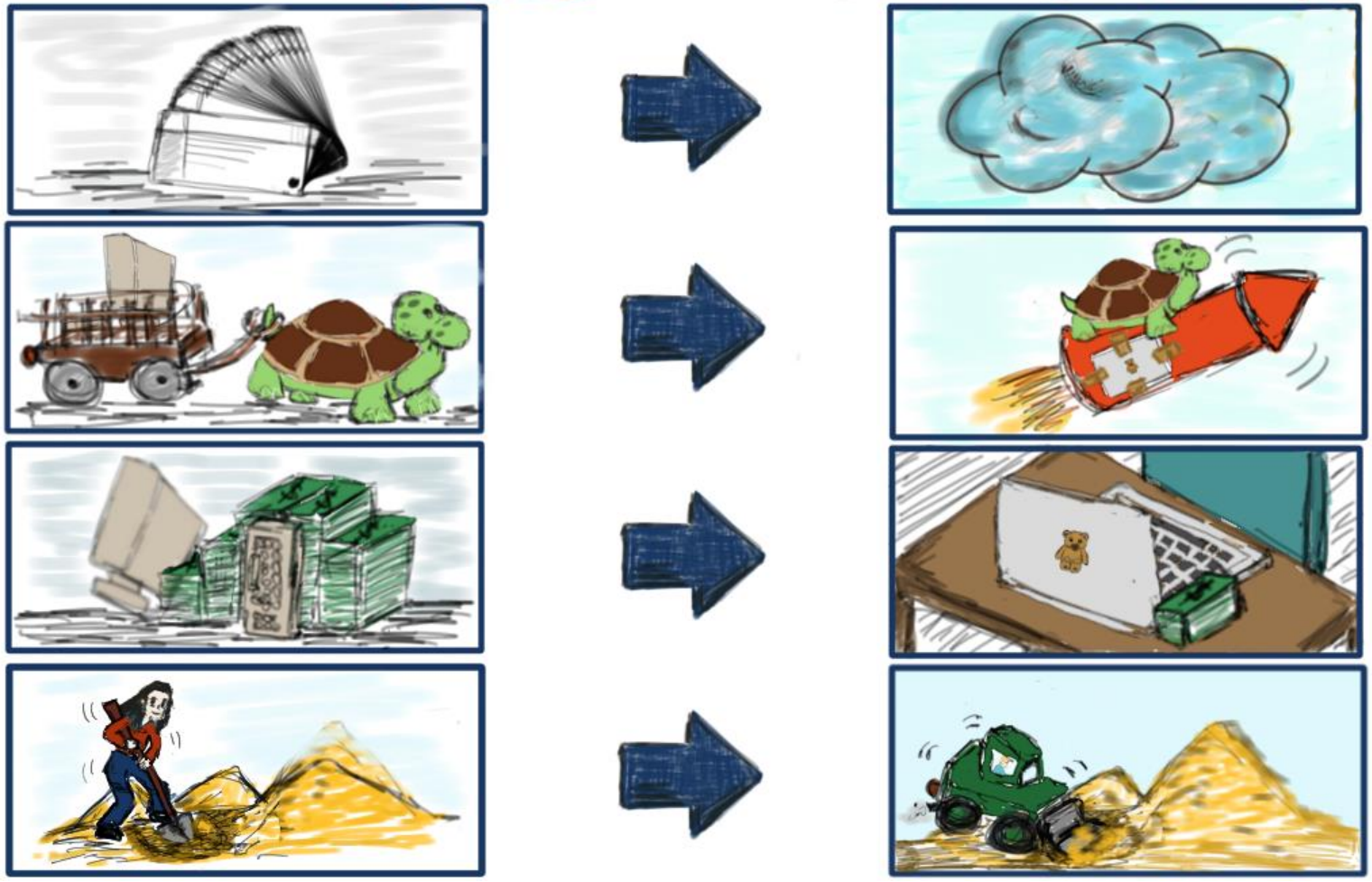
SOMETHING CHANGED IN THE LAST 30 YEARS



SOMETHING CHANGED IN THE LAST 30 YEARS



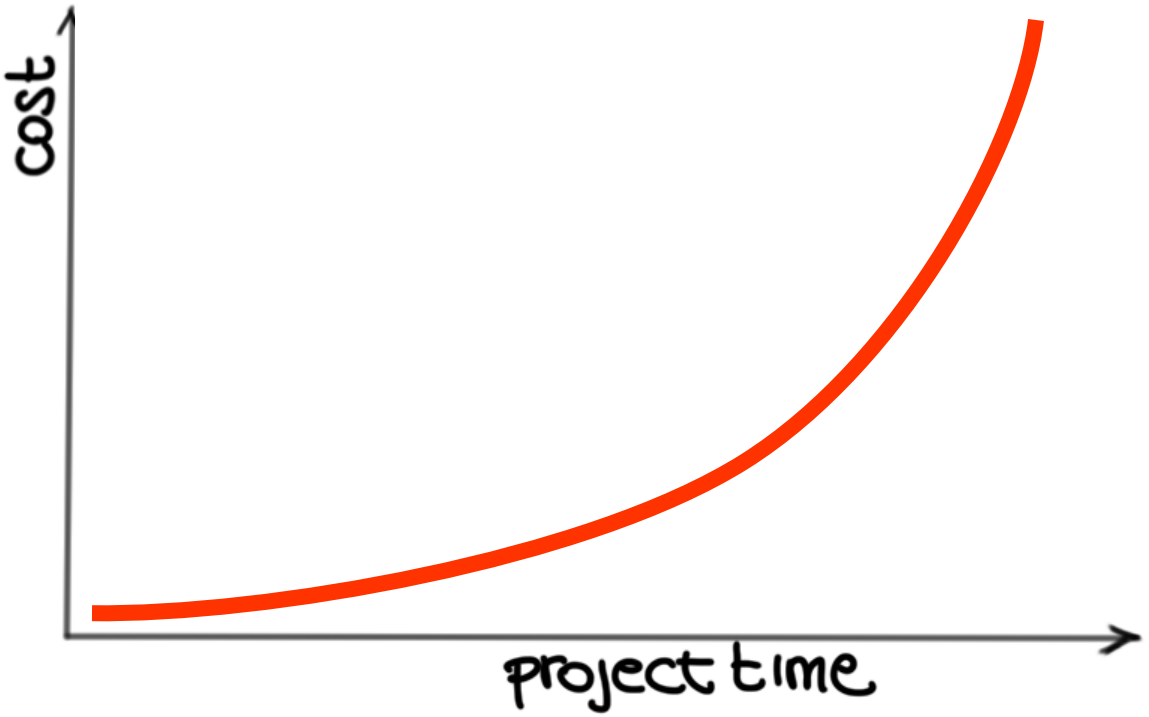
SOMETHING CHANGED IN THE LAST 30 YEARS



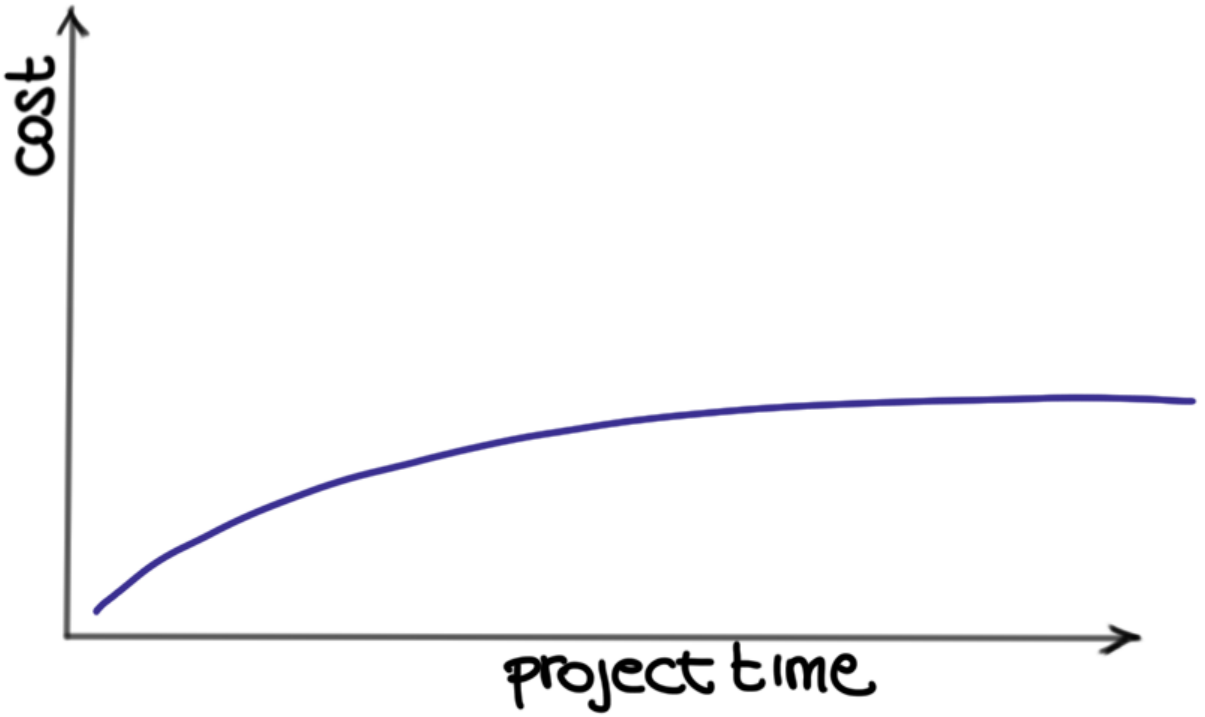
SOMETHING CHANGED IN THE LAST 30 YEARS

MAYBE THE COST OF CHANGE CAN BE FLAT?

