THE POWER OF FEEDBACK LOOPS

Hello!

I am Luca Mezzalira

Solutions Architect at Massive Interactive

Co-Organiser of Agile 101 and Agile Leadership Communities in London

Agenda

- What is the feedback loop ?
- Empirical vs Scientific approach
- How can I integrate the feedback loop in my software development flow ?
- Collect data to analyse

1.

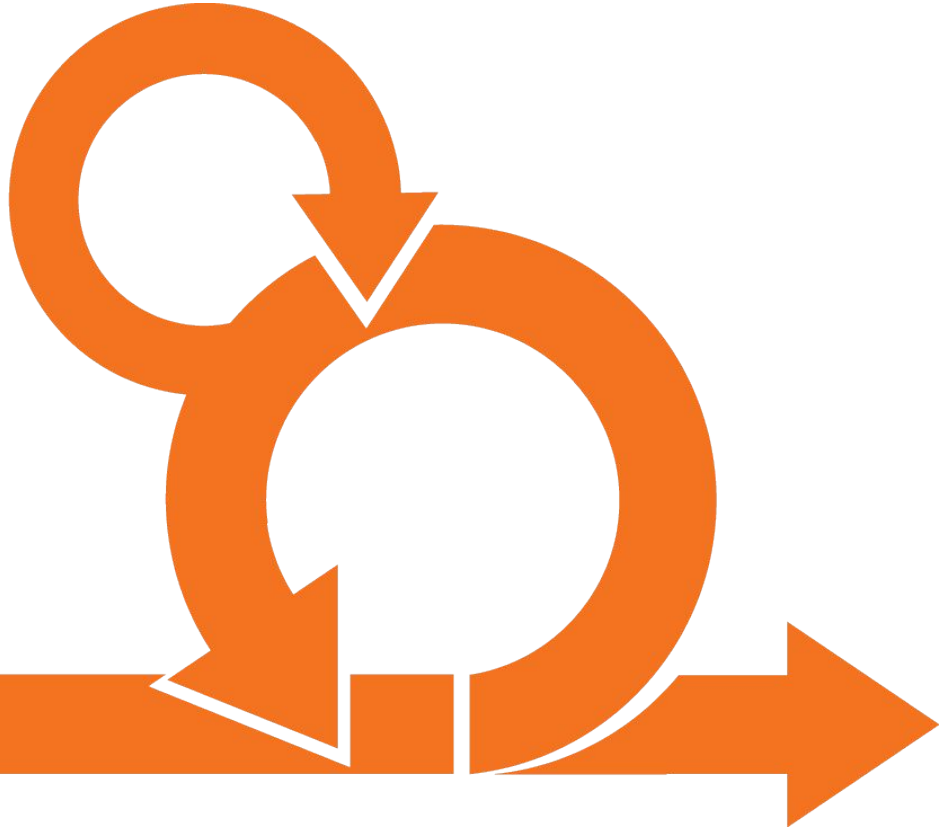
The FEEDBACK LOOP

*The section of a **control system** that allows for feedback and **self-correction** and that **adjusts its operation** according to differences between the actual and the desired or **optimal output***

http://www.thefreedictionary.com/feedback+loop

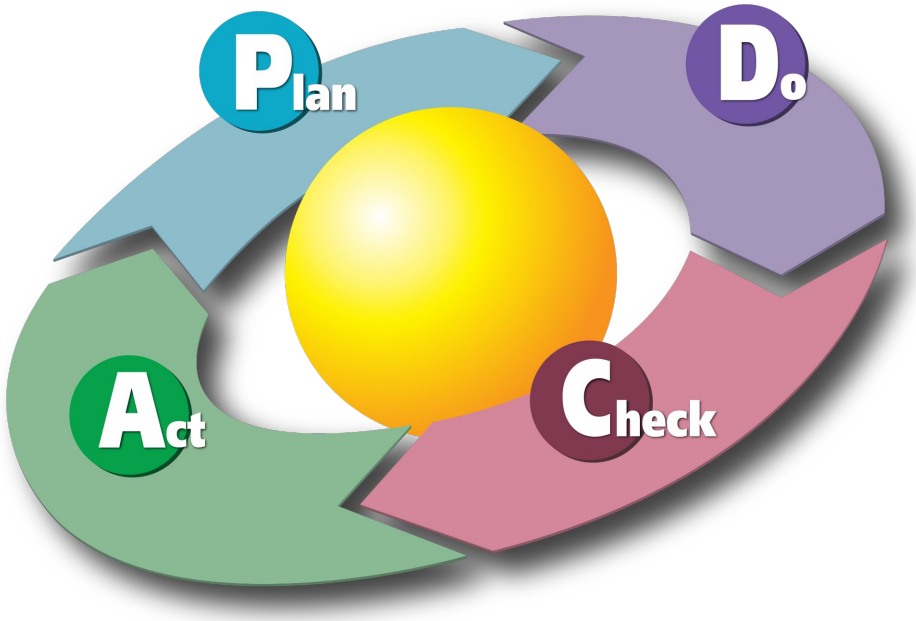
Feedback loops *provide generic mechanisms for* ***controlling the running, maintenance, and evolution of software*** *and computing systems.*

