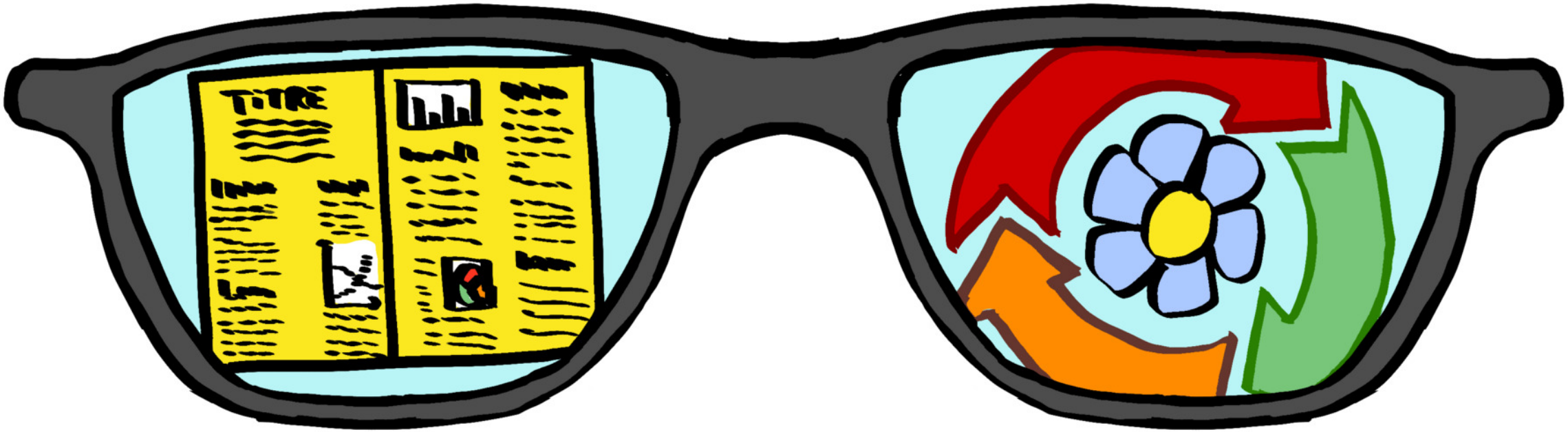



INTRO

{

NAME: VICTOR LAMBRET

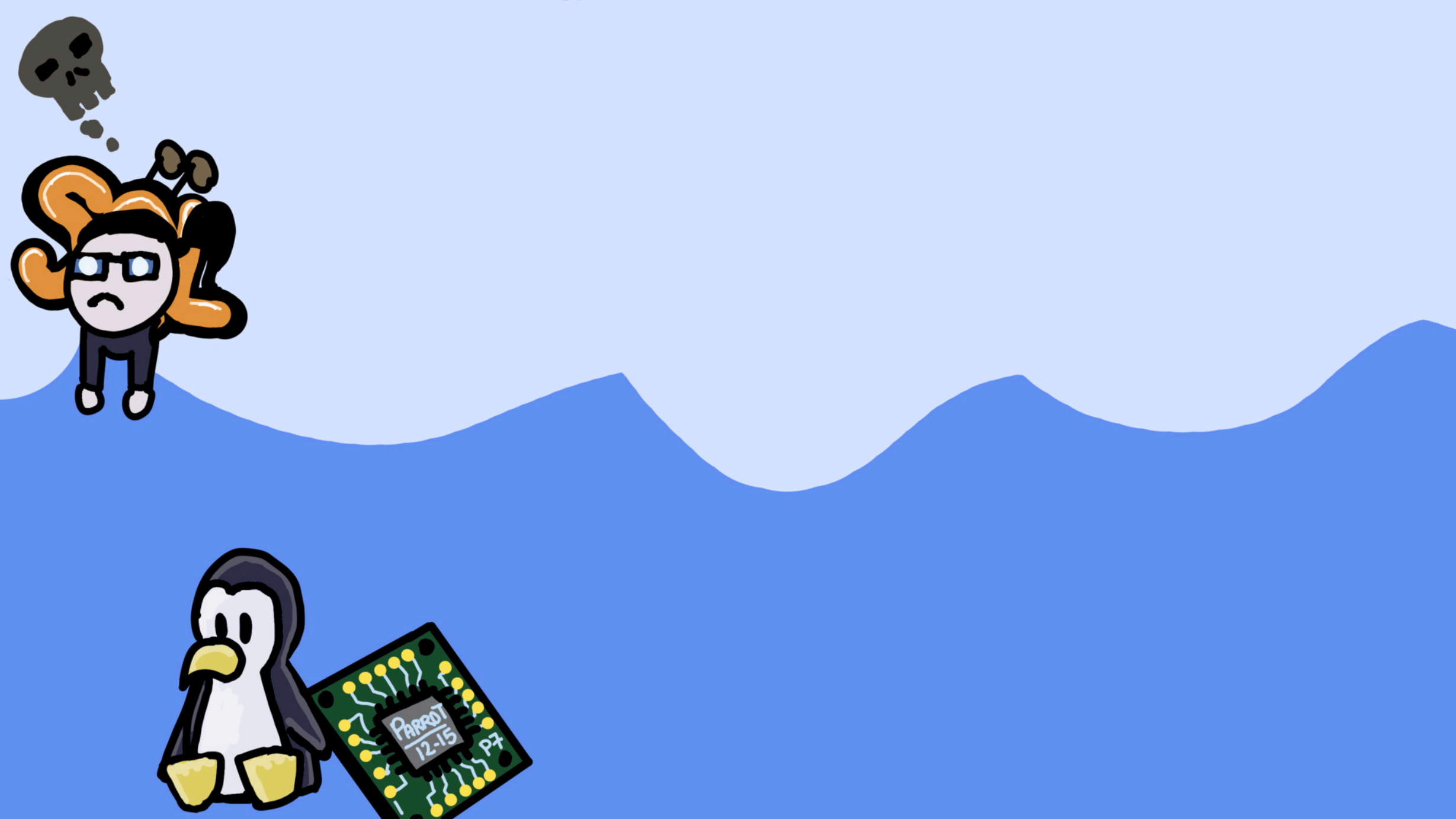
AGE: 0X28

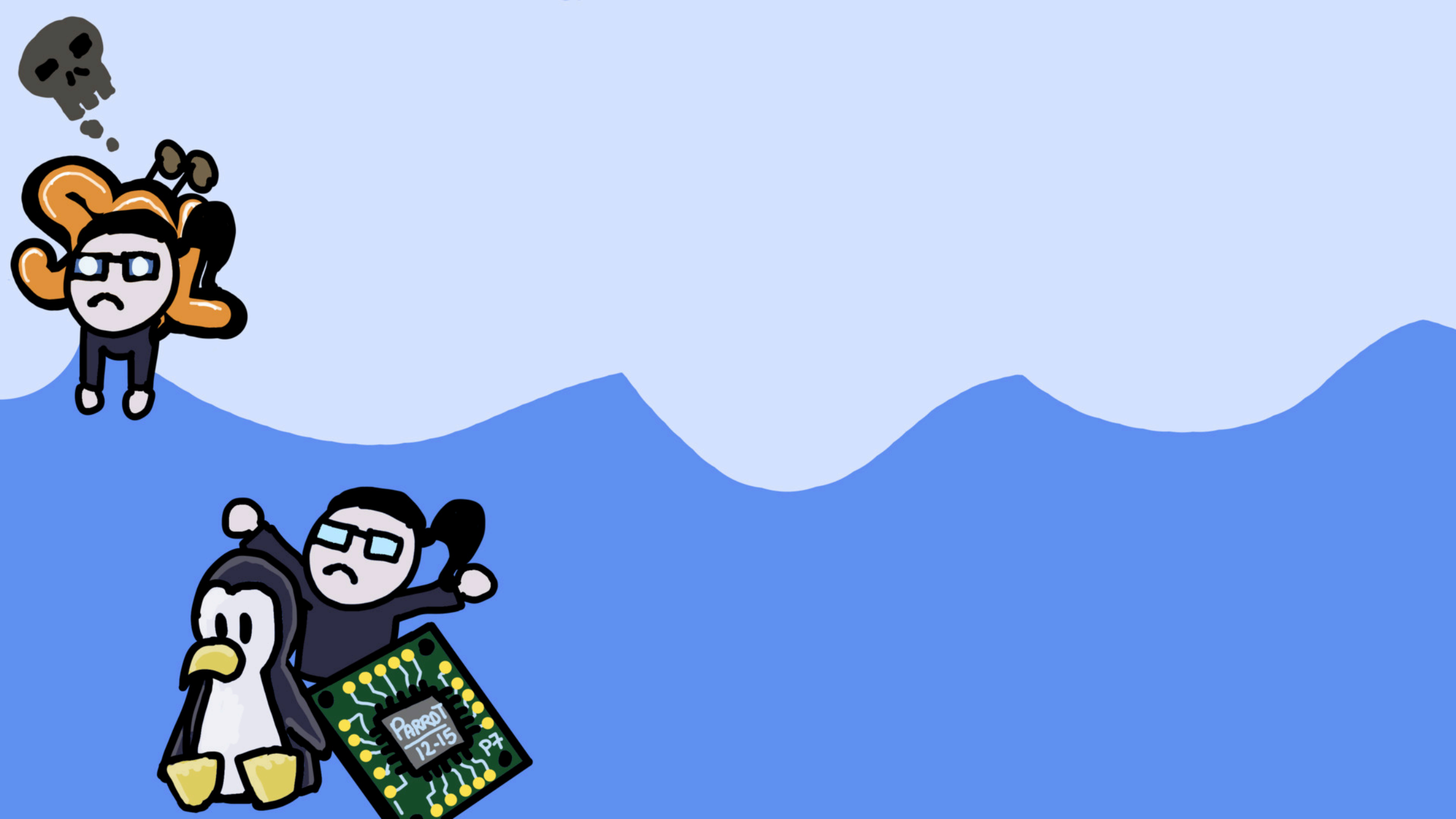
JOB: BACKEND DEVELOPPER

COMPANY: AGICAP

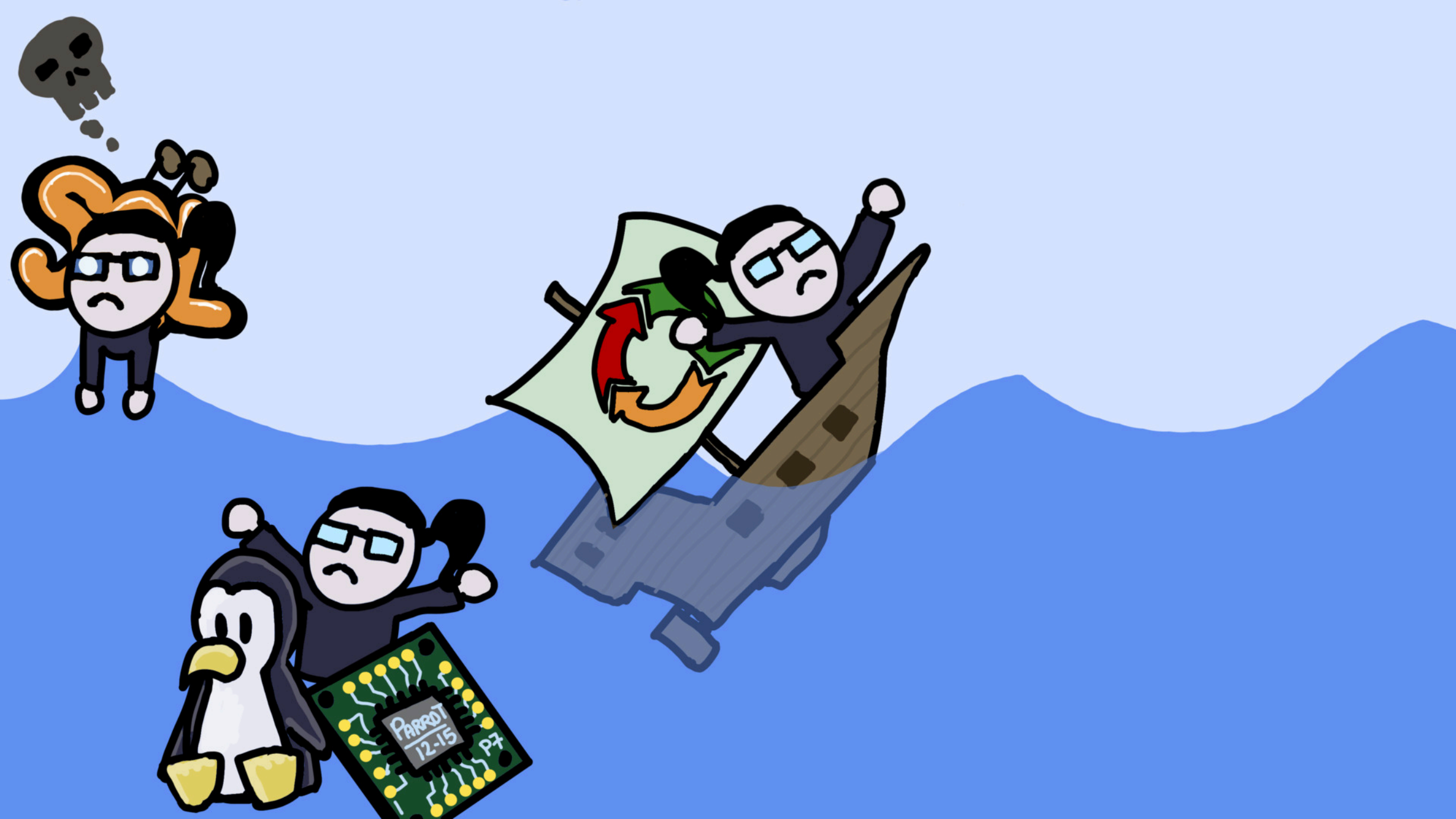