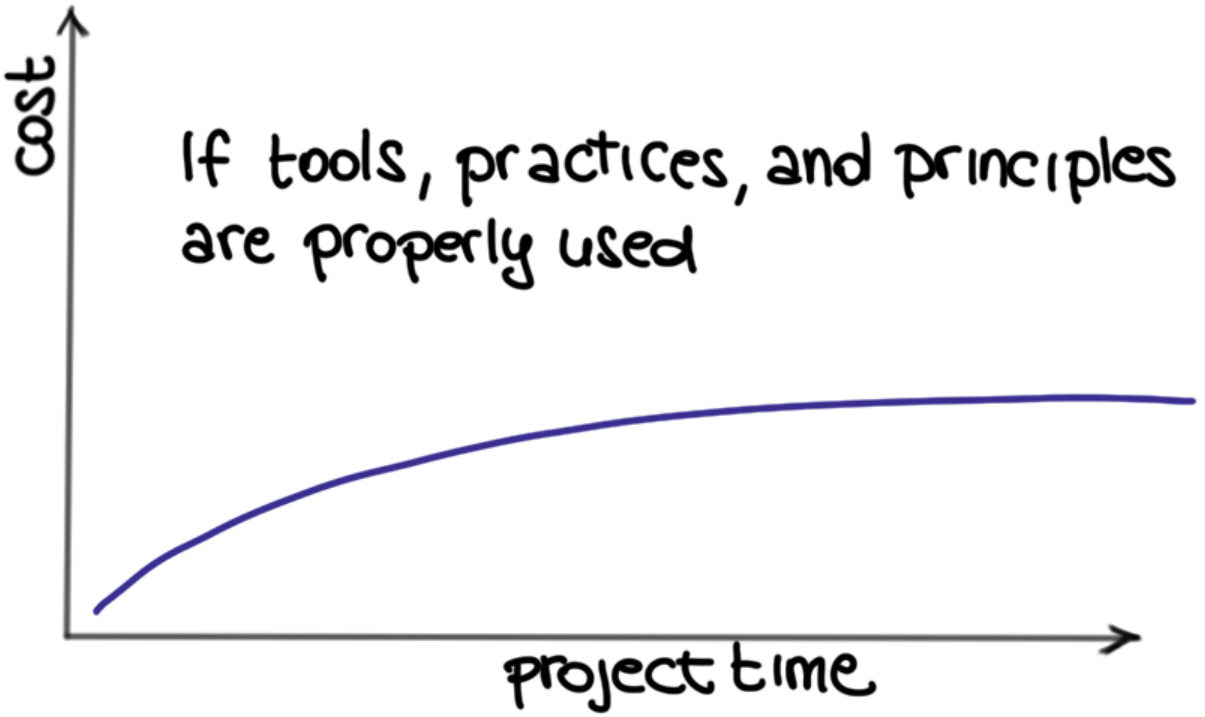
MAYBE THE COST OF CHANGE CAN BE FLAT?



MAYBE THE COST OF CHANGE CAN BE FLAT?



MAYBE THE COST OF CHANGE CAN BE FLAT?



IF COST IS FLAT...

IF COST IS FLAT...

Upfront work == liability

IF COST IS FLAT...

Upfront work == liability

Ambiguity, volability ⇒ good to delay

IF COST IS FLAT...

Upfront work == liability

Ambiguity, volability ⇒ good to delay

There is value in waiting!

AGILE METHODS AIM AT FLAT COST

AGILE METHODS AIM AT FLAT COST

Manifesto for agile software development

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

AGILE METHODS AIM AT FLAT COST

AGILE METHODS AIM AT FLAT COST

Focus on the code

AGILE METHODS AIM AT FLAT COST

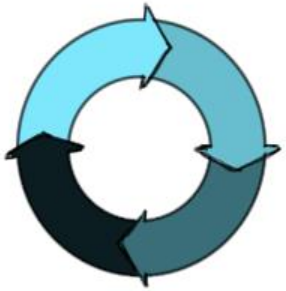
Focus on the code

People over process

AGILE METHODS AIM AT FLAT COST

Focus on the code

People over process

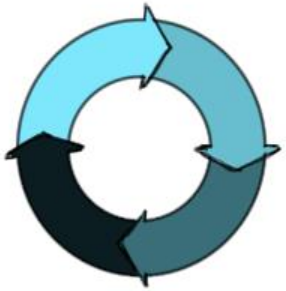


Iterative approach

AGILE METHODS AIM AT FLAT COST

Focus on the code

People over process



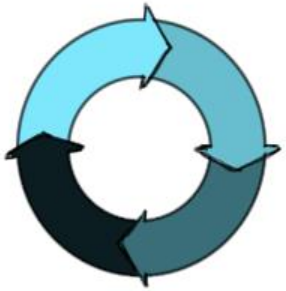
Iterative approach

Customer involvement

AGILE METHODS AIM AT FLAT COST

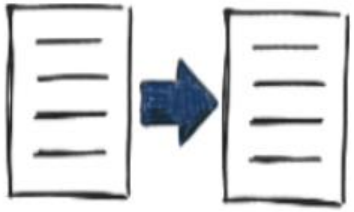
Focus on the code

People over process



Iterative approach

Customer involvement



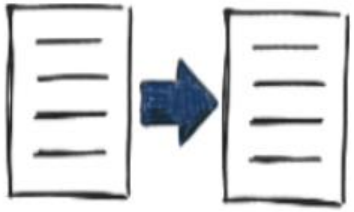
Expectation that requirements will change

AGILE METHODS AIM AT FLAT COST

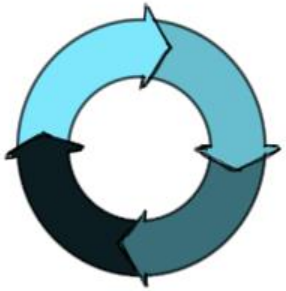
Focus on the code

Customer involvement

People over process



Expectation that requirements will change



Iterative approach

Simplicity

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

WHAT IS XP?

WHAT IS XP?

Lightweight

WHAT IS XP?

Lightweight

Humanistic

WHAT IS XP?

Lightweight

Discipline

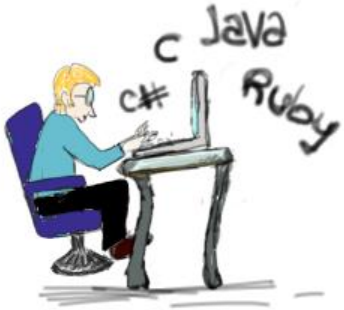
Humanistic

WHAT IS XP?