https://en.wikipedia.org/wiki/Feedback



Feedback Loop in Scrum

- Sprint
  - Daily standup
  - Sprint Retrospective
  - Sprint Review

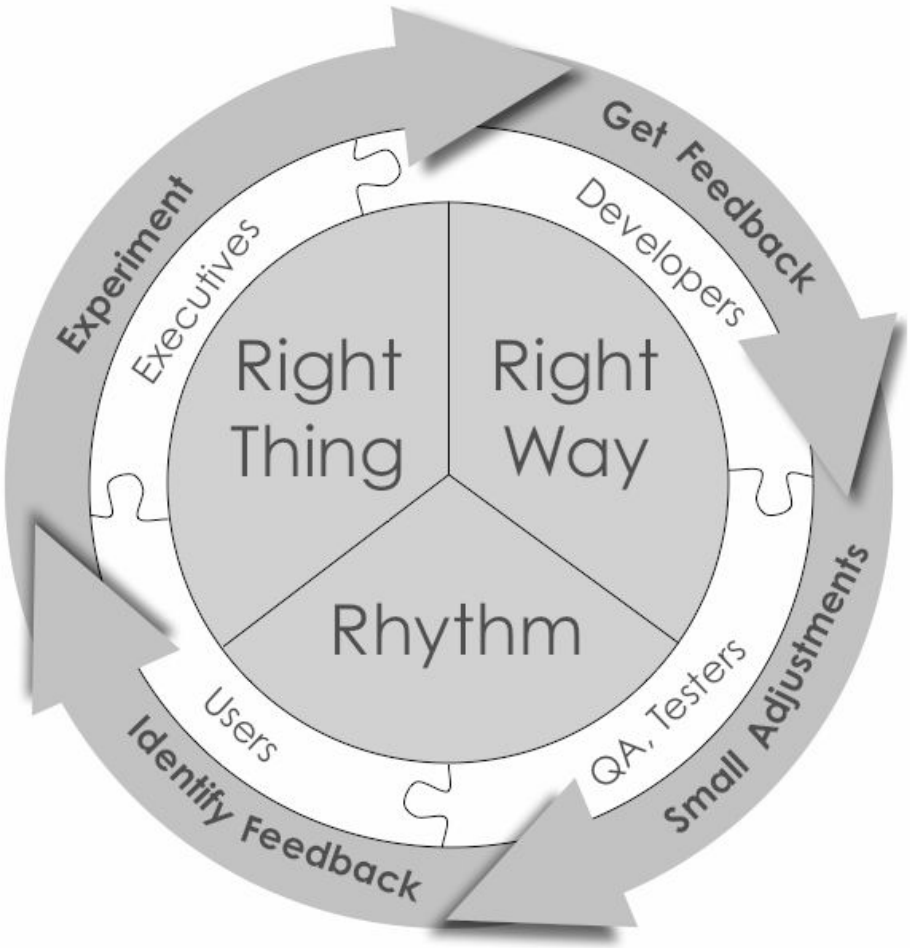


PDCA Cycle: the loop behind the continuous improvement

- Plan
- Do
- Check
- Act

Feedback Loops Characteristics in Grows Method

- real time
- direct
- short
- small
- concrete
- mesurable



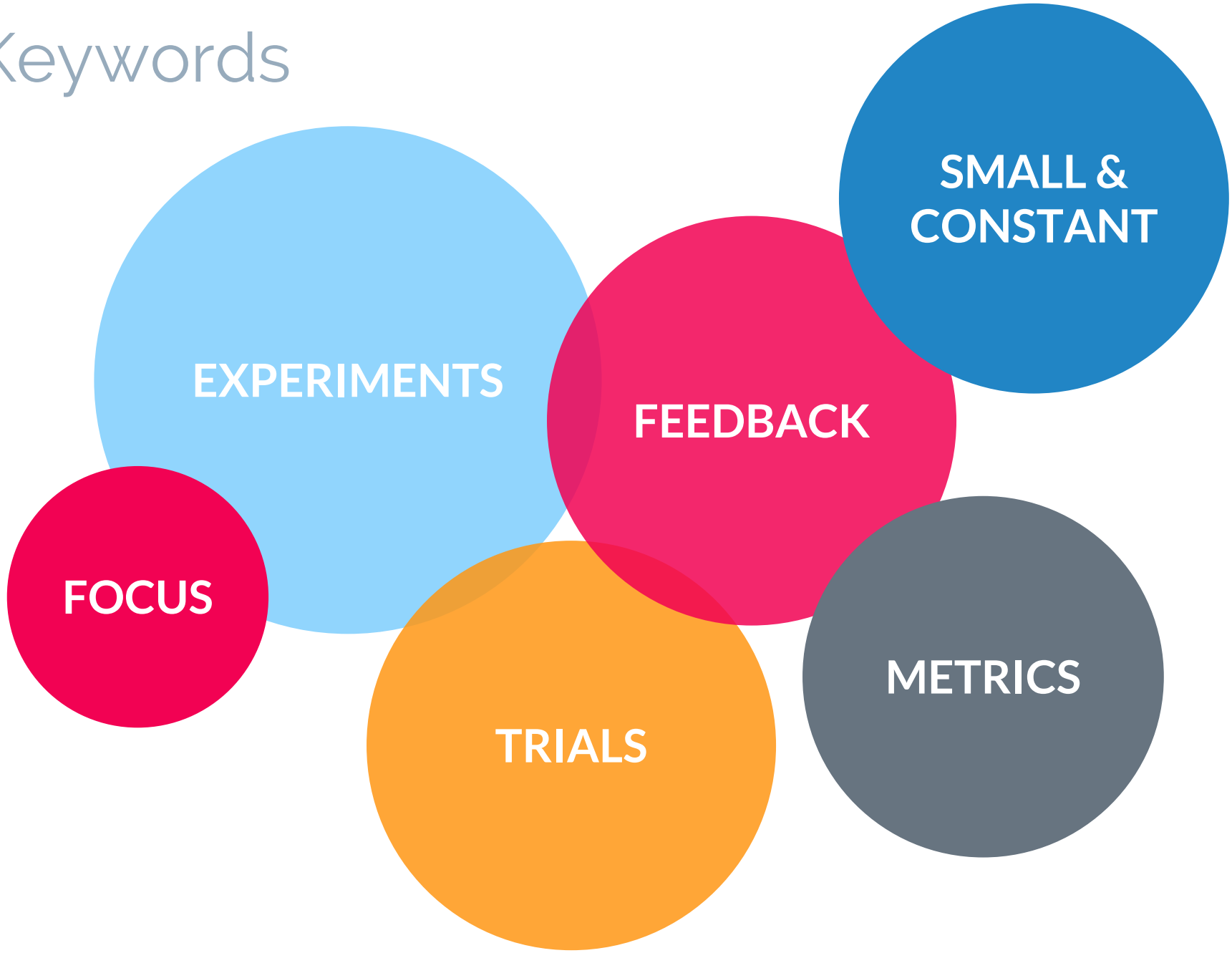
Why is feedback loop so important in software development ?