}

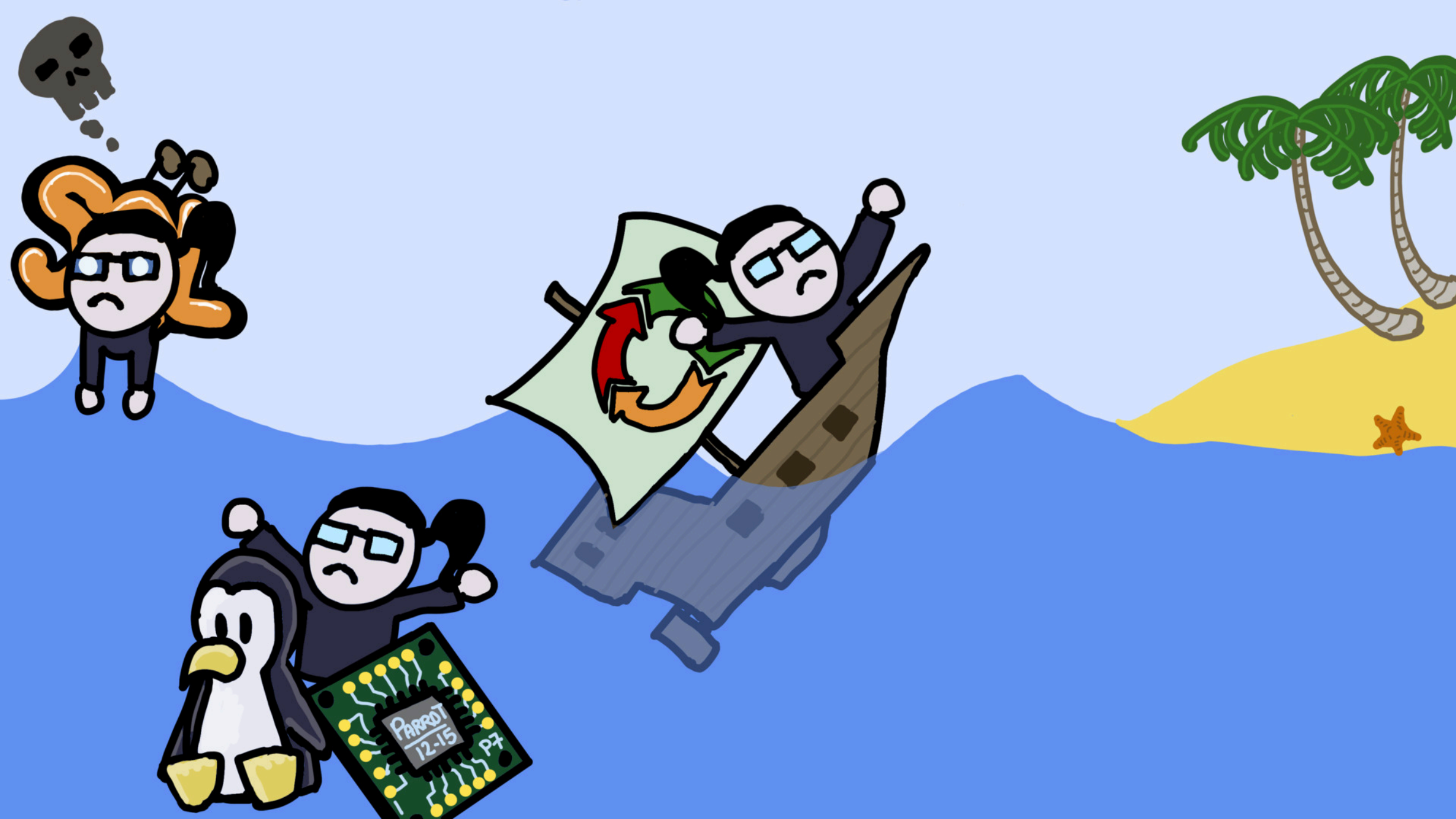

SQL

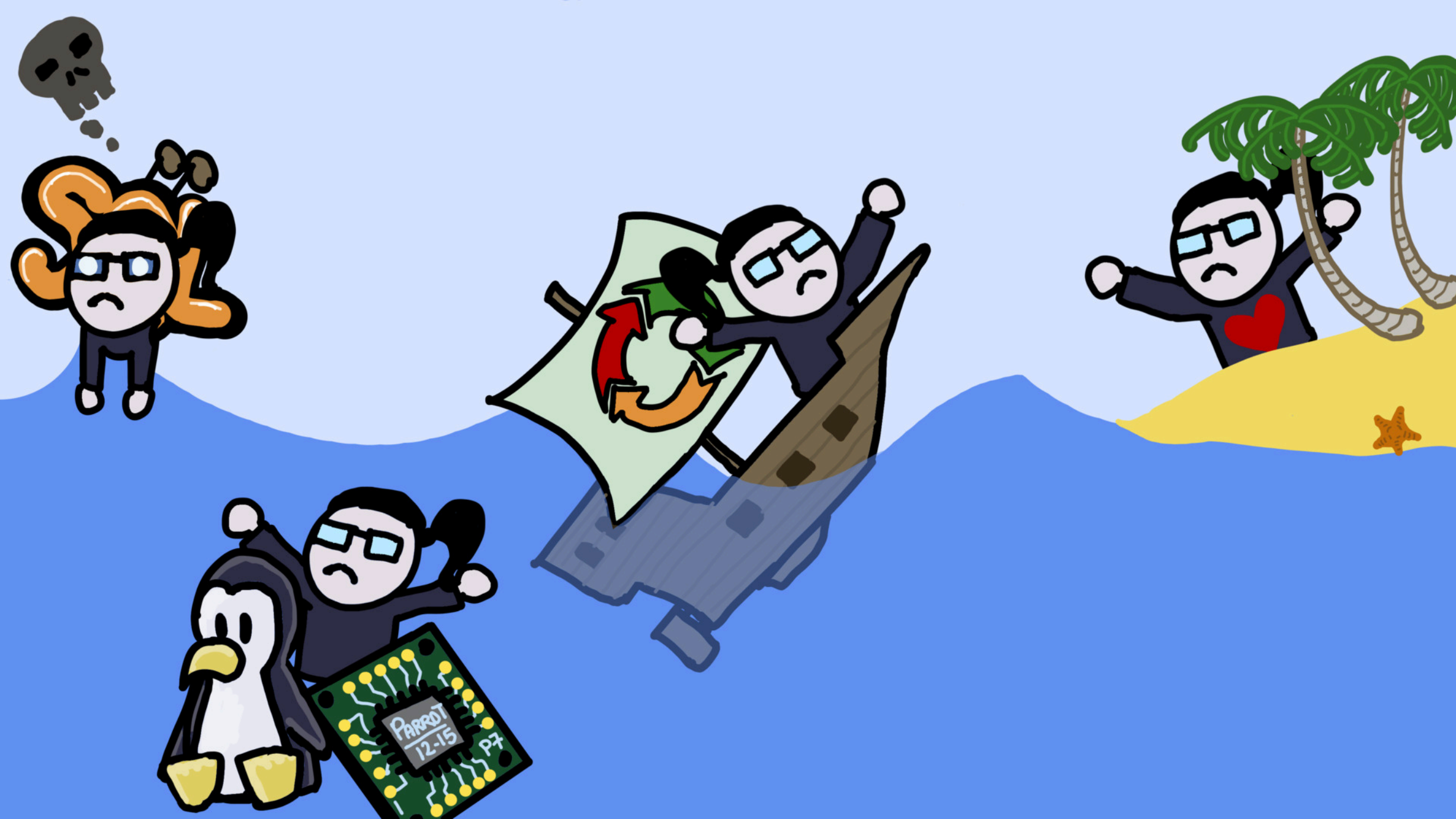

PARROT
12-15
P7

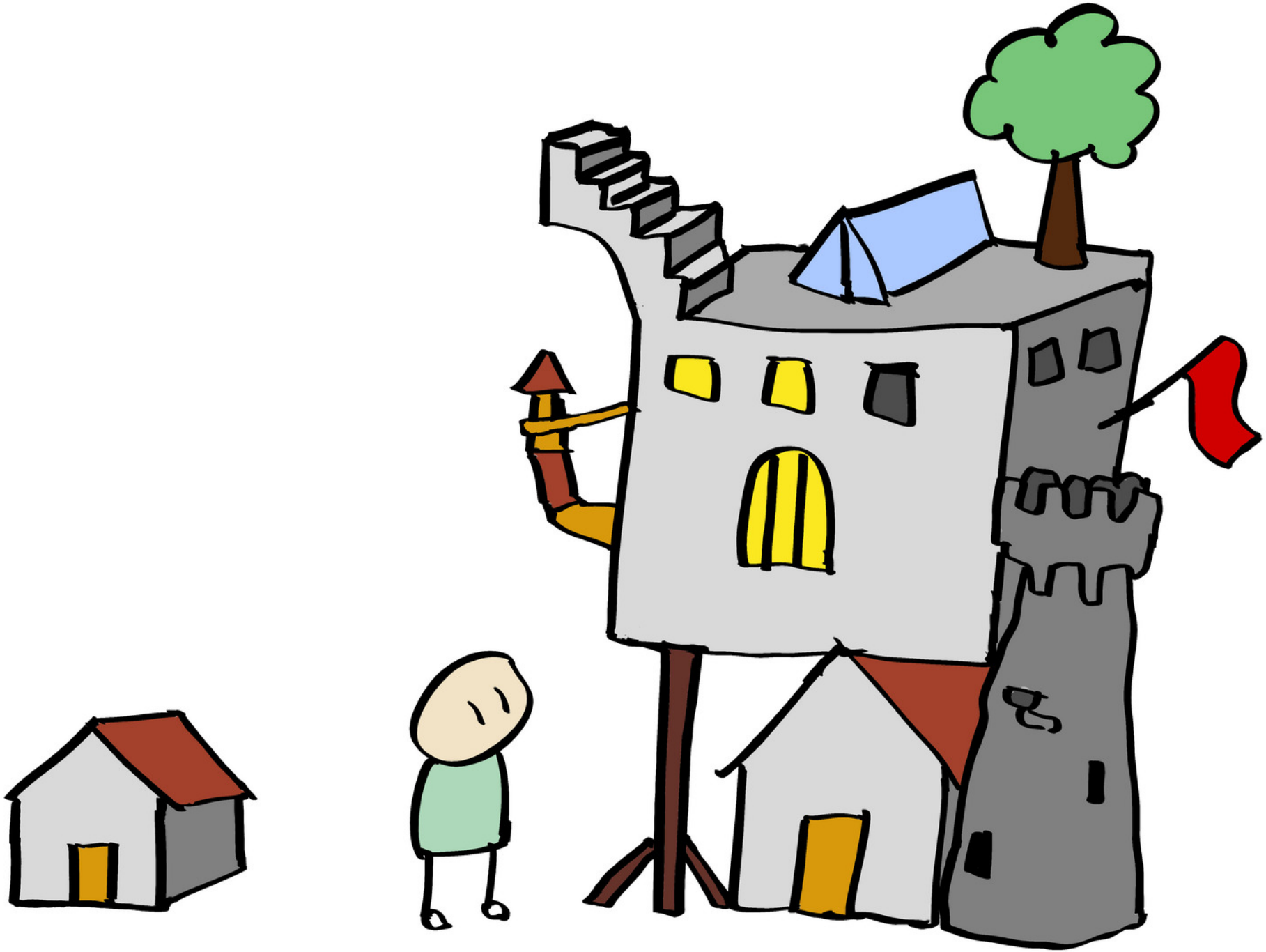

PARROT
12-15
P7

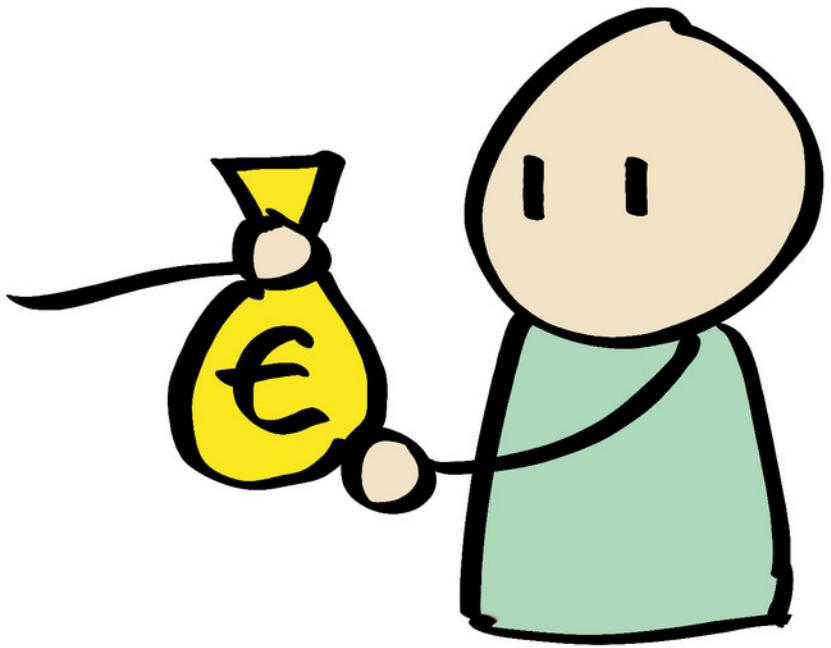

PARROT
12-15
P7