Lightweight

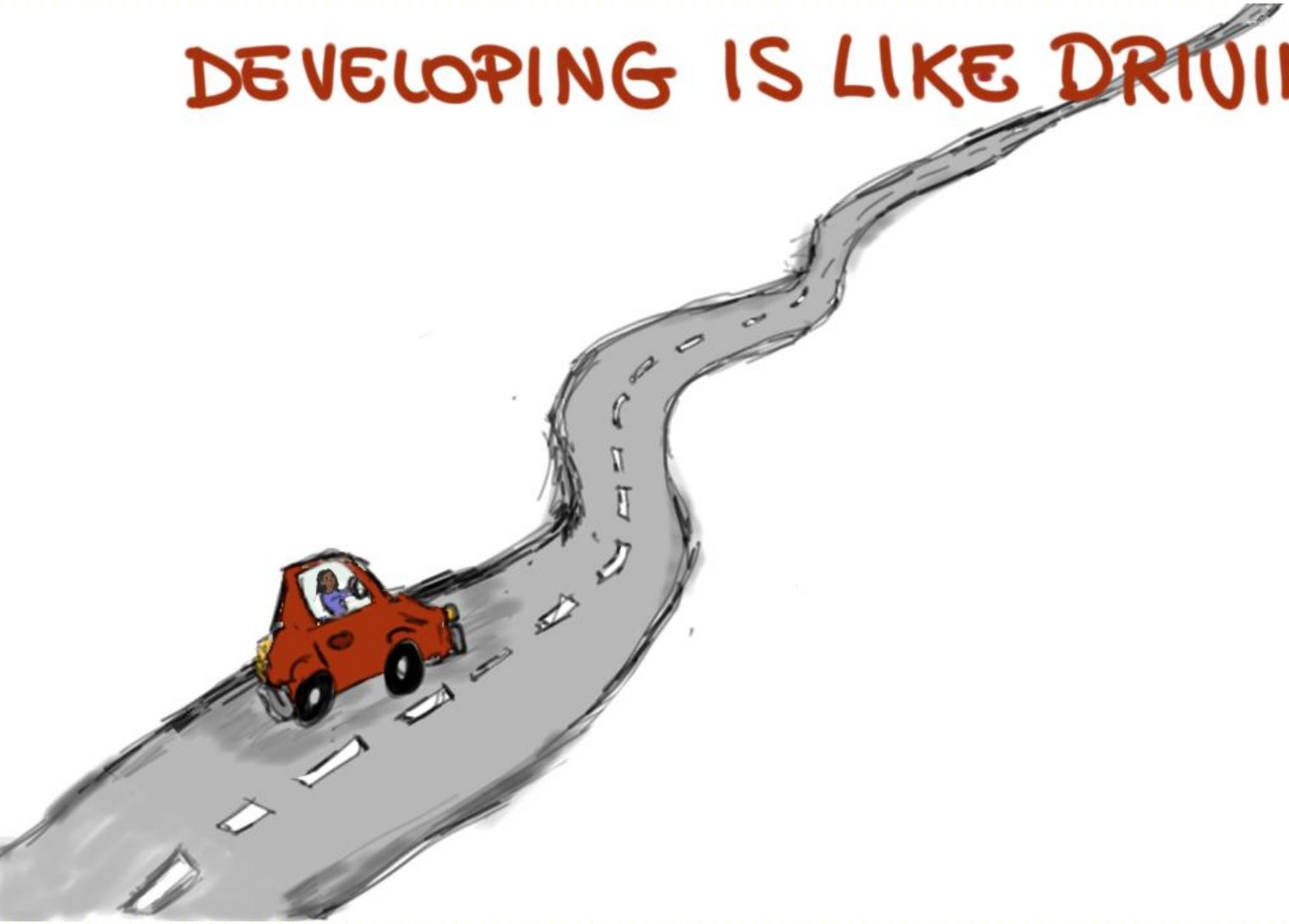
Discipline

Humanistic



Software development

DEVELOPING IS LIKE DRIVING



MENTALITY OF SUFFICIENCY

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

MENTALITY OF SUFFICIENCY

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

- Write tests

MENTALITY OF SUFFICIENCY

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

- Write tests
- Restructure often

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 the customer often

XP'S VALUES AND PRINCIPLES

XP'S VALUES AND PRINCIPLES

Communication

XP'S VALUES AND PRINCIPLES

Communication

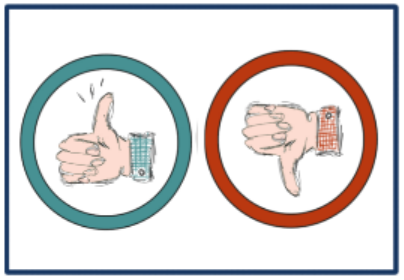
Simplicity

XP'S VALUES AND PRINCIPLES

Communication

Simplicity

Feedback

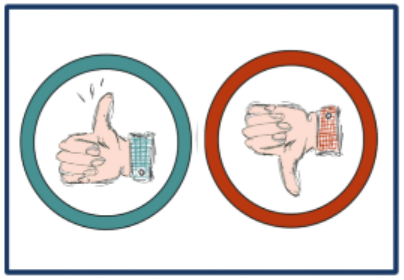


XP'S VALUES AND PRINCIPLES

Communication

Simplicity

Feedback



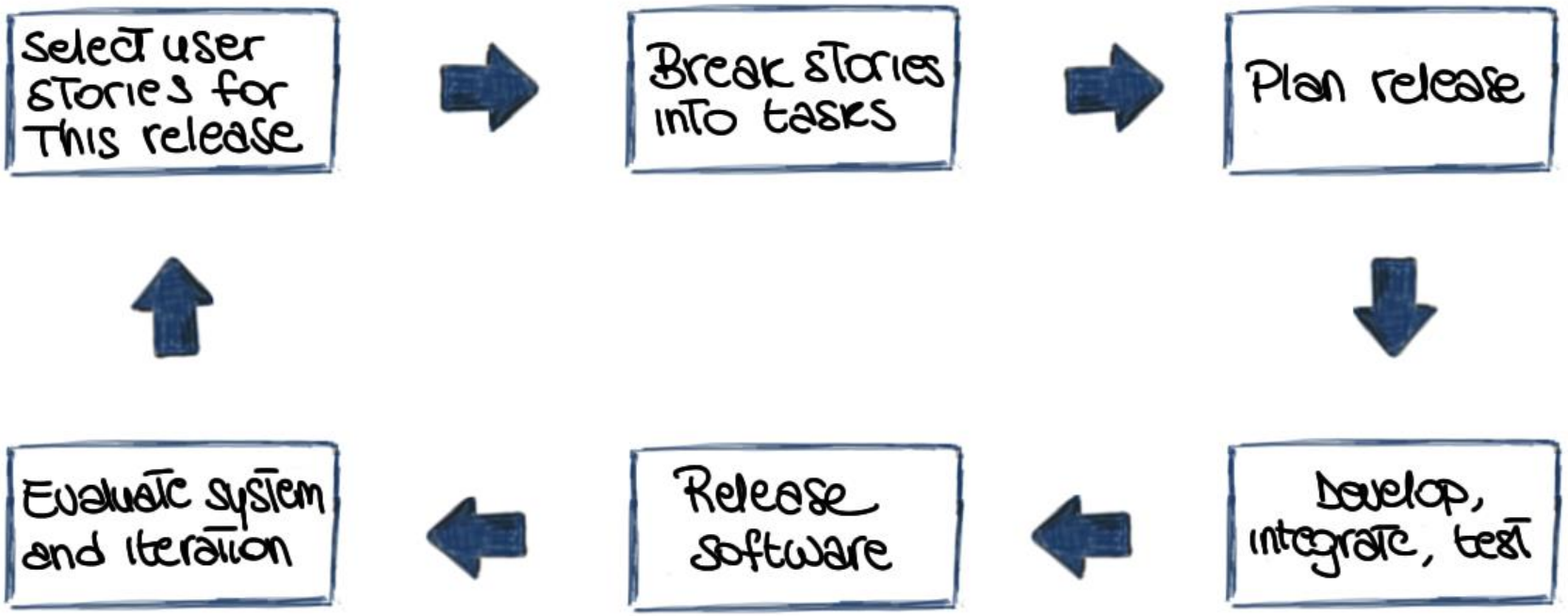
Courage

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

...