2.

EMPIRICAL vs SCIENTIFIC


H2O

Keywords


SMALL & CONSTANT
EXPERIMENTS
FEEDBACK
FOCUS
METRICS
TRIALS

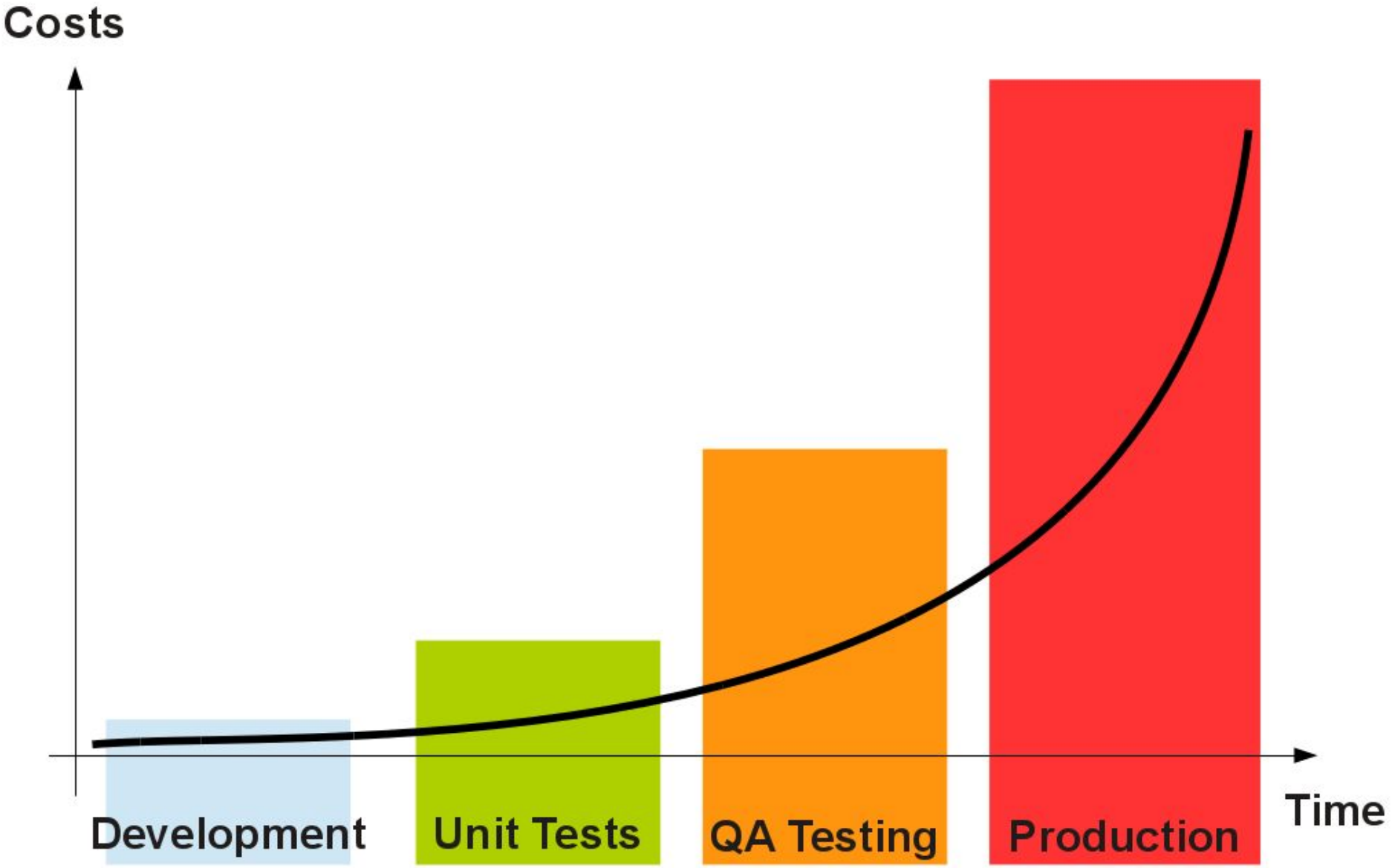
Let's create a chain of loops!