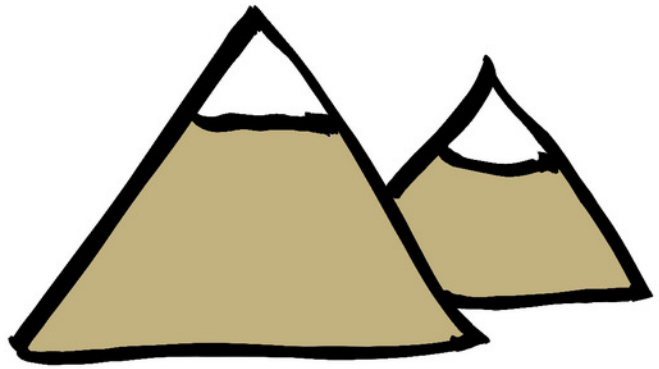

PARROT
12-15
P7

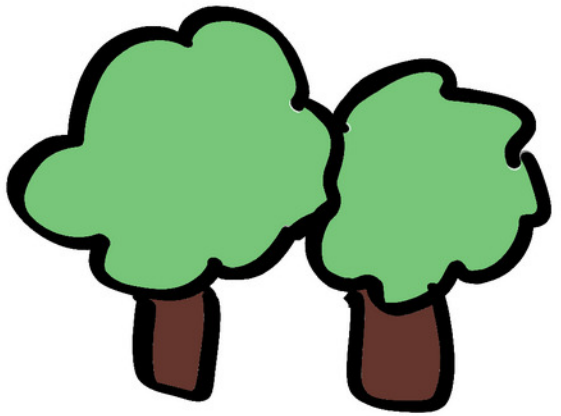

PARROT
12-15
P7

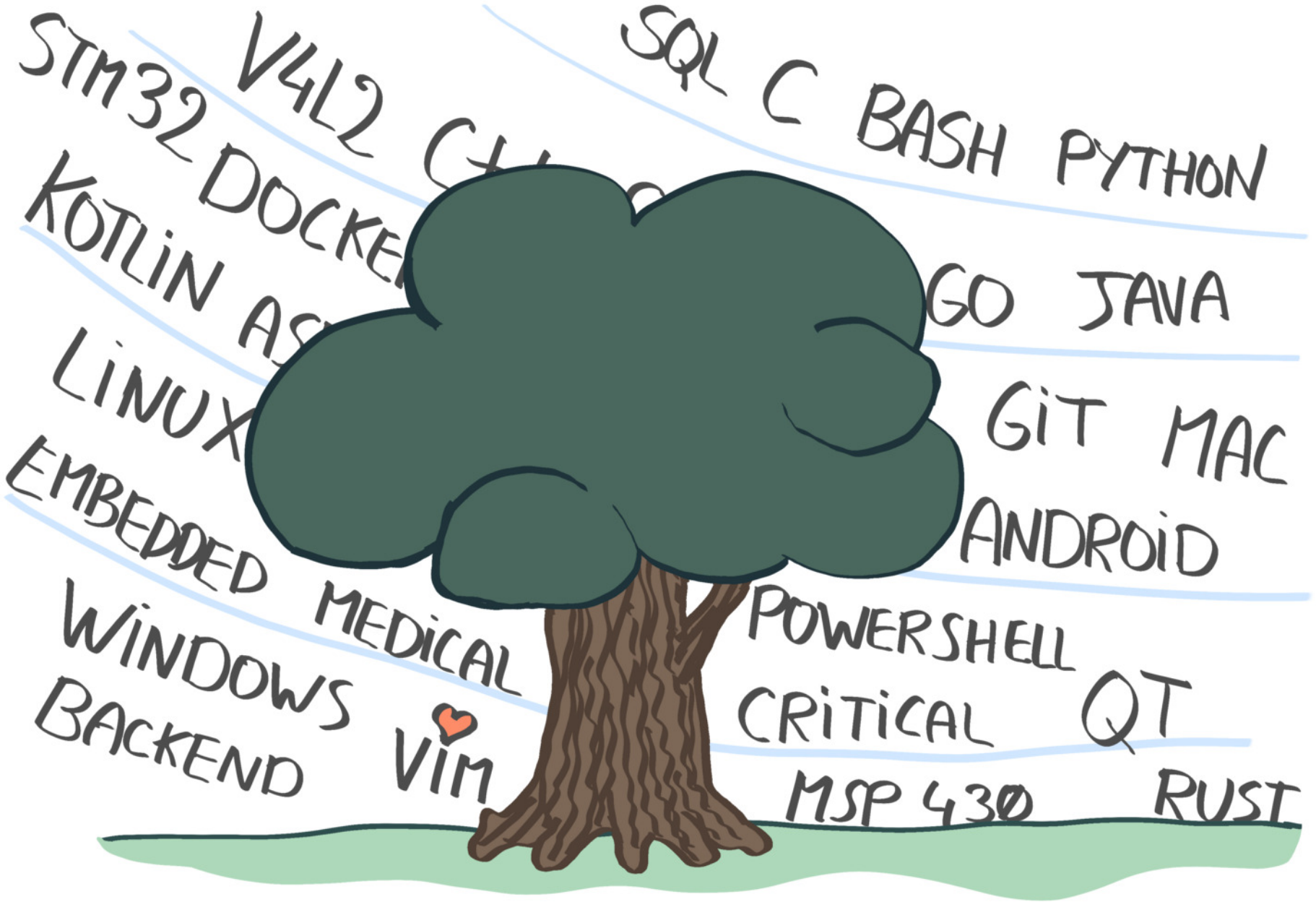

PARROT
12-15
P7

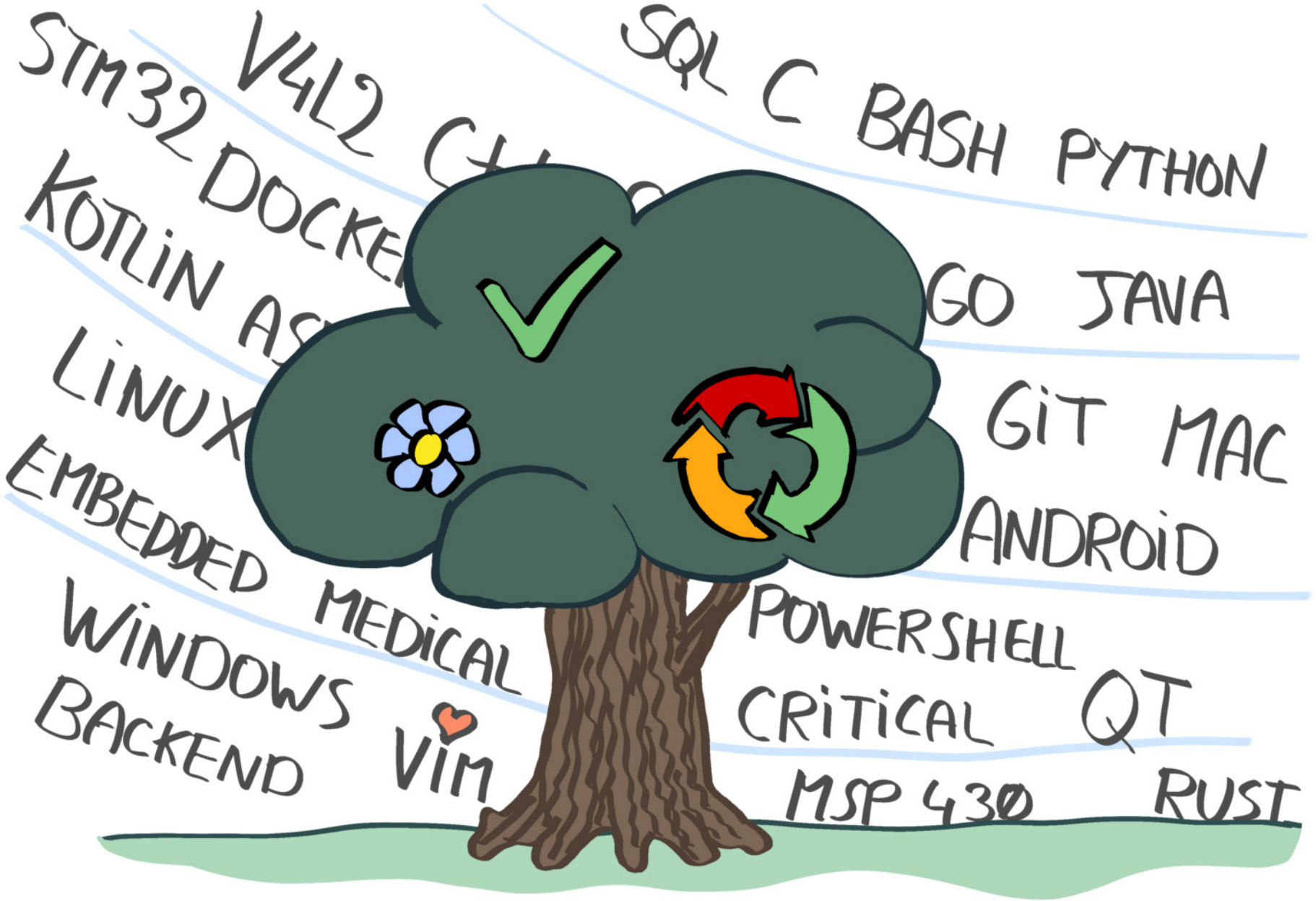


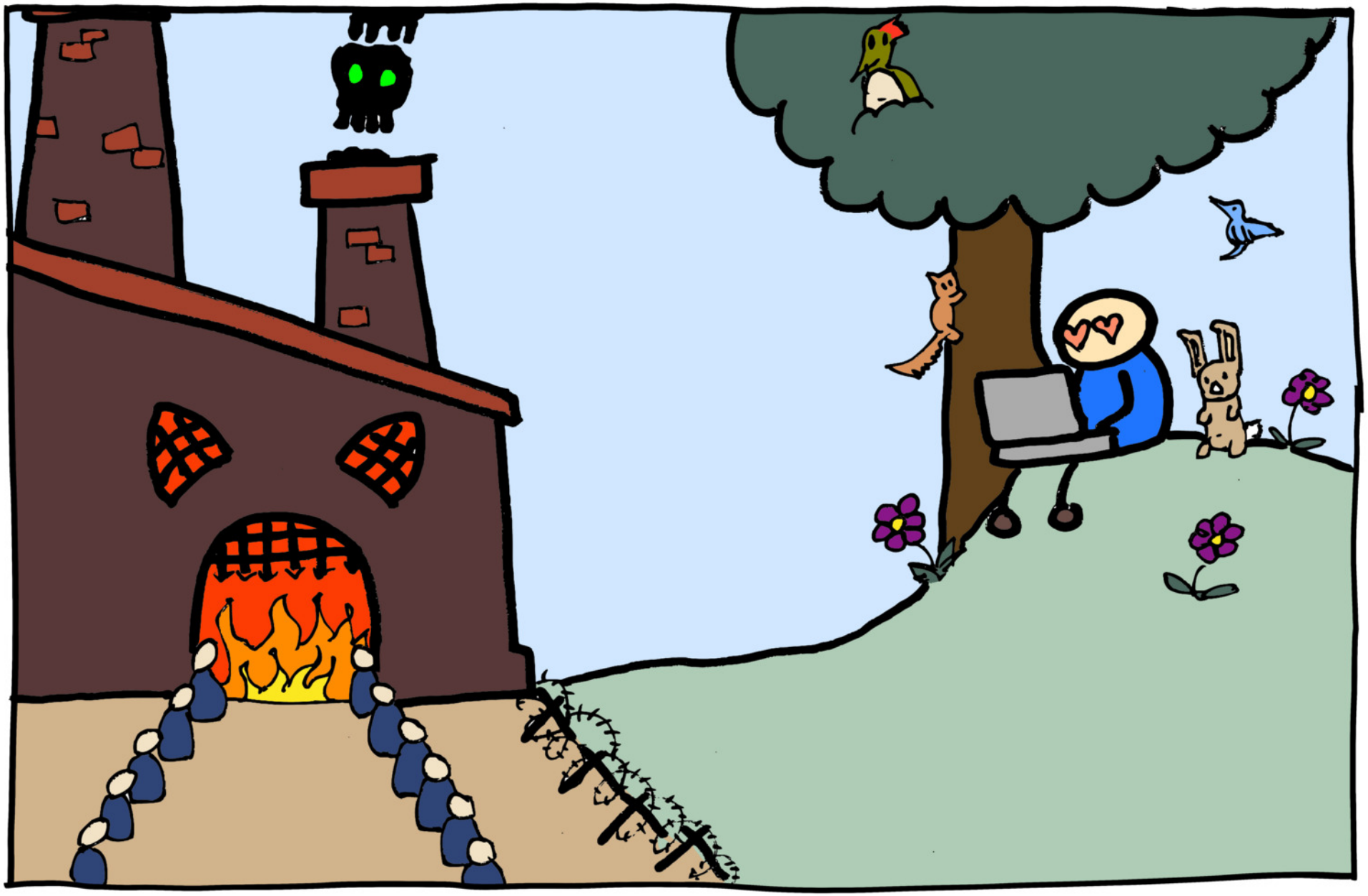