INCREMENTAL PLANNING



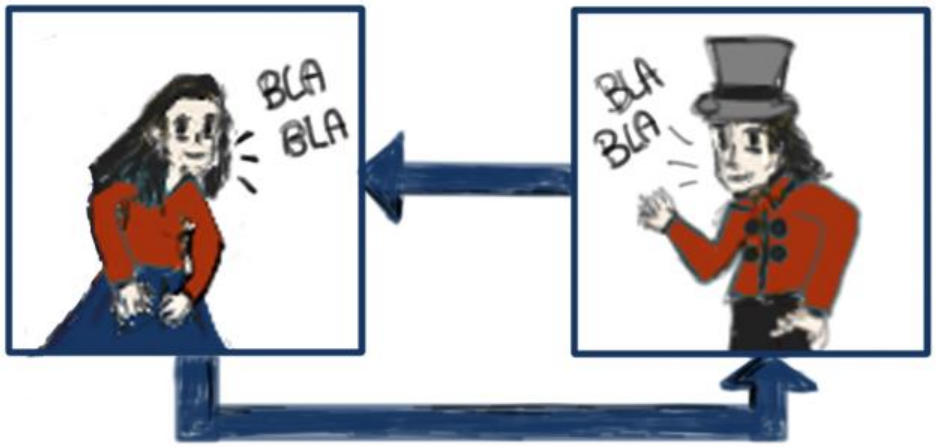
SMALL RELEASES

SMALL RELEASES

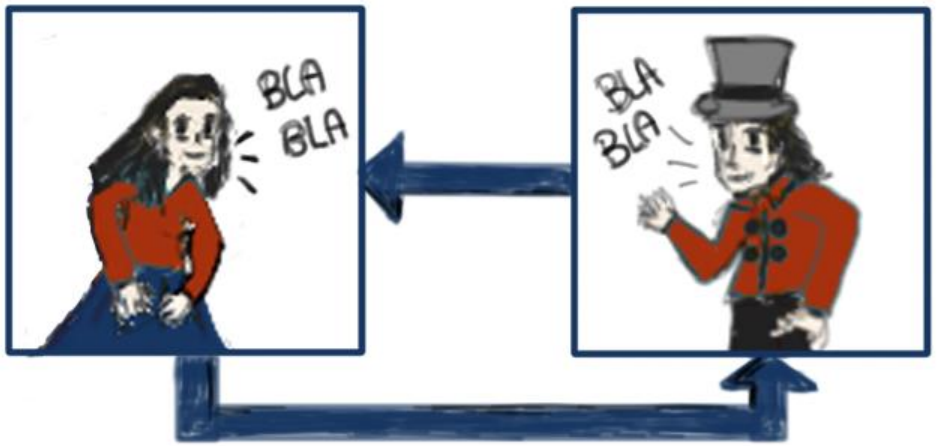
SMALL RELEASES

SMALL RELEASES

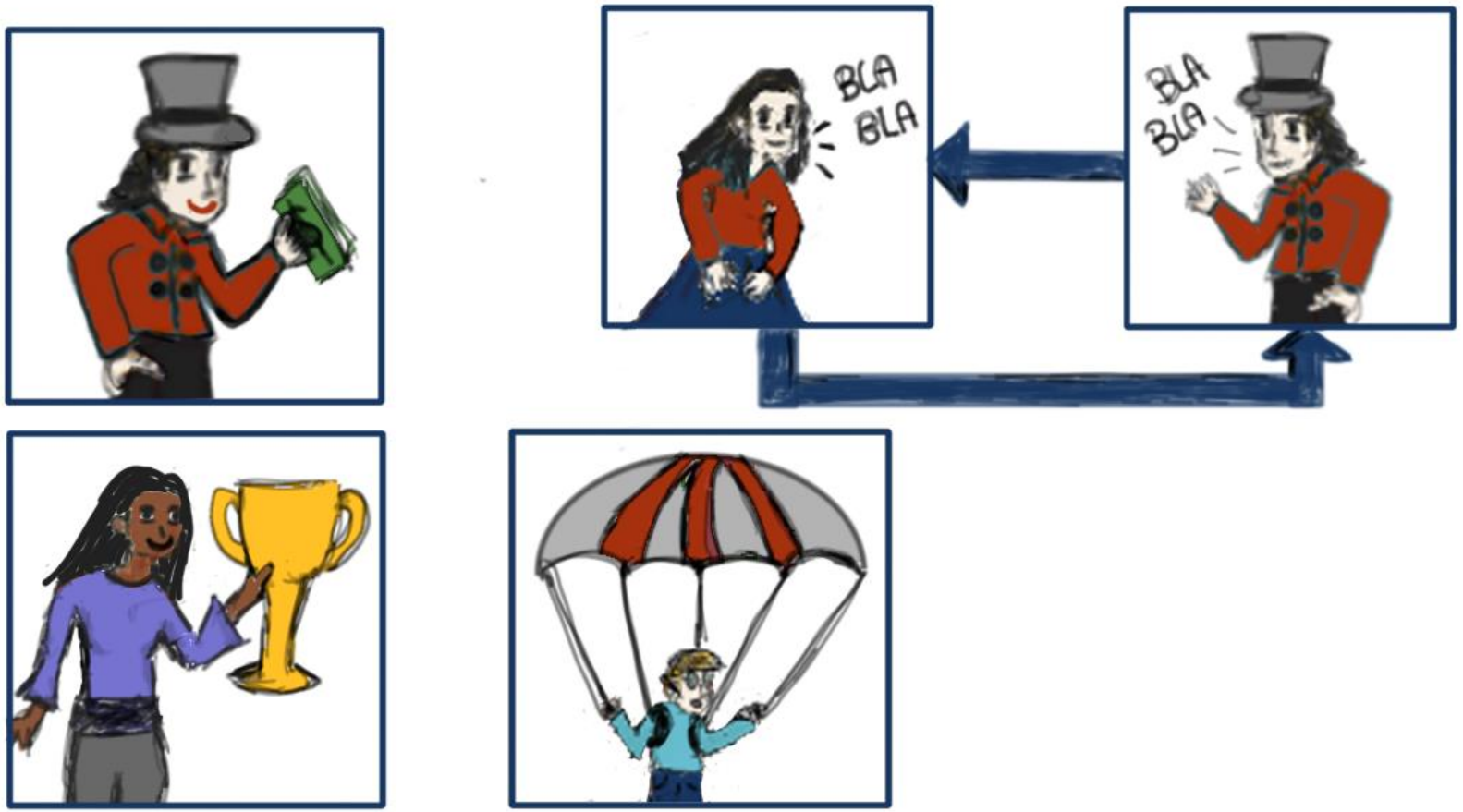
SMALL RELEASES



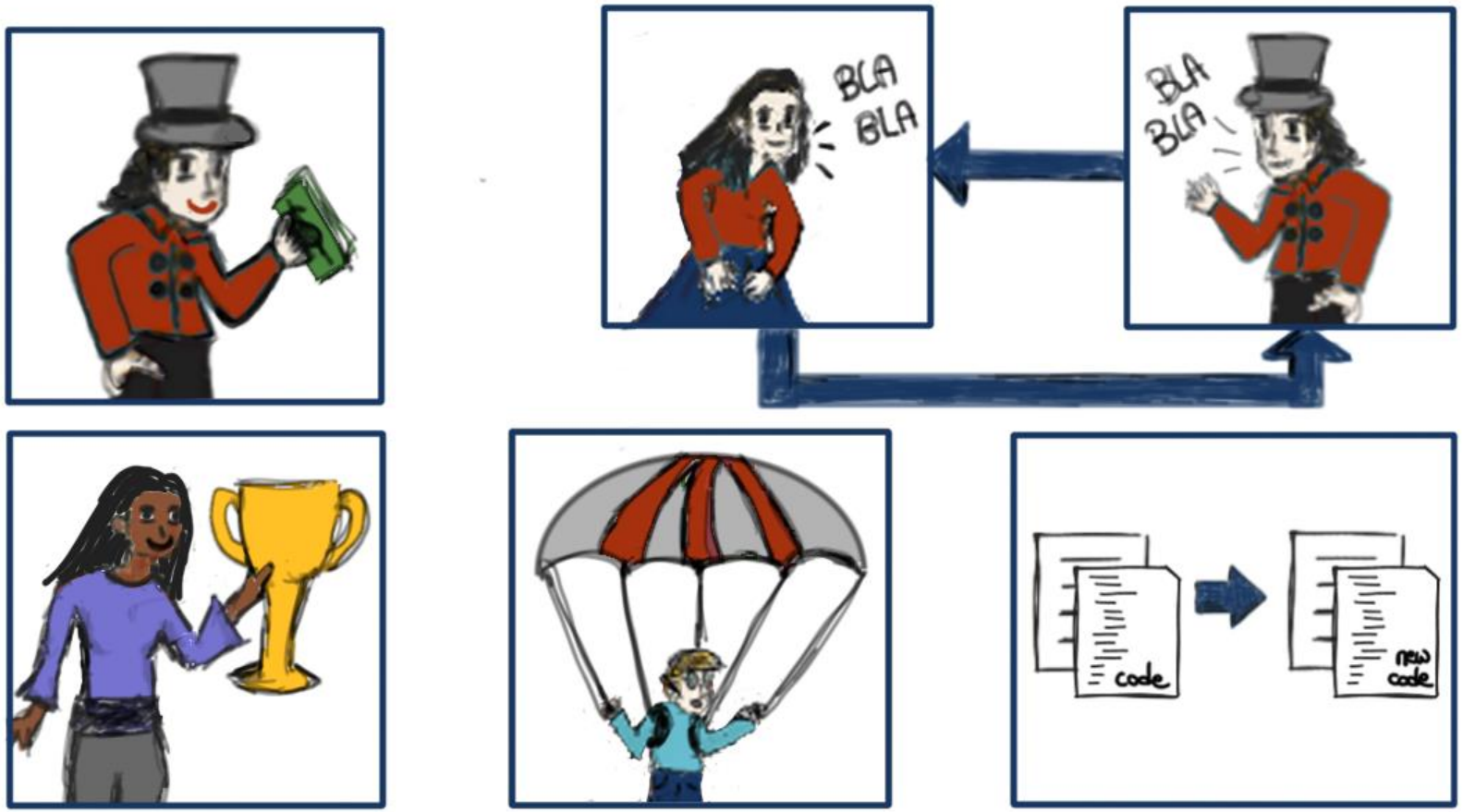
SMALL RELEASES



SMALL RELEASES



SMALL RELEASES



SIMPLE DESIGN

Enough to meet the requirements

No duplicated functionality

Fewest possible classes and methods

SIMPLE DESIGN

Enough to meet the requirements

No duplicated functionality

Fewest possible classes and methods