- Automation Testing*
- Continuous Integration* — secs
- Pair Programming*
- Code Reviews* — mins
- Static Analysis*
- QA testing — hours
- Emergent Design*
- Standup — days
- Sprint Review
- Sprint Retrospective — weeks

MAIN GOALS

- CODE QUALITY
- CODE STANDARDS
- CATCH BUGS EARLY
- BETTER PERFORMANCES
- MAINTAINABILITY
- MEET THE REQUIREMENTS
- CREATE VALUE
- SHARE KNOWLEDGE WITHIN THE TEAM
- CONTINUOUS IMPROVEMENT

Remember remember...

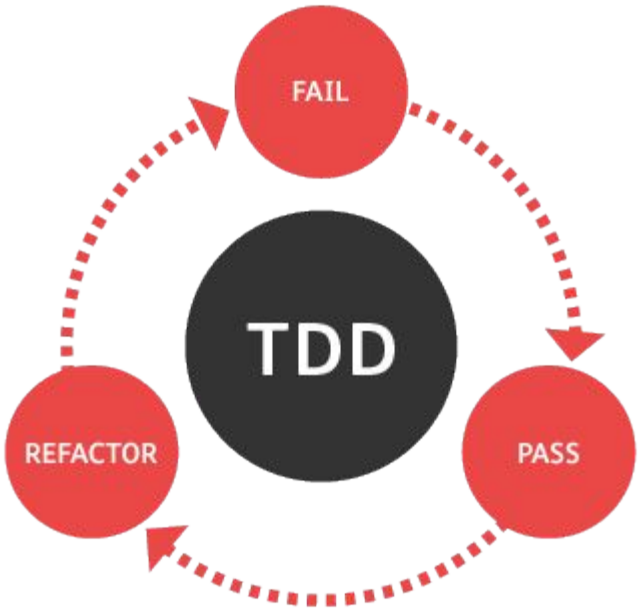


3.
INTEGRATE feedback loops inside your FLOW

Automation Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test.

Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation.



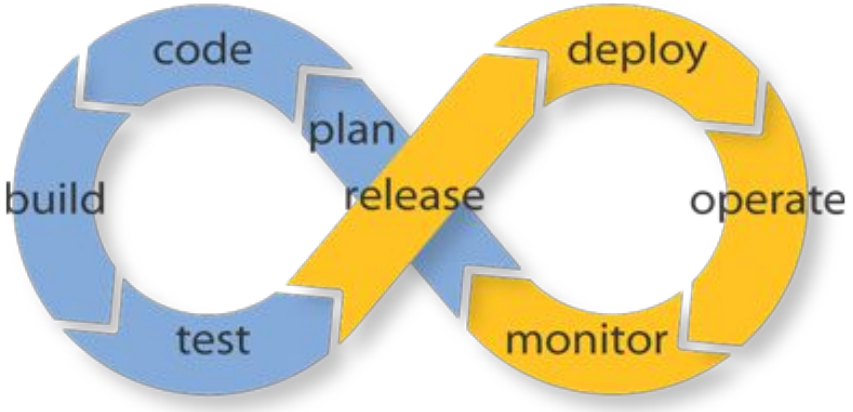
Feedback frequency: **SECONDS**

Goals: **CATCH BUGS EARLIER, CODE QUALITY & MAINTAINABILITY**

https://en.wikipedia.org/wiki/Software_testing

Continuous Integration

Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.



By integrating regularly, you can detect errors quickly, and locate them more easily.
Feedback frequency: **SECONDS**

Goals: **RELIABLE CODE, CREATE VALUE, CODE QUALITY & STANDARDS, CATCH BUGS EARLIER**

https://www.thoughtworks.com/continuous-integration

Pair Programming

Pair programming (sometimes referred to as peer programming) is an agile software development technique in which two programmers work as a pair together on one workstation.

One, the driver, writes code while the other, the observer, pointer or navigator, reviews each line of code as it is typed in.

The two programmers switch roles frequently.

Feedback frequency: **SECONDS**

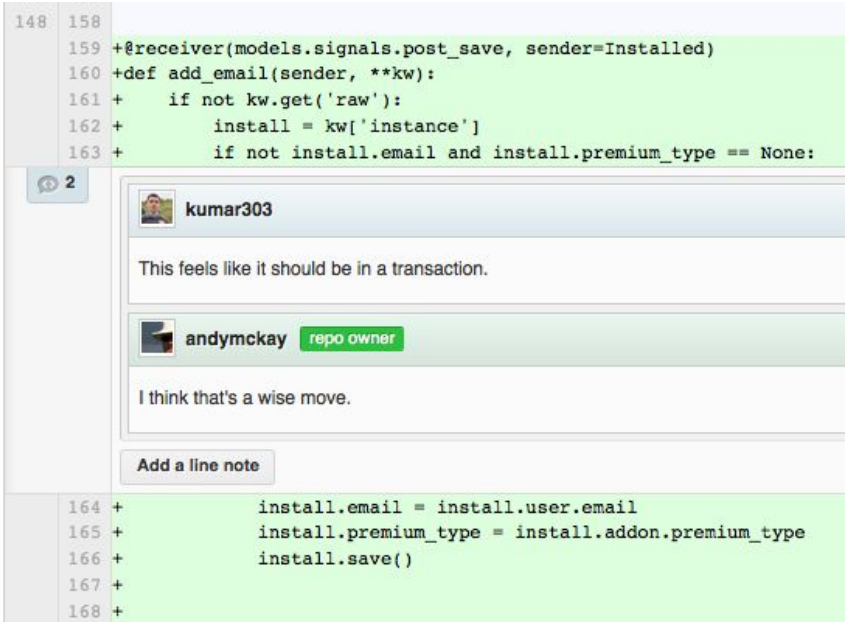
Goals: **SHARED KNOWLEDGE & BETTER CODE QUALITY**