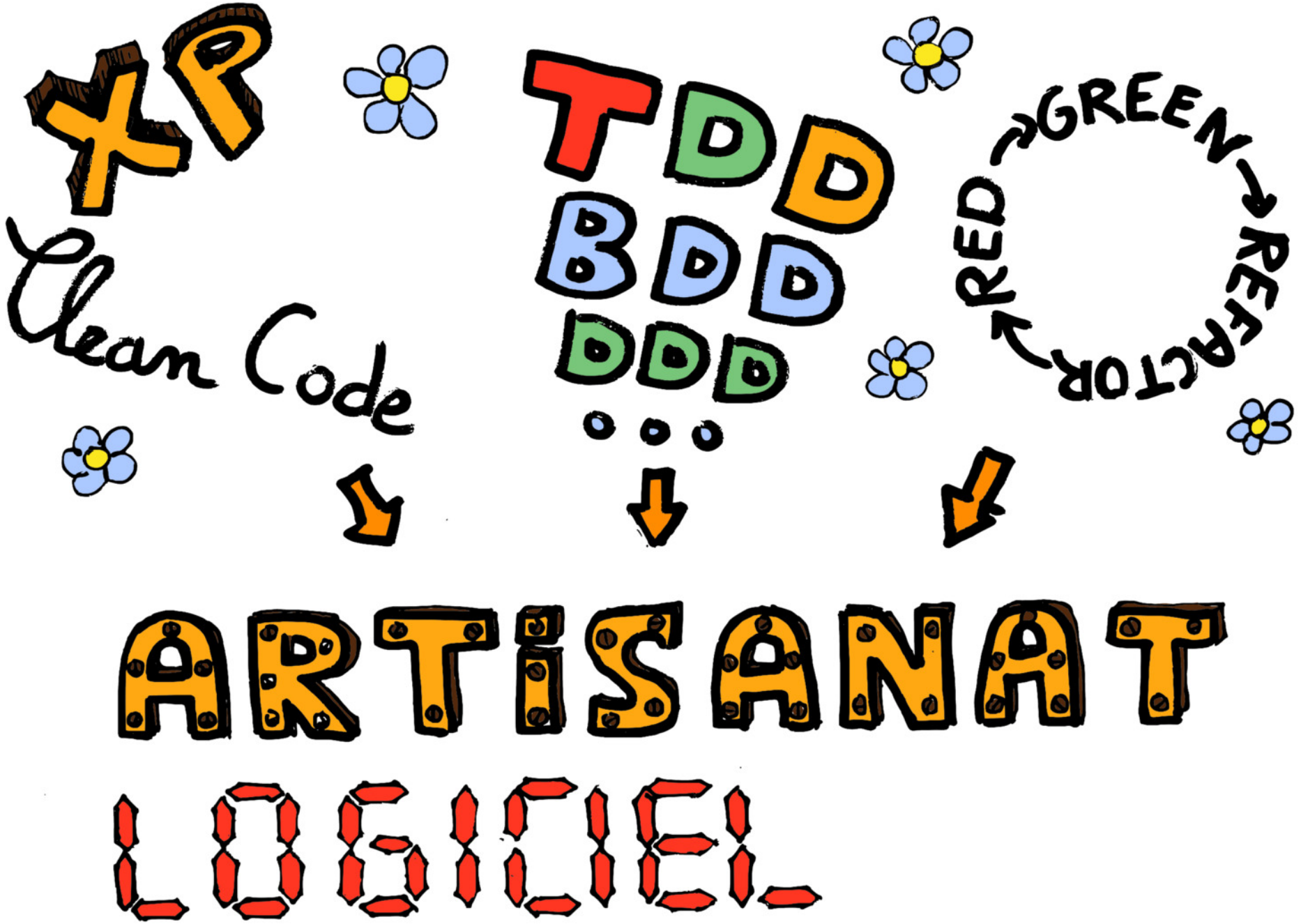





SQL C BASH PYTHON
V4L2
STM32 DOCKE
KOTLIN AS
GO JAVA
LINUX
GIT MAC
EMBEDDED MEDICAL
ANDROID
WINDOWS
POWERSHELL
BACKEND
VIM
CRITICAL
QT
MSP 430
RUST


V4L2
SQL
C
BASH
PYTHON
STM32
DOCKER
GO
JAVA
KOTLIN
LINUX
GIT
MAC
ANDROID
EMBEDDED
MEDICAL
POWERSHELL
WINDOWS
CRITICAL
QT
BACKEND
VIM
MSP 430
RUST




XP
Clean Code
TDD
BDD
DDD
...
RED → GREEN → REFACTOR → RED
ARTISANAT
LOGICIEL

XP

Clean Code

TDD

BDD

DDD

...