SIMPLE DESIGN

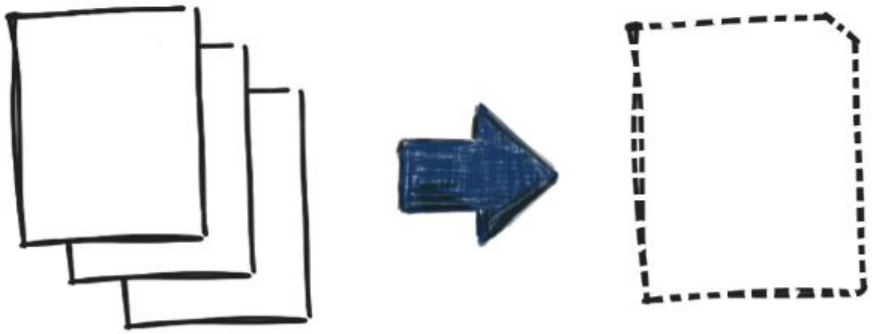
Enough to meet the requirements

No duplicated functionality

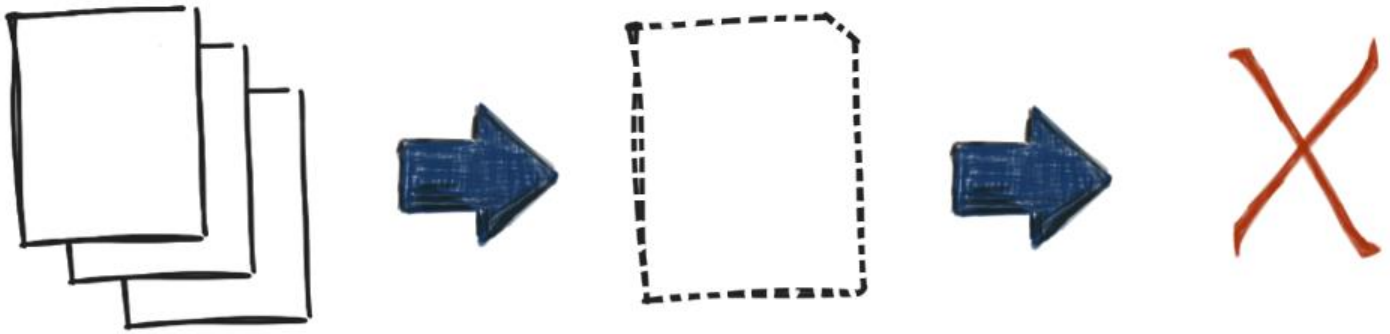
Fewest possible classes and methods

TEST- FIRST DEVELOPMENT

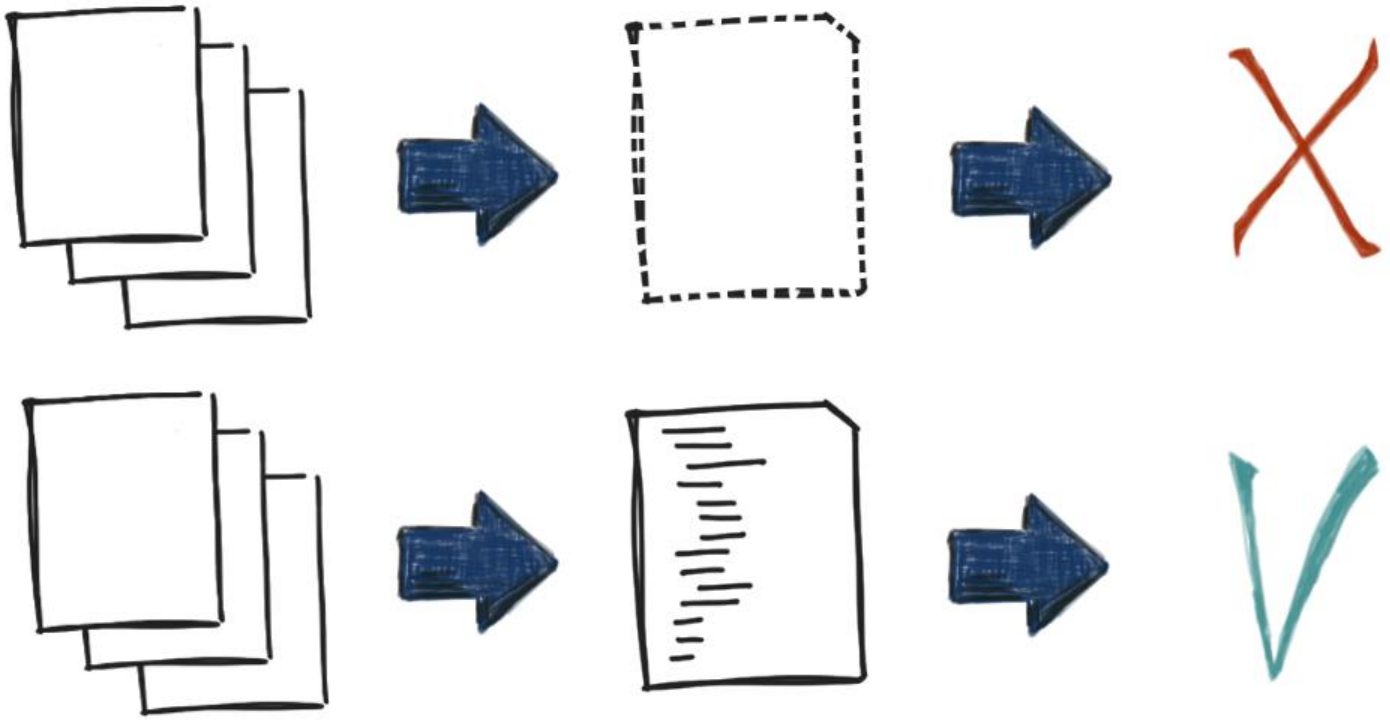
TEST-FIRST DEVELOPMENT



TEST-FIRST DEVELOPMENT



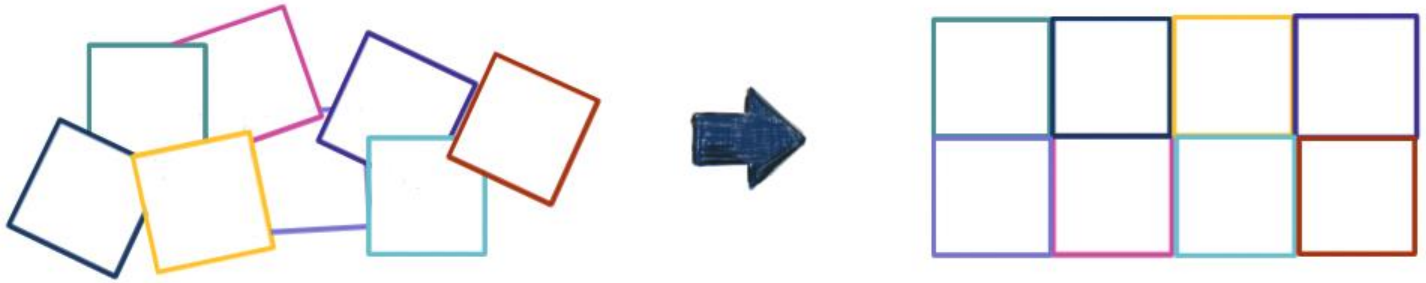
TEST-FIRST DEVELOPMENT



REFACTORING

REFACTORING

REFACTORING



PAIR PROGRAMMING

PAIR PROGRAMMING

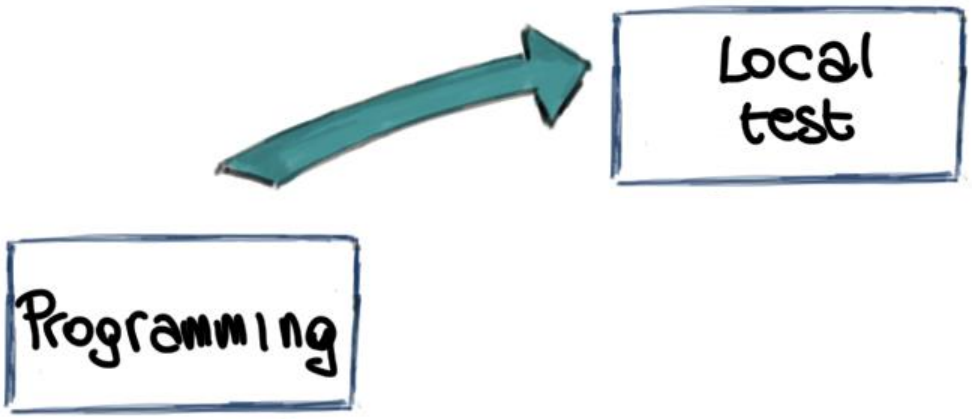
PAIR PROGRAMMING

programming ⇔ strategizing