https://en.wikipedia.org/wiki/Pair_programming

Code Reviews

Code review is systematic examination (often known as peer review) of computer source code.

It is intended to find and fix mistakes overlooked in the initial development phase, improving both the overall quality of software and the developers' skills



Feedback frequency: **MINUTES**

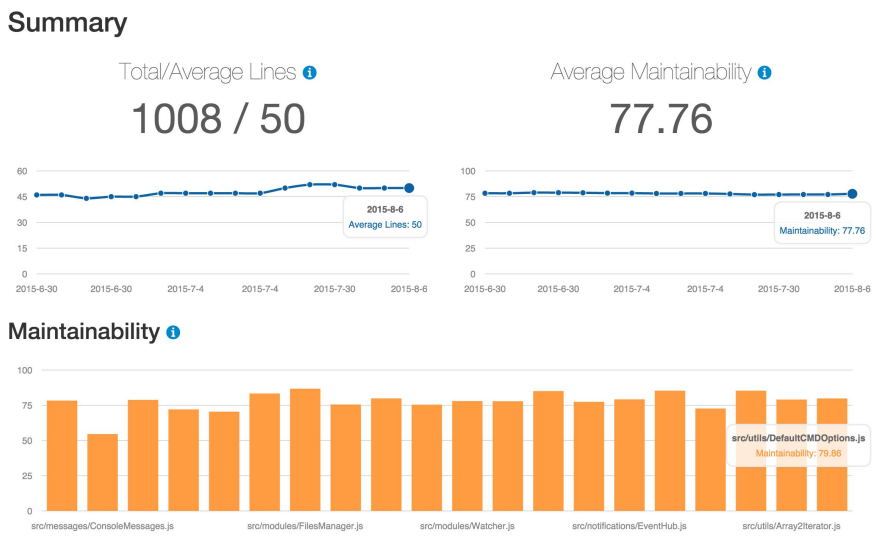
Goals: **BETTER CODE QUALITY, PERFORMANCE IMPROVEMENTS & SHARED KNOWLEDGE**

https://en.wikipedia.org/wiki/Code_review

Static Analysis

Static program analysis is the analysis of computer software that is performed without actually executing programs (analysis performed on executing programs is known as dynamic analysis).

In most cases the analysis is performed on some version of the source code, and in the other cases, some form of the object code.



Feedback frequency: **MINUTES**

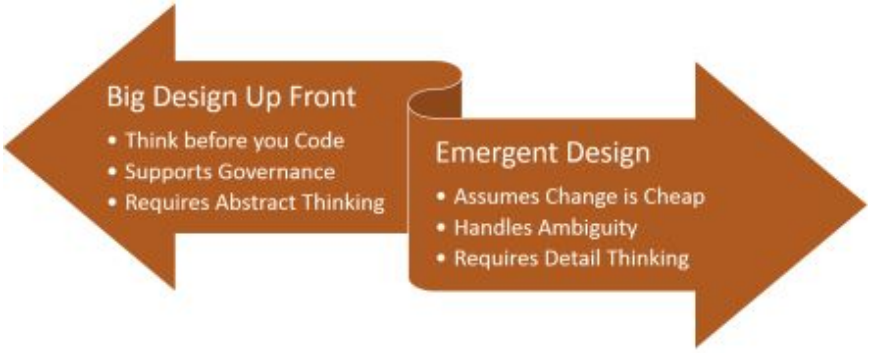
Goals: **BETTER CODE QUALITY, CODE STYLE COMPLIANT & PERFORMANCE IMPROVEMENT**

https://en.wikipedia.org/wiki/Code_review

Emergent Design

Emergent design is a consistent topic in agile software development, as a result of the methodology's focus on delivering small pieces of working code with business value.

The end result is a simpler design with a smaller code base, which is more easily understood and maintained and naturally has less room for defects