RED → GREEN → REFACTOR

ARTISANAT LOGICIEL

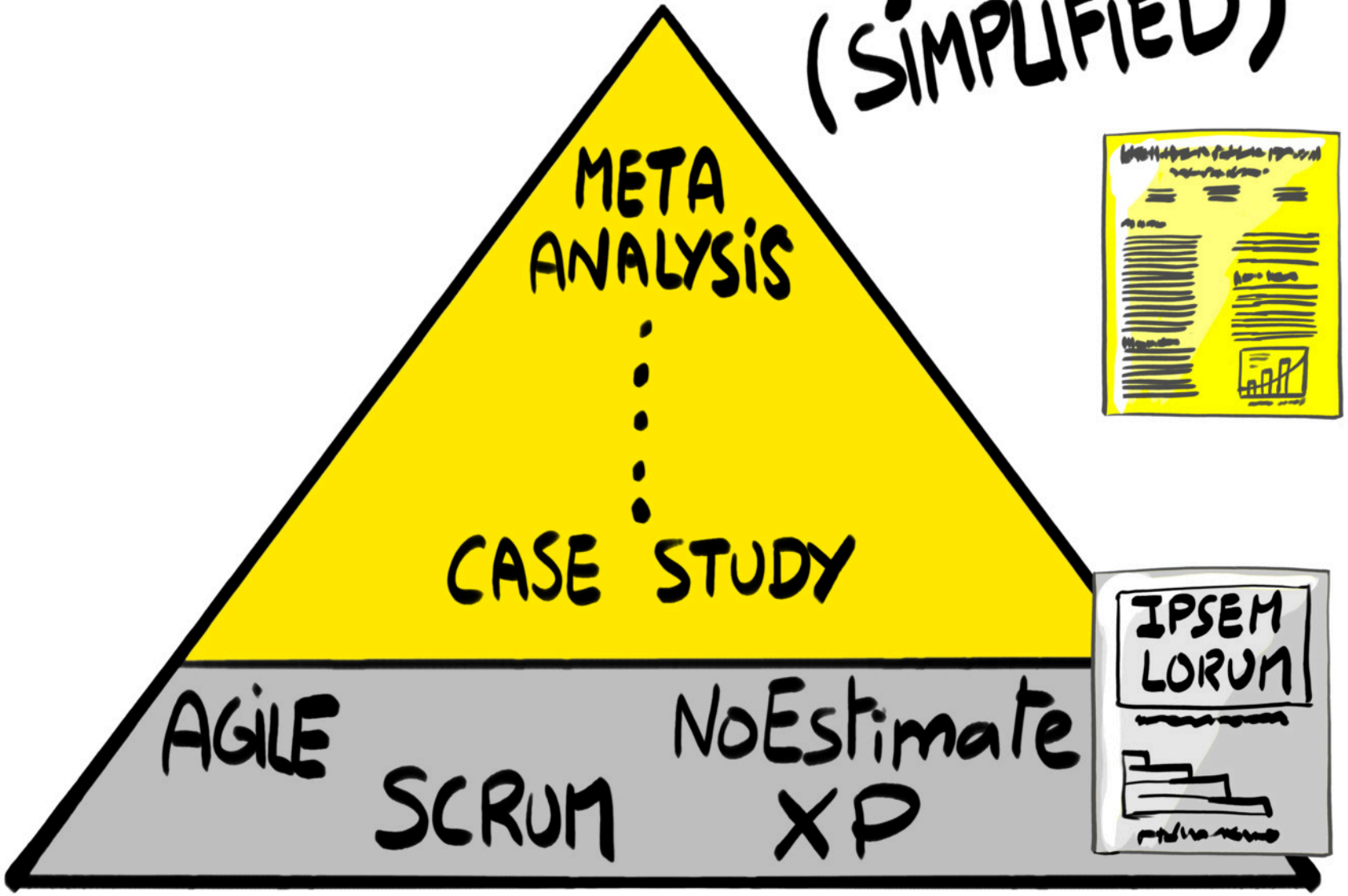
ARTISANAT LOGICIEL

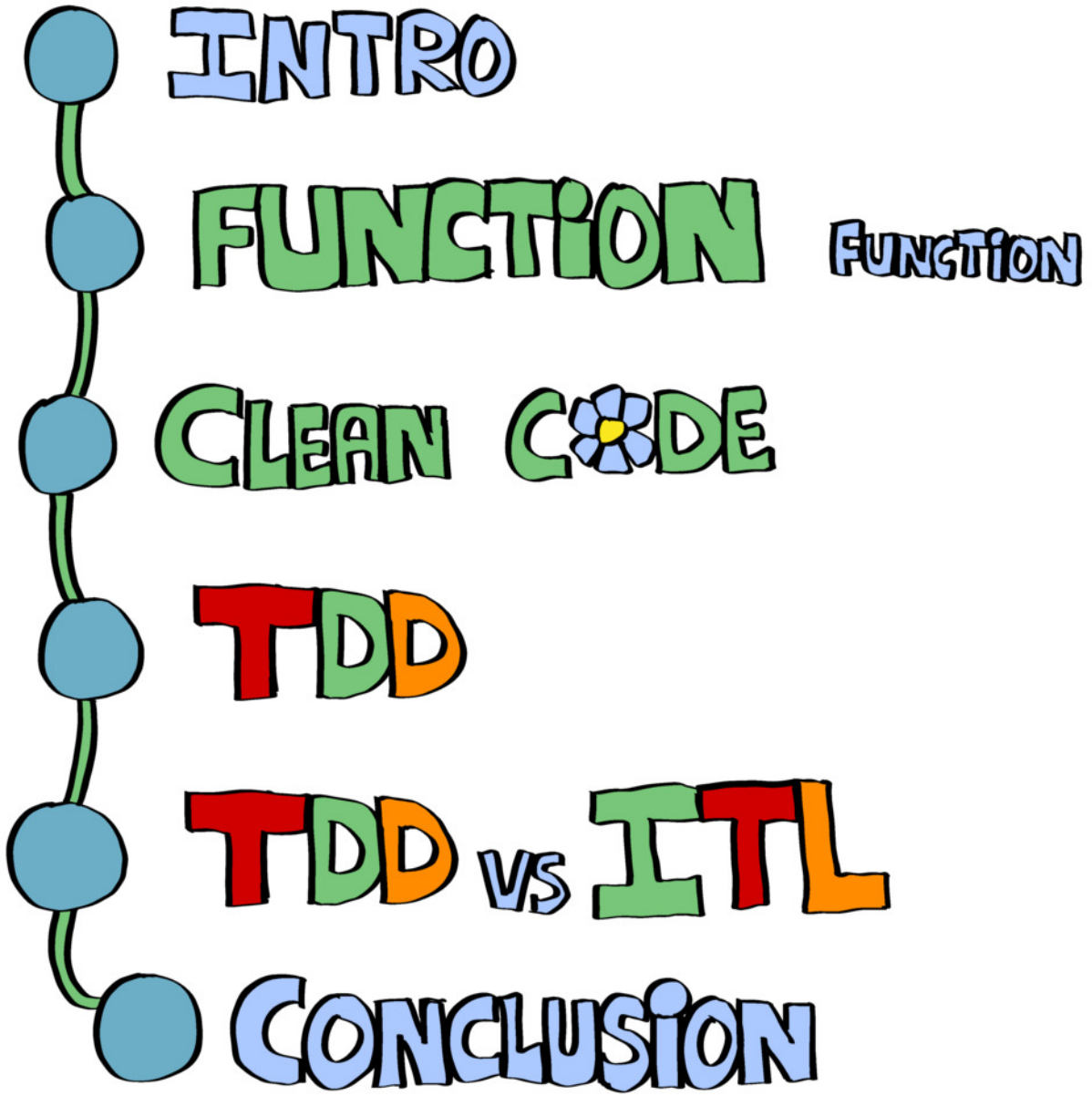
UN [illegible] ZÉTÉTIQUE

LE TDD

SANS COMMENCER PAR LES TESTS ?

LEVELS OF EVIDENCE (SIMPLIFIED)




INTRO
FUNCTION
FUNCTION
CLEAN CODE
TDD
TDD VS ITL
CONCLUSION


INTRO
FUNCTION
FUNCTION

CLEAN CODE: A HANDBOOK OF AGILE SOFTWARE CRAFTSMANSHIP

Martin, Robert

2008

CLEAN CODE : TAILLE DES FONCTIONS

> *The first rule of functions is that they should be small. The second rule of functions is that they should be smaller than that.*

CLEAN CODE : TAILLE DES FONCTIONS

Every function in this program was just two, or three, or four lines long. Each was transparently obvious. Each told a story. And each led you to the next in a compelling order. That's how short your functions should be !

CLEAN CODE : TAILLE DES FONCTIONS

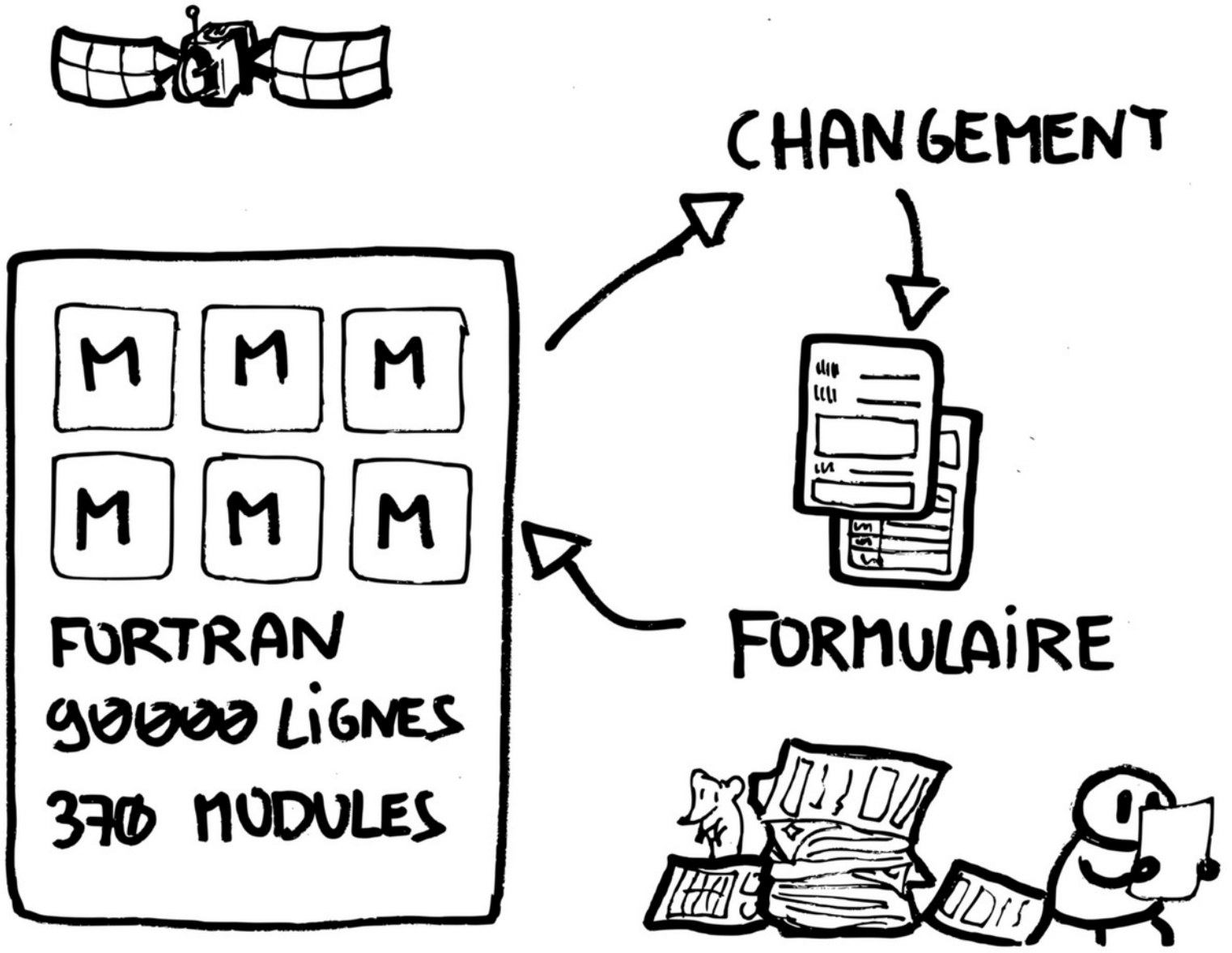
> *This is not an assertion that I can justify. I can't provide any references to research that shows that very small functions are better*