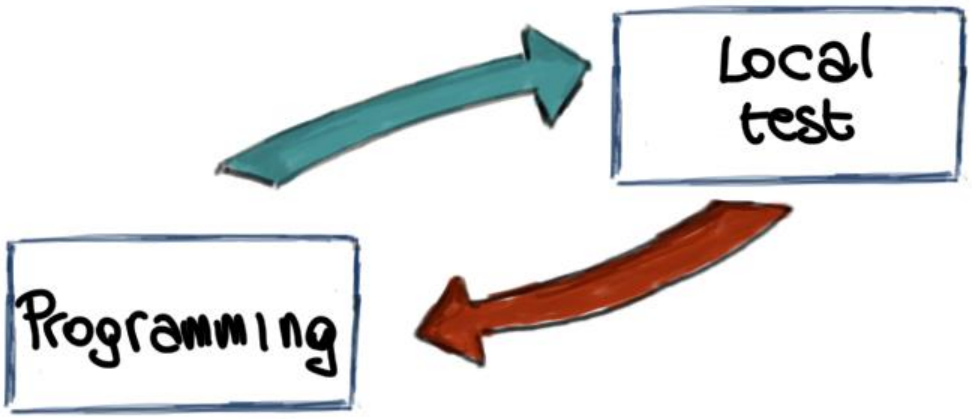
CONTINUOUS INTEGRATION

CONTINUOUS INTEGRATION

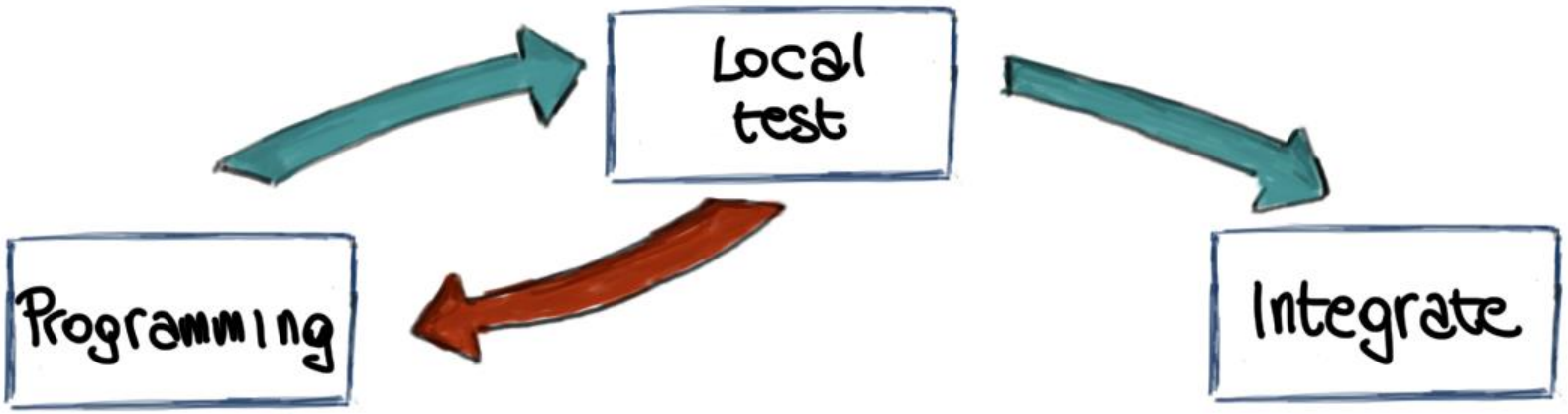
CONTINUOUS INTEGRATION



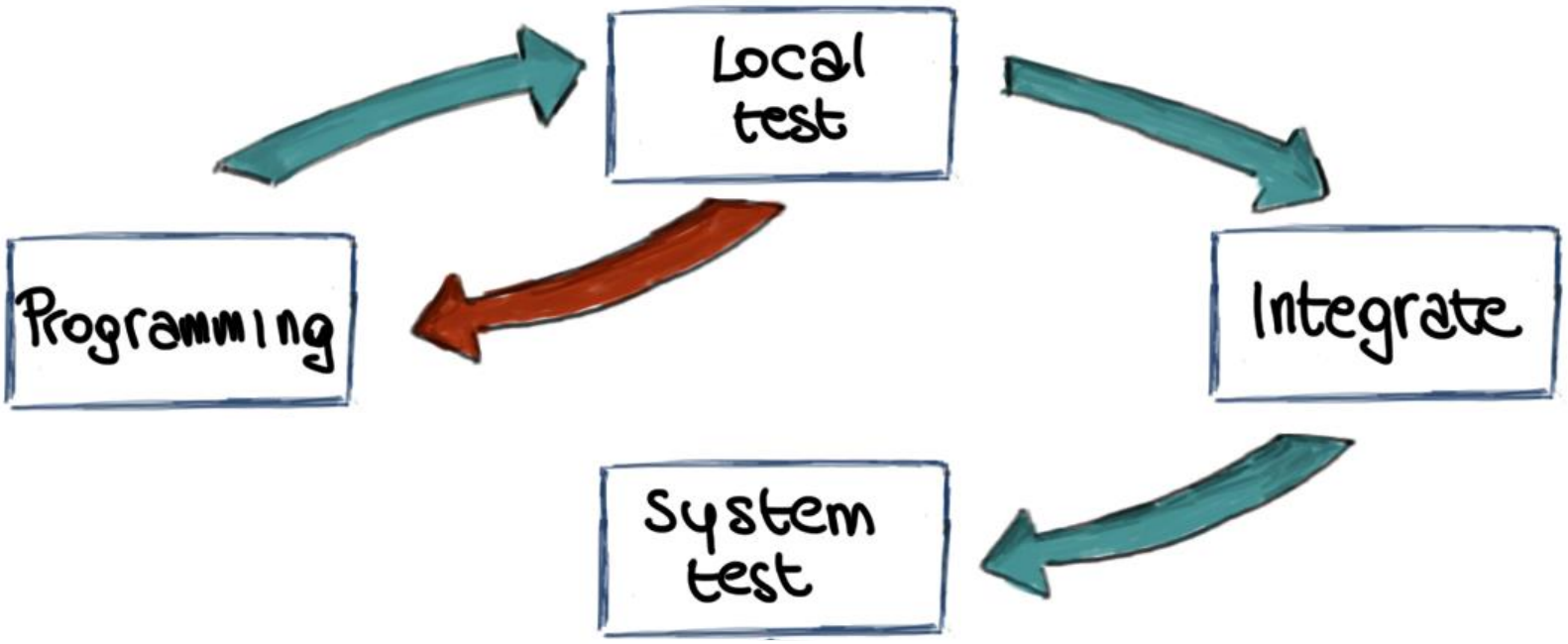
CONTINUOUS INTEGRATION



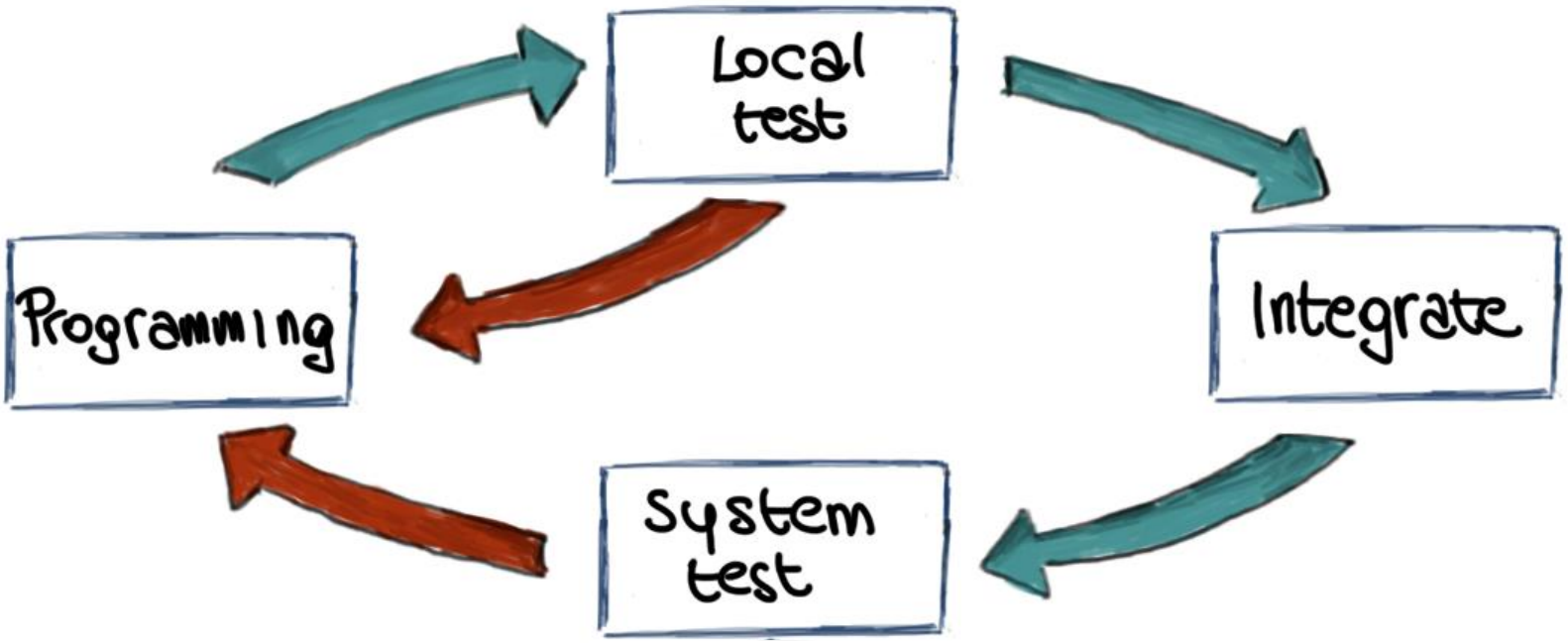
CONTINUOUS INTEGRATION



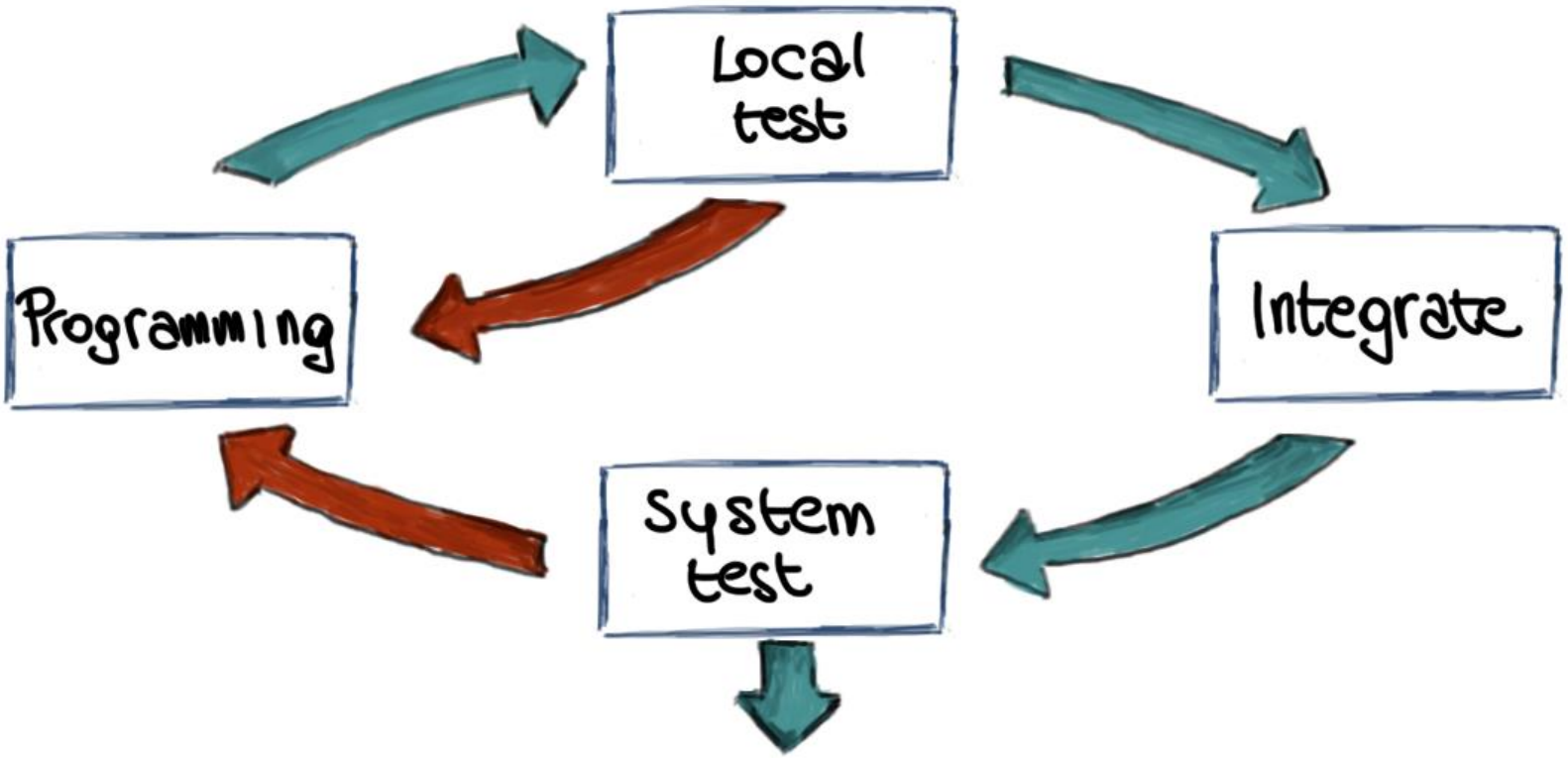
CONTINUOUS INTEGRATION



CONTINUOUS INTEGRATION



CONTINUOUS INTEGRATION



ON-SITE CUSTOMER

The customer is an actual member of the team

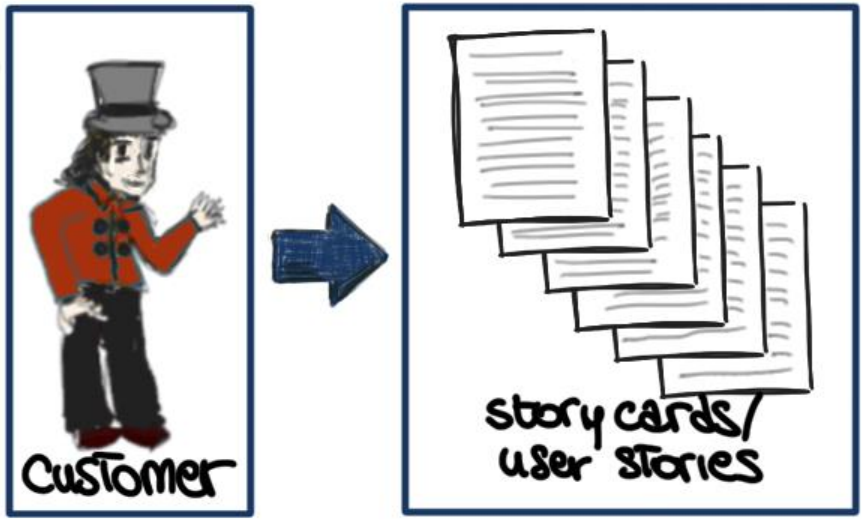
ON-SITE CUSTOMER