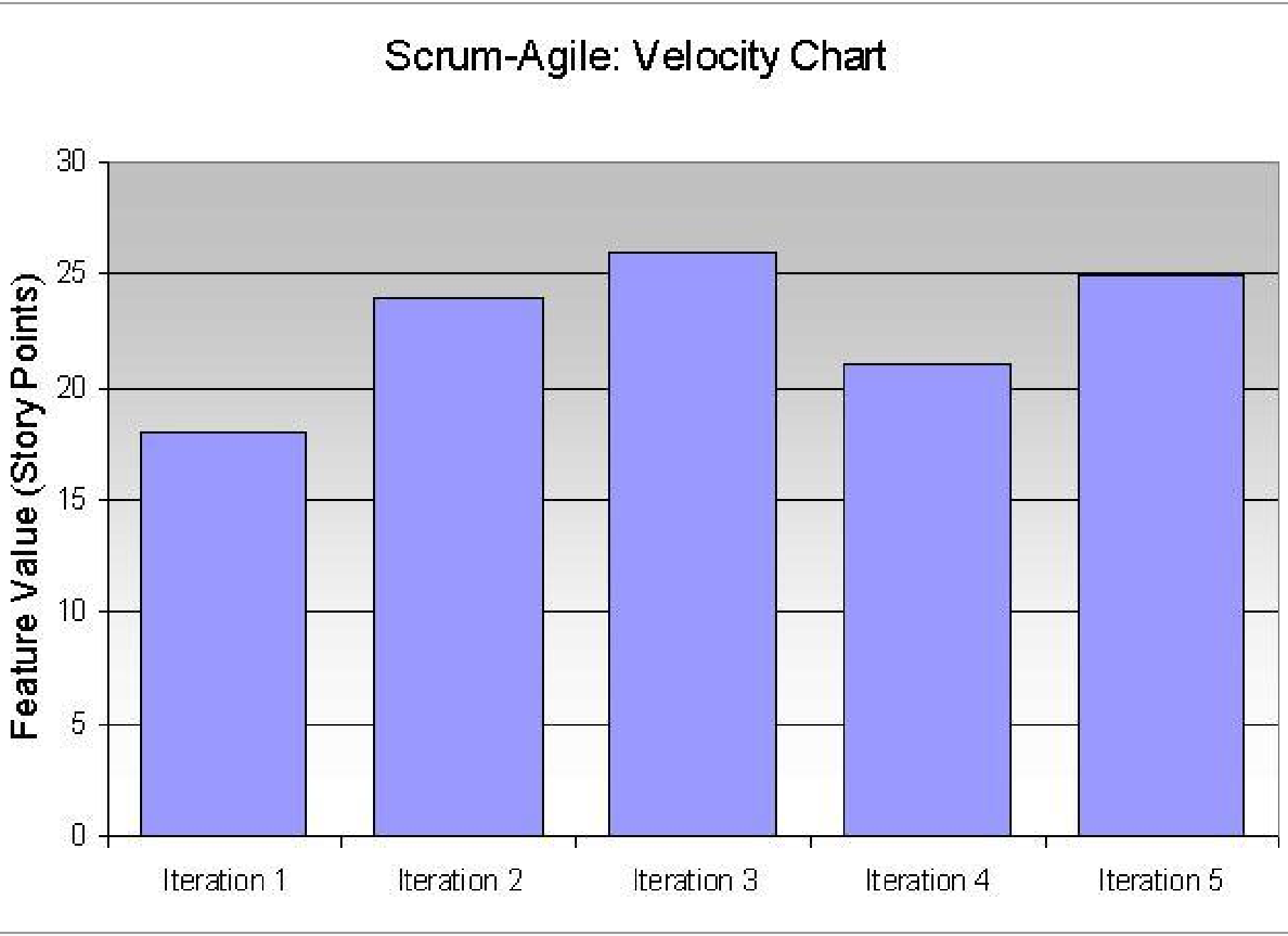


Feedback frequency: **HOURS**

Goals: **BETTER CODE QUALITY, HANDLE NEW REQUIREMENTS & MAINTAINABILITY**

https://en.wikipedia.org/wiki/Emergent_Design

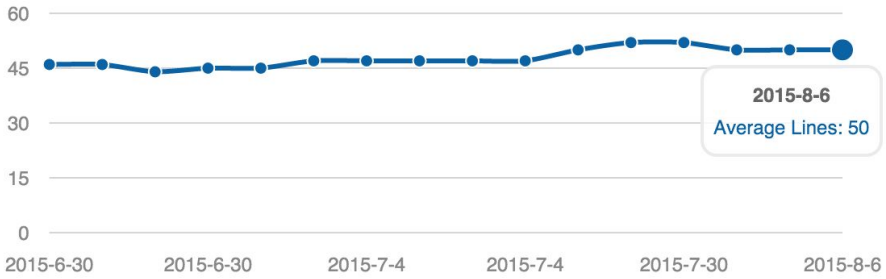
4.
Collect data to analyse


Scrum-Agile: Velocity Chart
Feature Value (Story Points)
30
25
20
15
10
5
0
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5