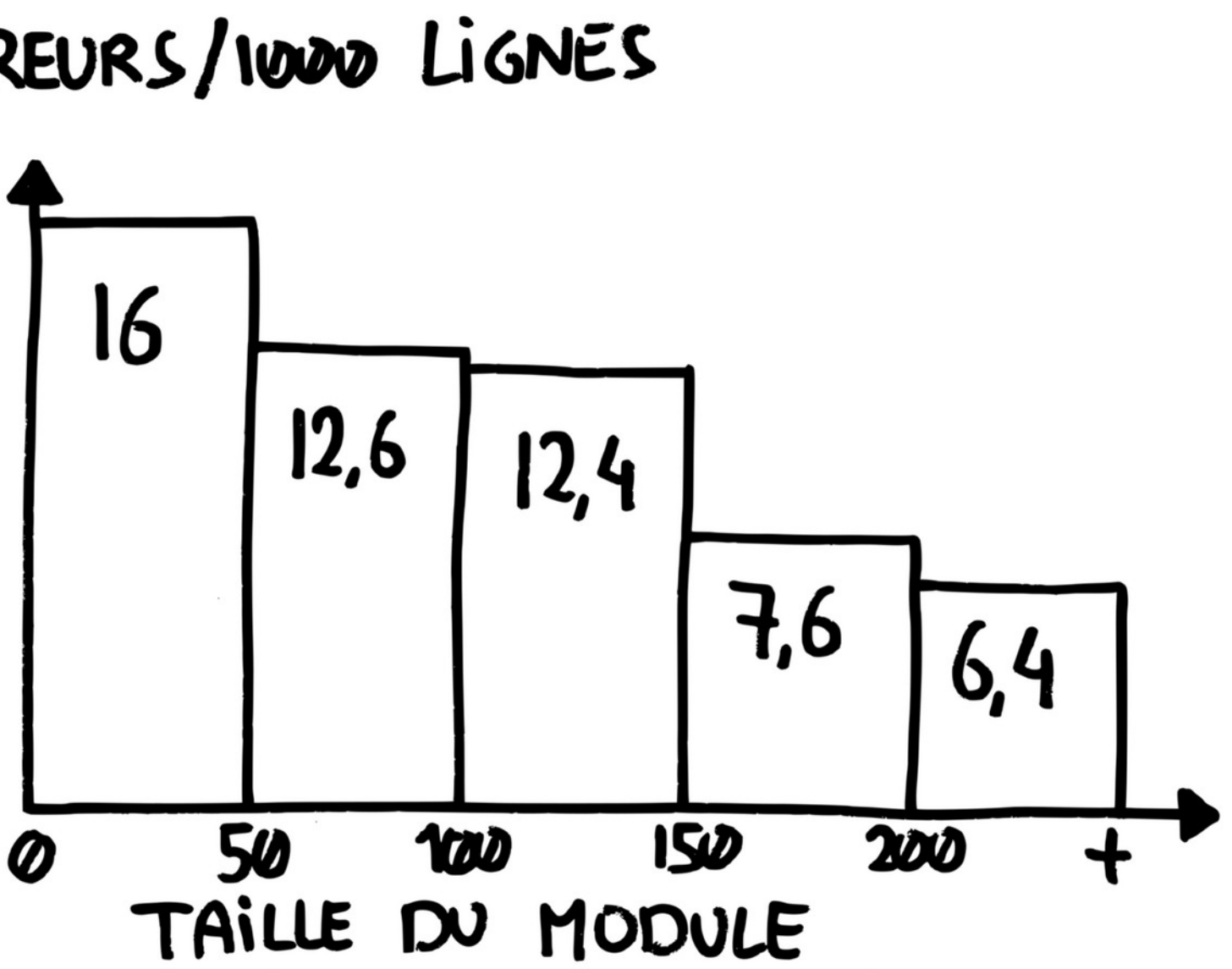
Pas de références ? Aucun soucis, c'est une telle évidence qu'on va trouver facilement :-)

SOFTWARE ERRORS AND COMPLEXITY: AN EMPIRICAL INVESTIGATION

Basili, V. and Perricone, B.

1983


CHANGEMENT
M M M
M M M
FORTRAN
90000 LIGNES
370 MODULES
FORMULAIRE


ERREURS/1000 LIGNES
16
12,6
12,4
7,6
6,4
0
50
100
150
200
+
TAILLE DU MODULE

CONCLUSION

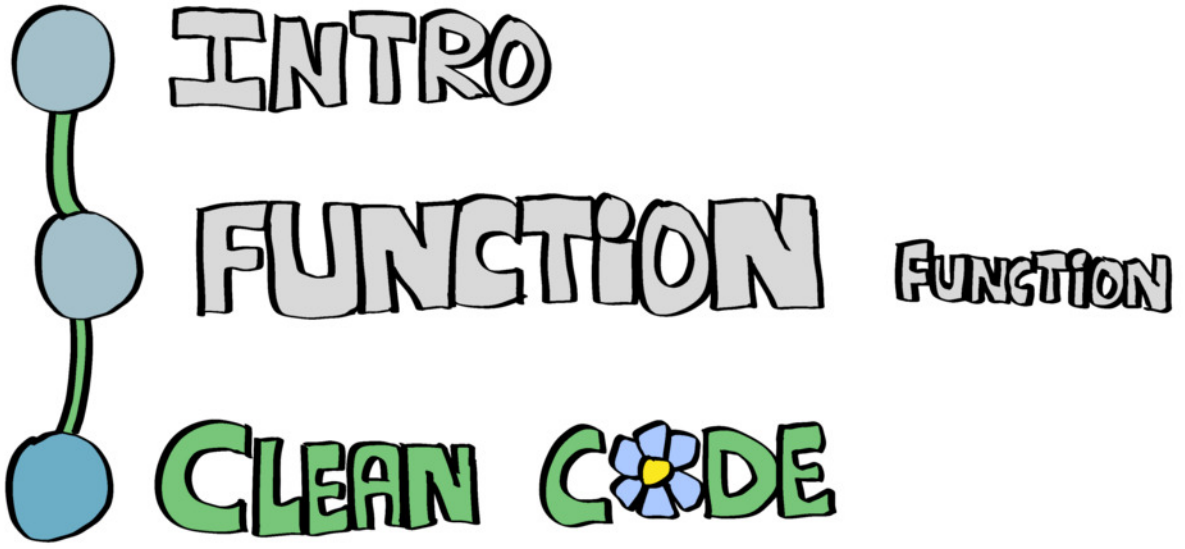
One surprising result was that module size did not account for error proneness. In fact, it was quite the contrary […]. This result implies we are not yet ready to put artificial limits on module size and complexity.

MINCE ALORS !

Les résultats ne sont pas ceux attendus !

Mais c’est une vielle étude. Maintenant, dans le contexte de Clean Code ça marche forcément !

C’est une telle évidence qu’on va trouver facilement :-)