The customer is an actual member of the team

- sits with the team
- brings requirements

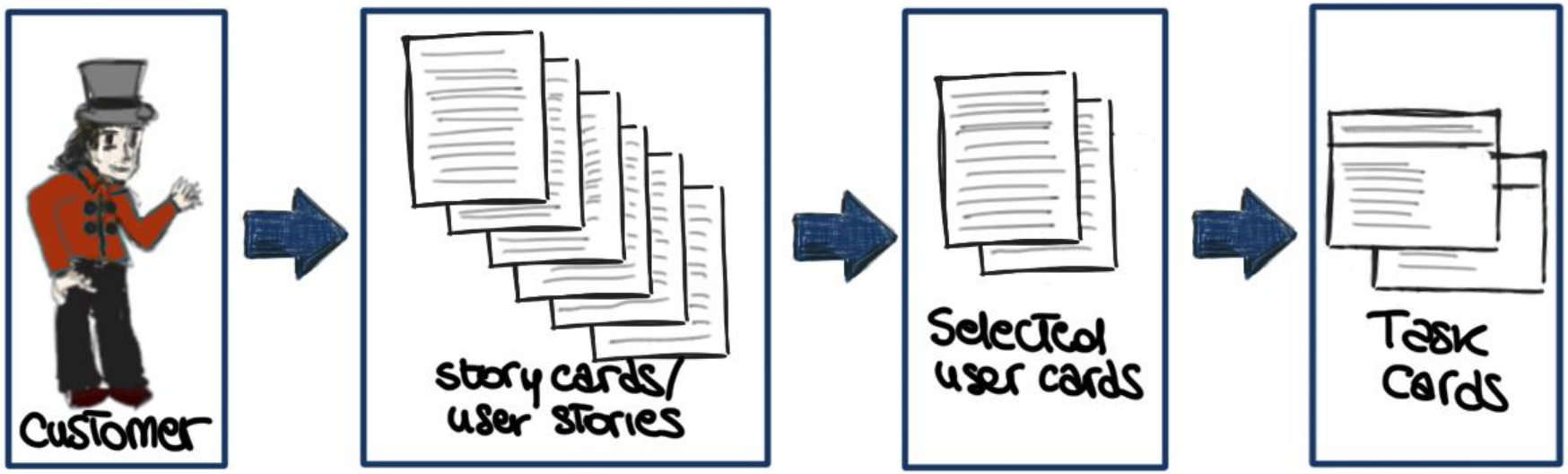
REQUIREMENTS ENGINEERING IN XP

REQUIREMENTS ENGINEERING IN XP

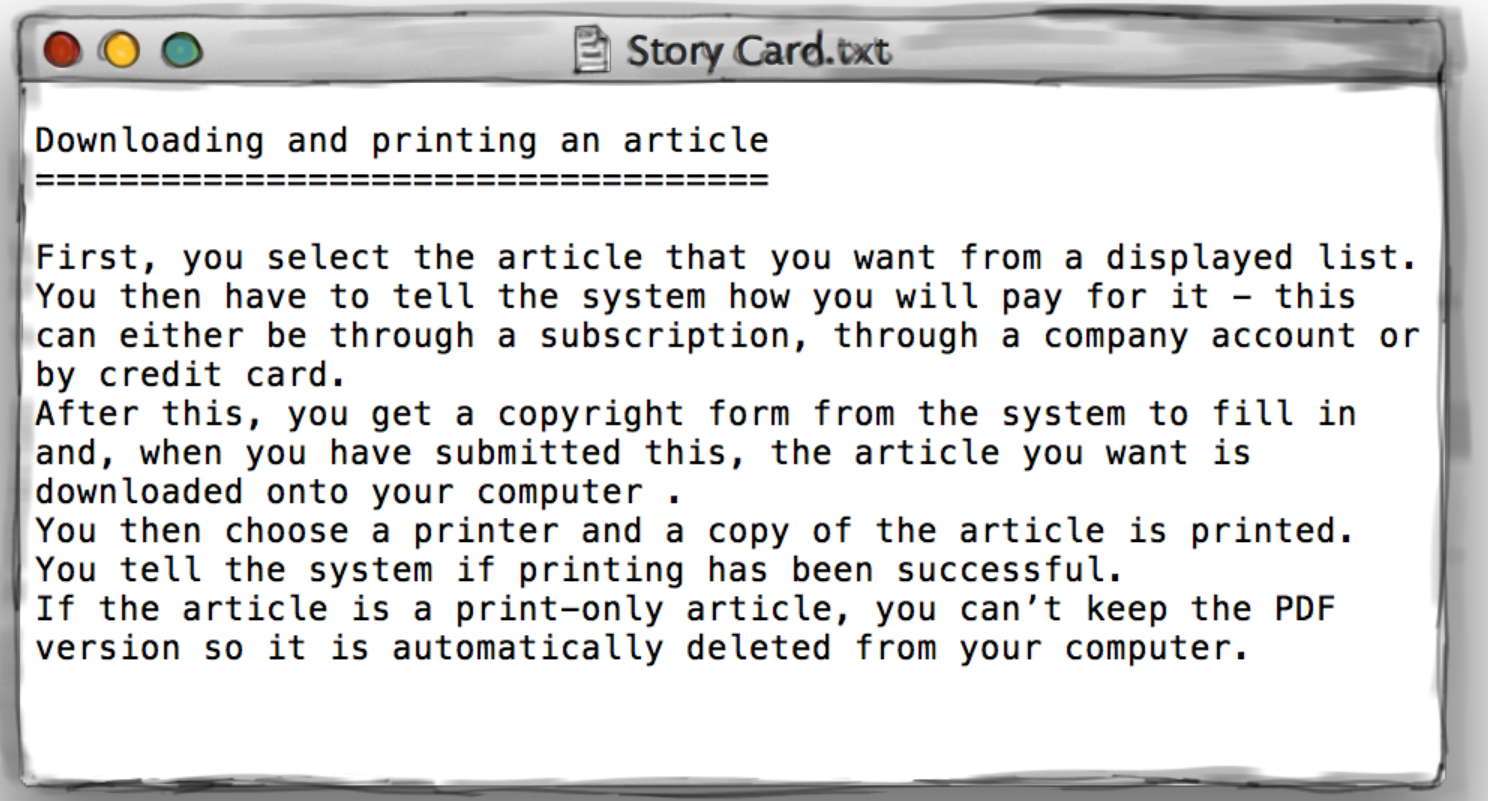


REQUIREMENTS ENGINEERING IN XP

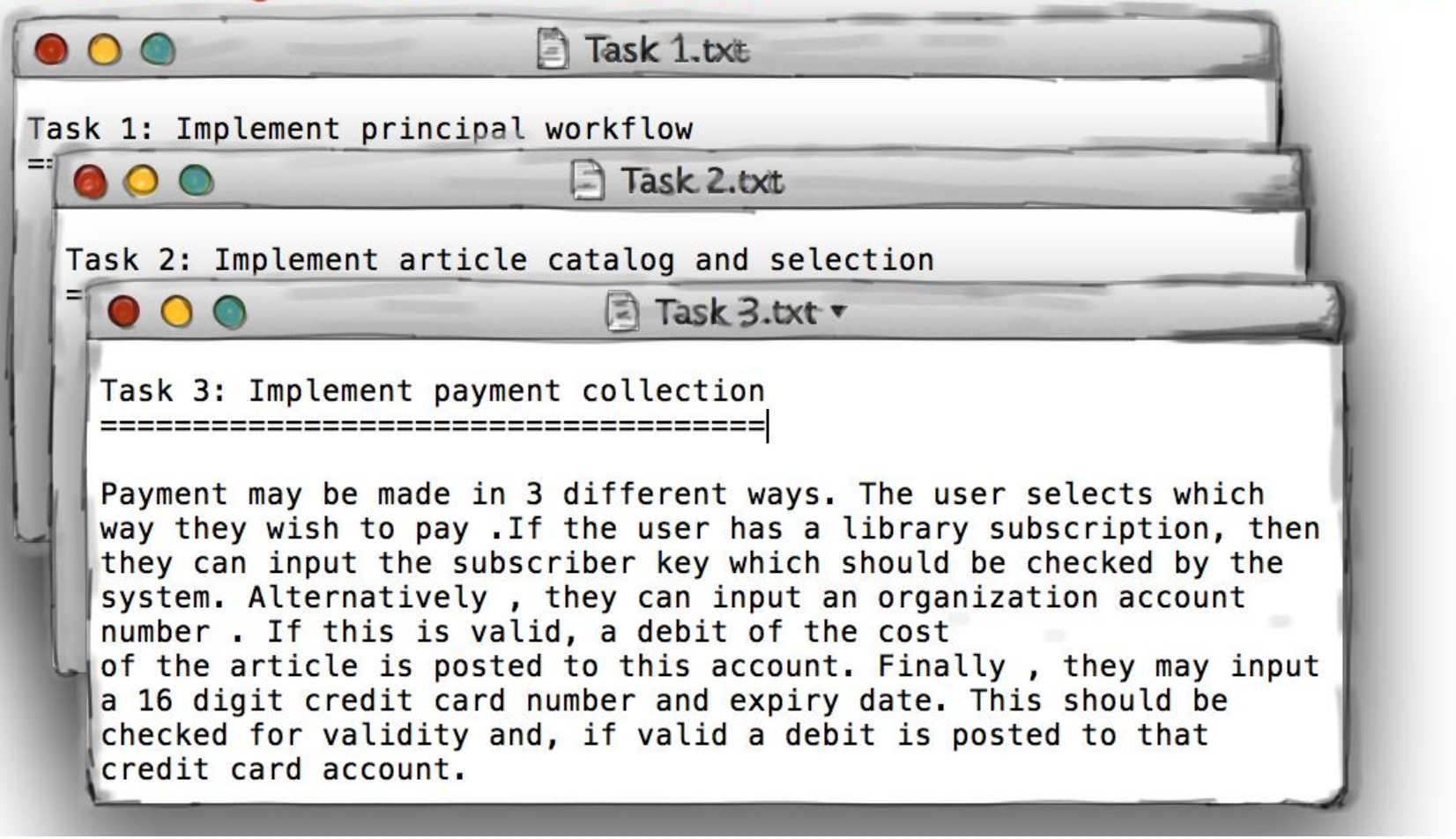
REQUIREMENTS ENGINEERING IN XP



STORY CARD FOR DOCUMENT DOWNLOADING



TASK CARDS FOR DOCUMENT DOWNLOADING



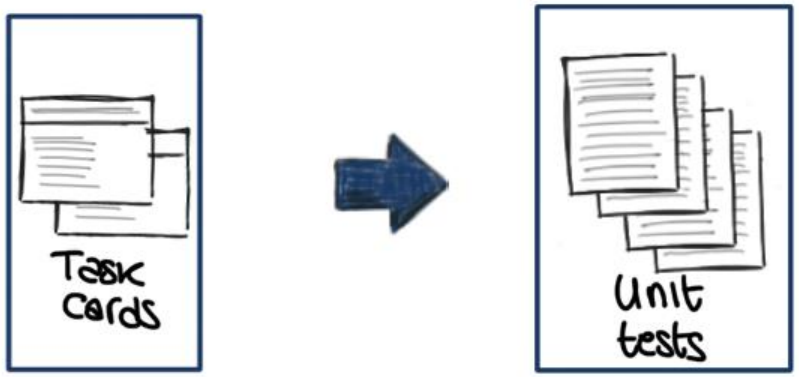
TESTING STRATEGY

TESTING STRATEGY

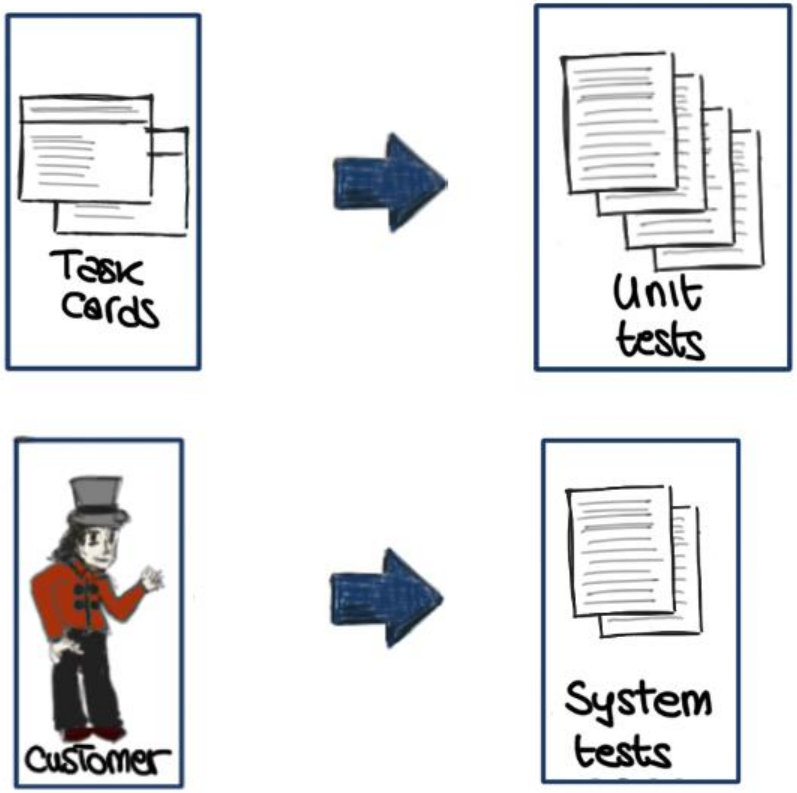
Testing is
coded confidence