Summary

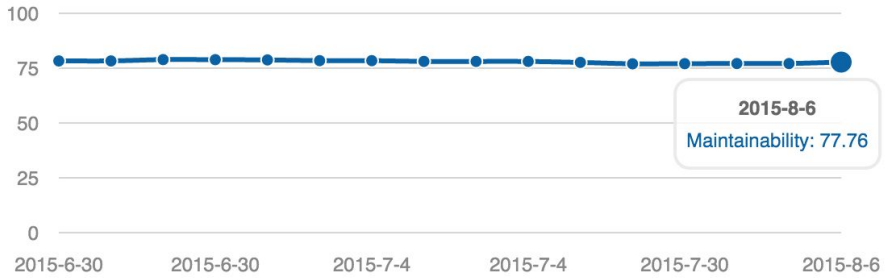
Total/Average Lines

1008 / 50

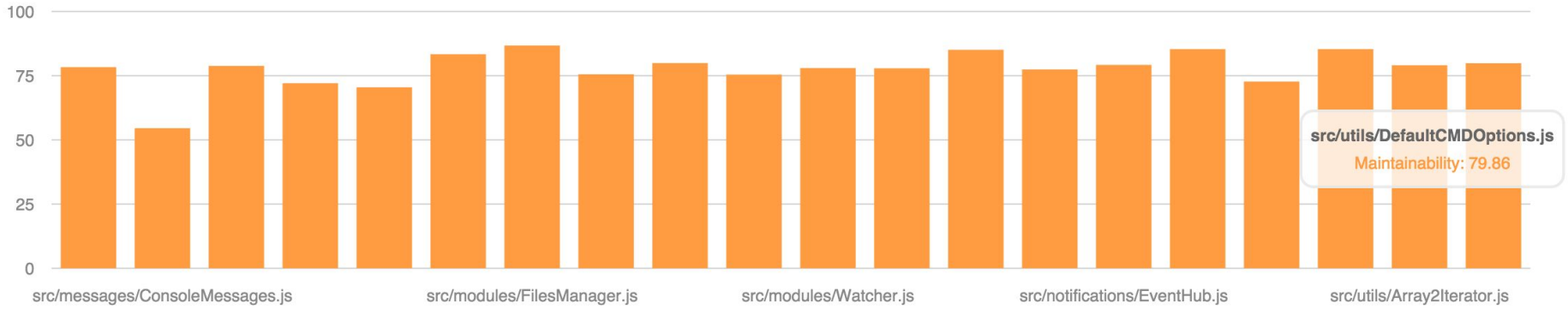


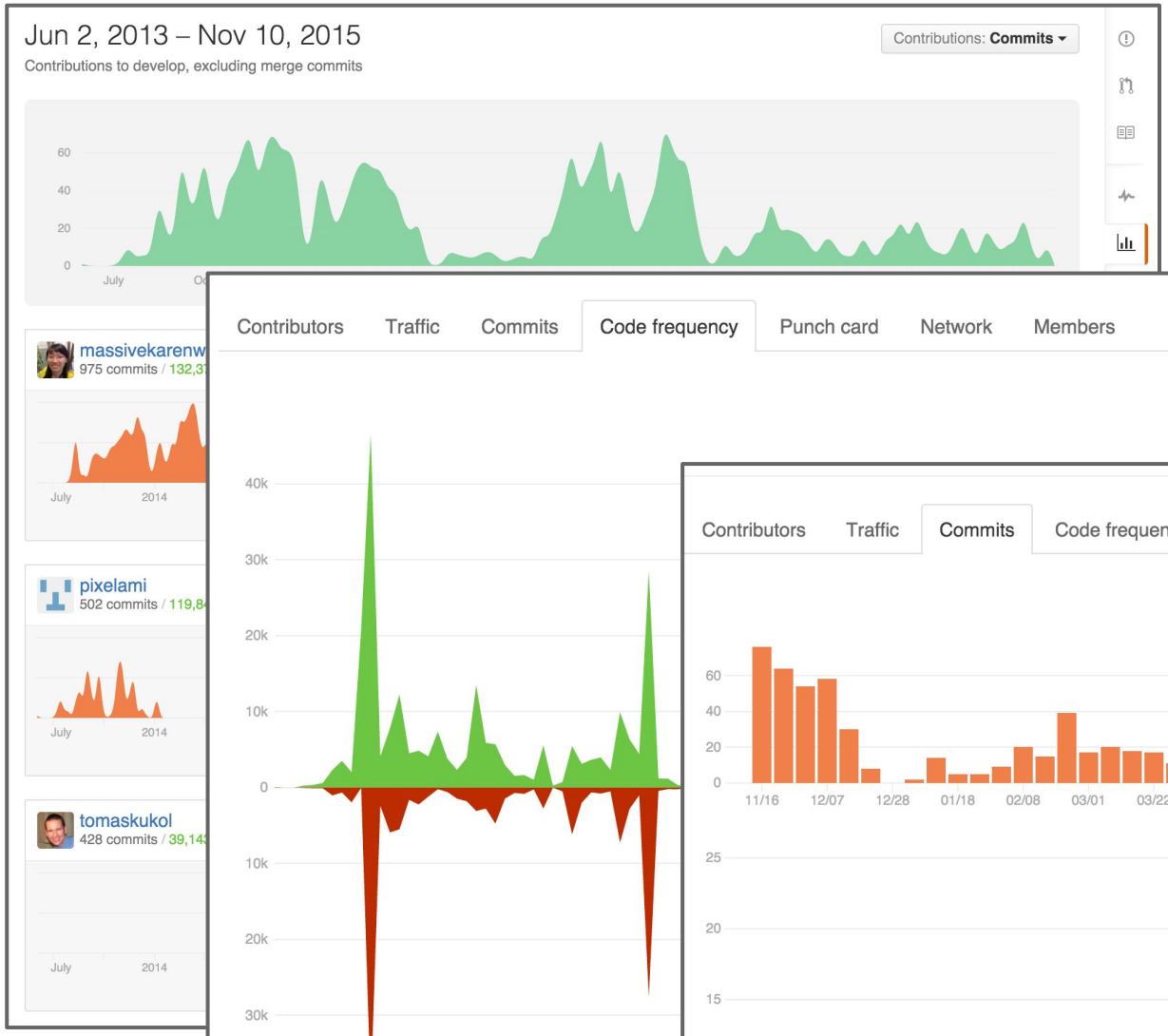
Average Maintainability

77.76



Maintainability

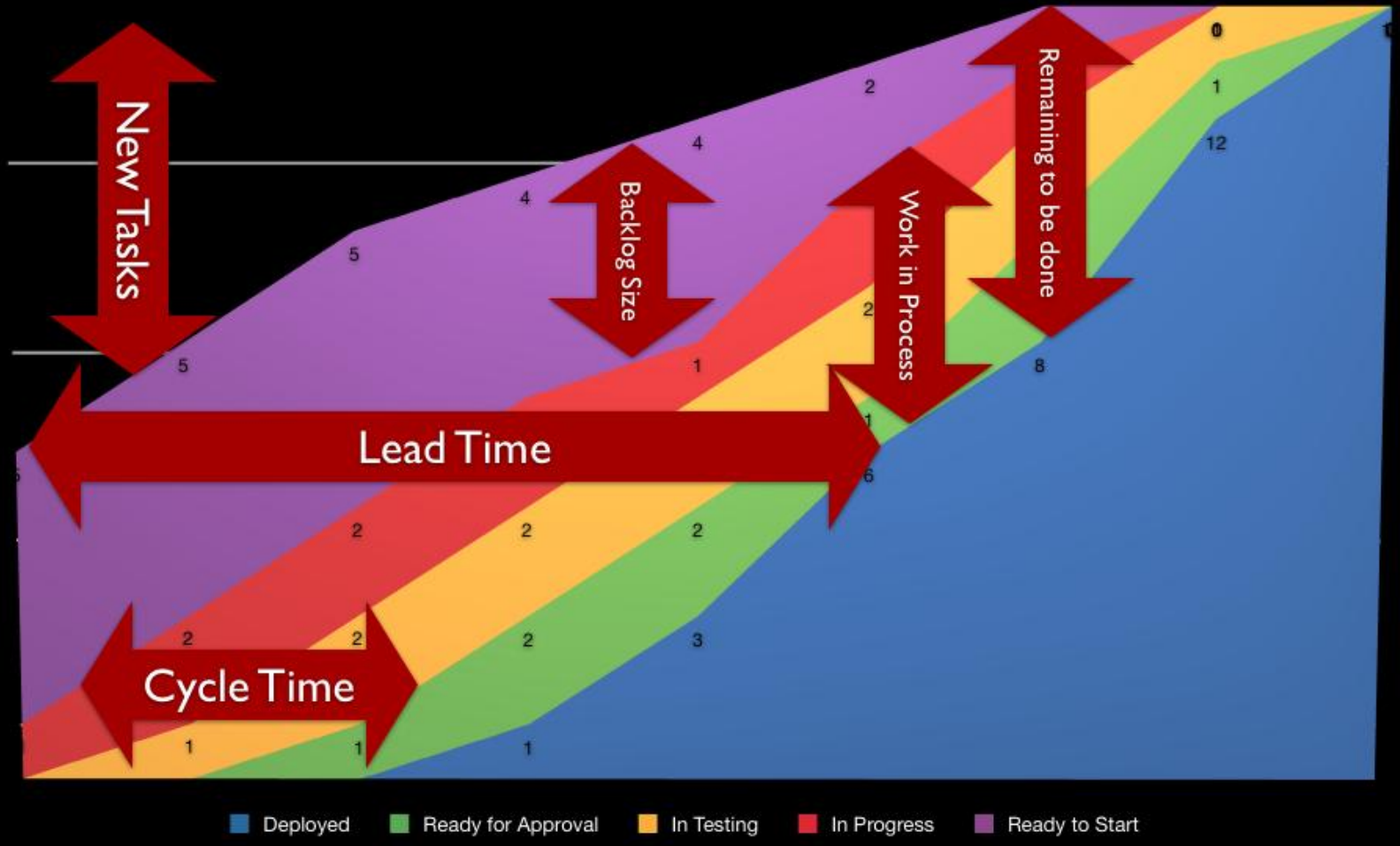



Jun 2, 2013 – Nov 10, 2015
Contributions to develop, excluding merge commits
Contributions: Commits
60
40
20
0
July
massivekarenw
975 commits / 132,3
July
2014
pixelami
502 commits / 119,8
July
2014
tomaskukol
428 commits / 39,14
July
2014

Contributors
Traffic
Commits
Code frequency
Punch card
Network
Members
Additions and Deletions per week
40k
30k
20k
10k
0
10k
20k
30k
40k
07/13
10/13
01/14

Contributors
Traffic
Commits
Code frequency
Punch card
Network
Members
Use ← ar
60
40
20
0
11/16
12/07
12/28
01/18
02/08
03/01
03/22
04/12
05/03
05/24
06/14
07/05
07/26
08/16
09/06
09/27
25
20
15
10
5
0
17
13
9
8
6
0
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday

Cumulative Flow Diagram



Wrap up

Instructions for use

OFTEN

More feedback loops will allow your project to hit your users/customers goals

SHORT

Keep the iterations as short as possible in order to close more feedback loops as possible

CONTINUOUS IMPROVEMENT

Add an improvement per time, don't add too many techniques too often!

MEASURABLE

Collect information and metrics in order to improve your current situation

Thanks!

Any questions?

You can find me at:

@lucamezzalira
mezzalab@gmail.com
www.lucamezzalira.com