TESTING STRATEGY

TESTING STRATEGY



TESTING STRATEGY



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

SCRUM ACTORS

SCRUM ACTORS

Product owner (customer)

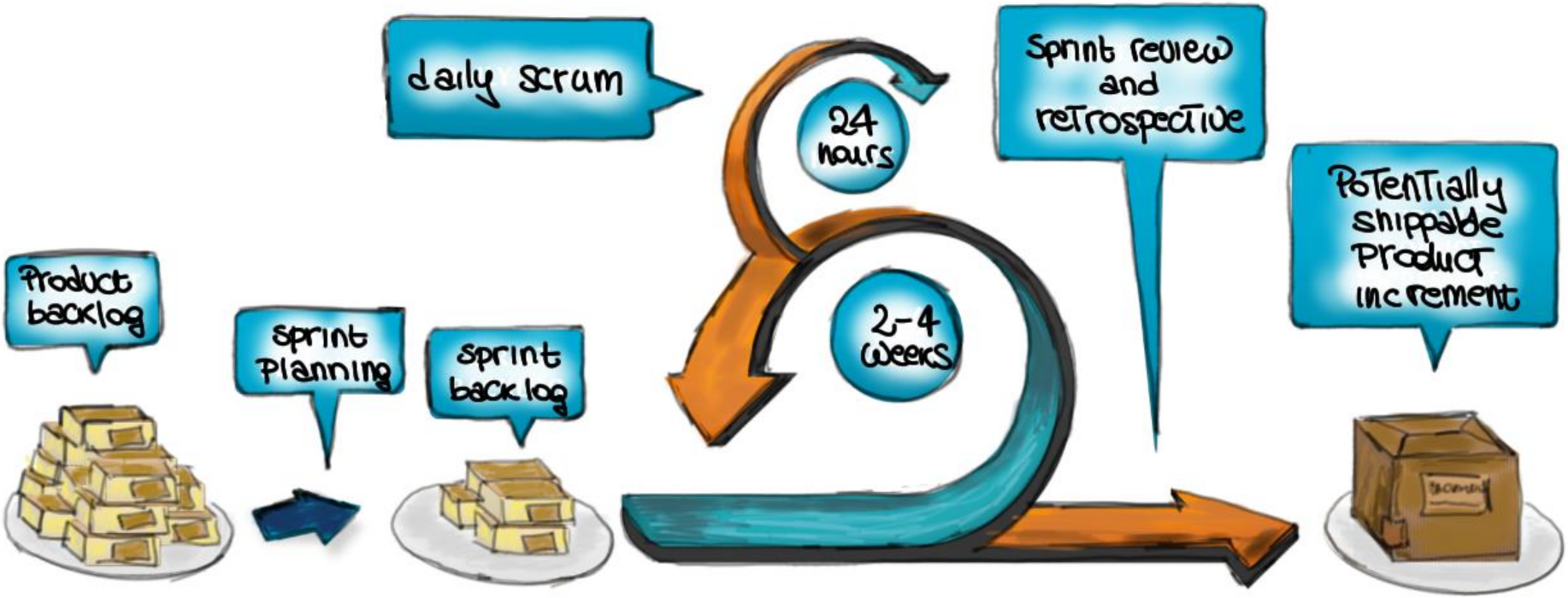
SCRUM ACTORS

Team

Product owner (customer)

Scrum master

HIGH-LEVEL PROCESS



Agile Development Resources

- WWW
  - www.extremeprogramming.org
  - www.refactoring.com
  - www.XProgramming.com
  - www.scrum.org
- Books
  - *Many* on agile software development
  - Refactoring: Improving the Design of Existing Code (Addison-Wesley)