INTRO
FUNCTION
FUNCTION
CLEAN CODE

EFFECTS OF CLEAN CODE ON UNDERSTANDABILITY

Henning Grimeland Koller

2016

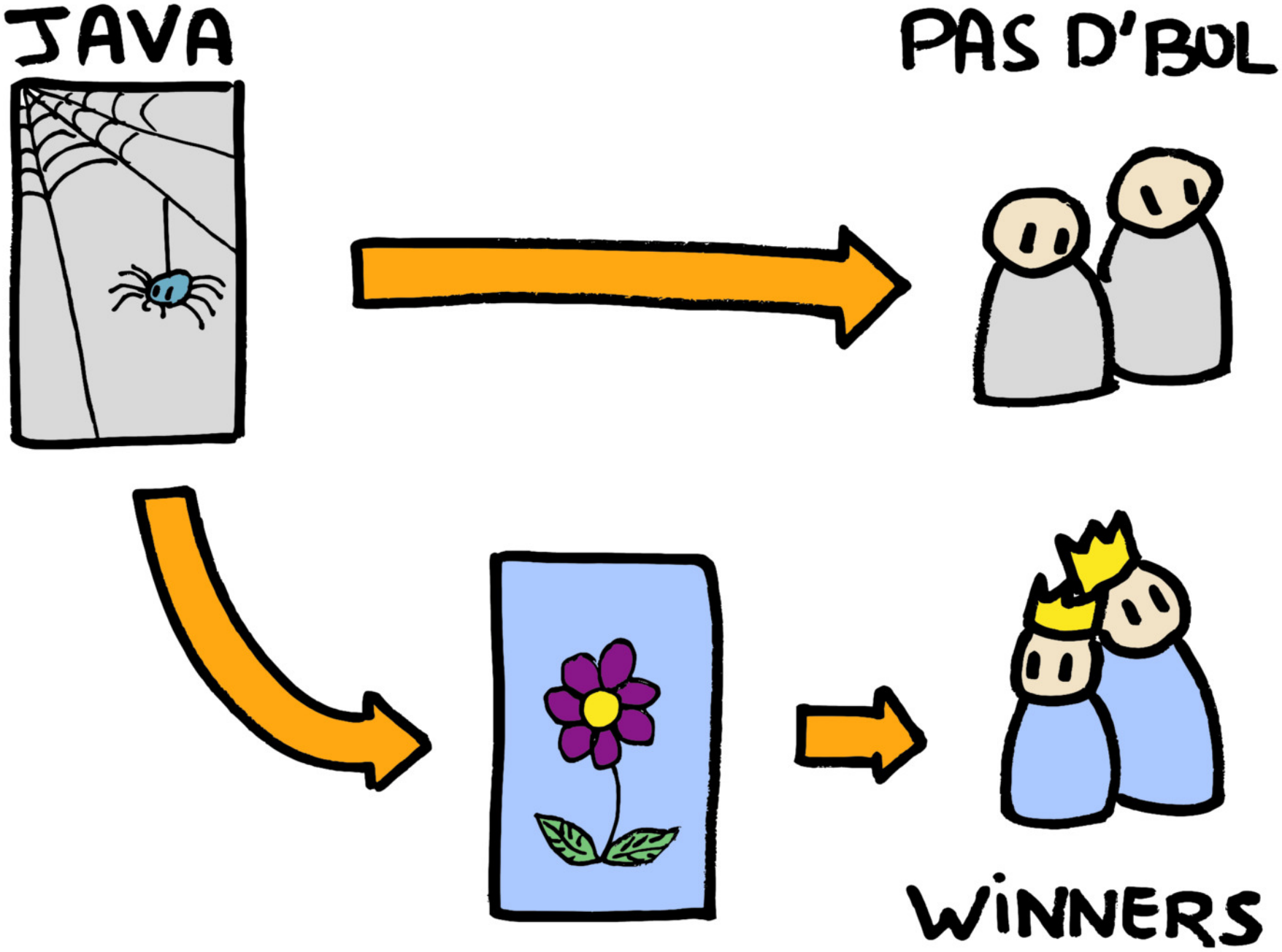

JAVA
PAS D'BOL
WINNERS

Figure 5.11: Second Run of the Experiment: Time used by participants on assignment 1

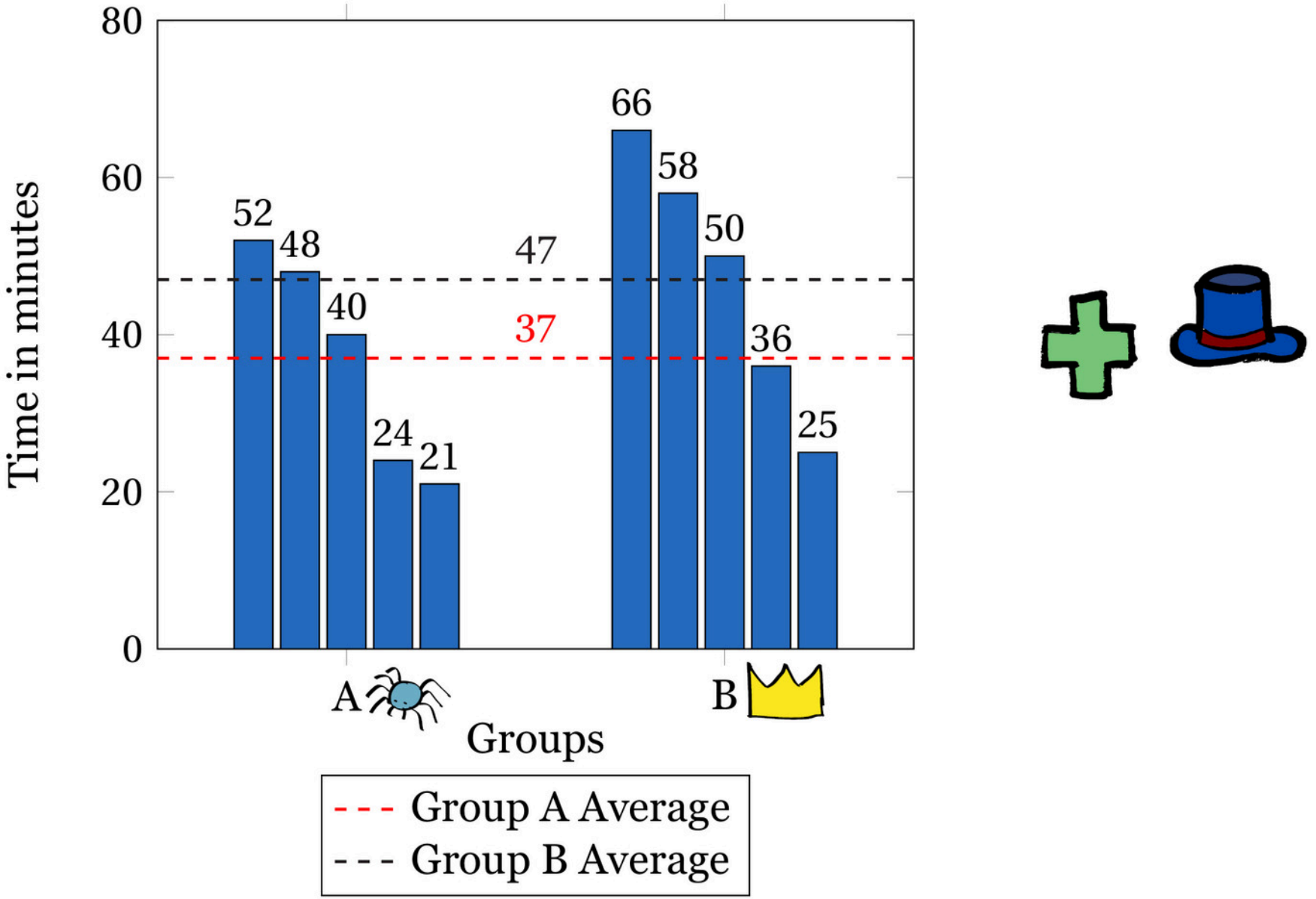


Figure 5.12: Second Run of the Experiment: Time used by participants on assignment 2

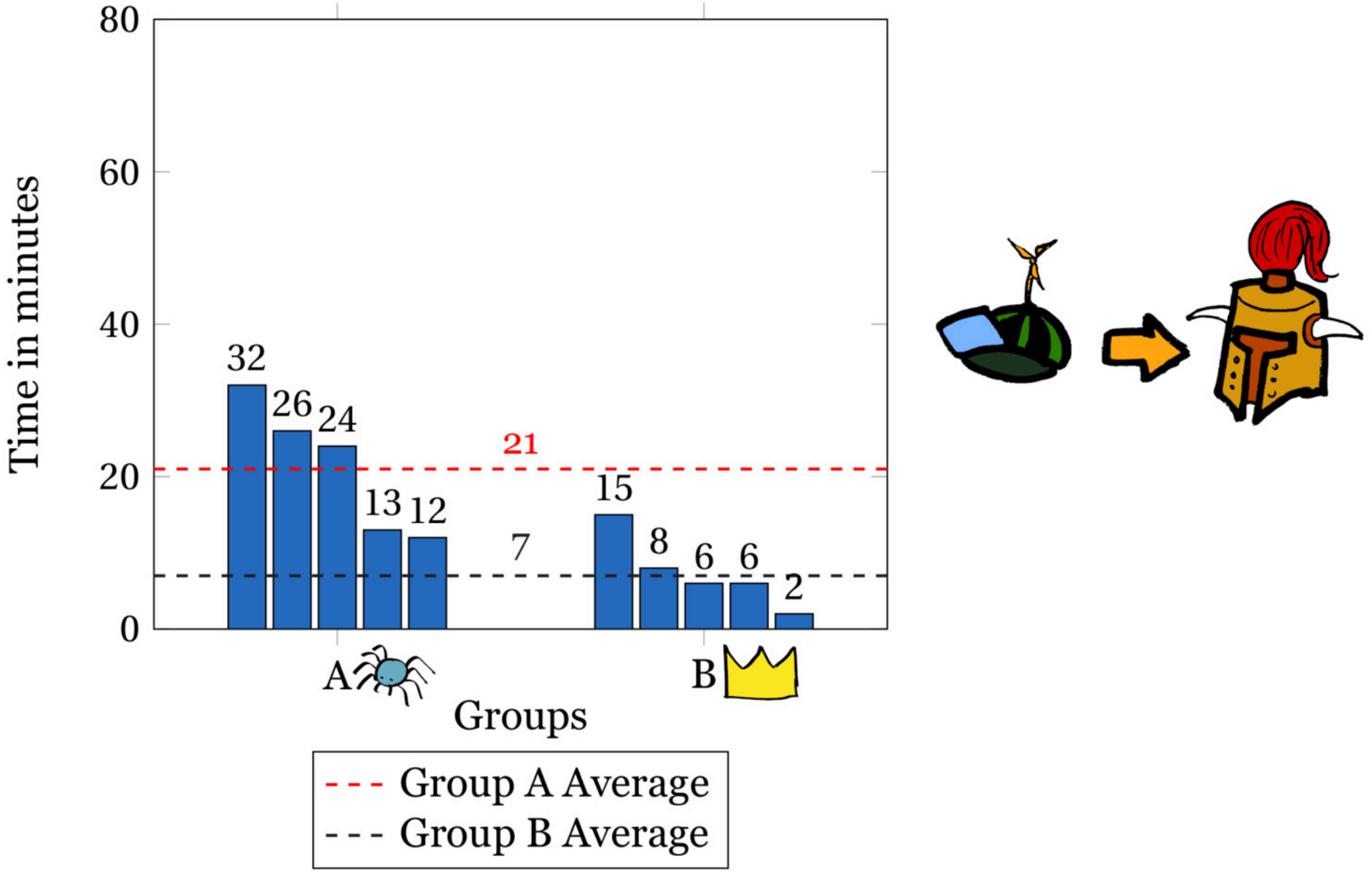
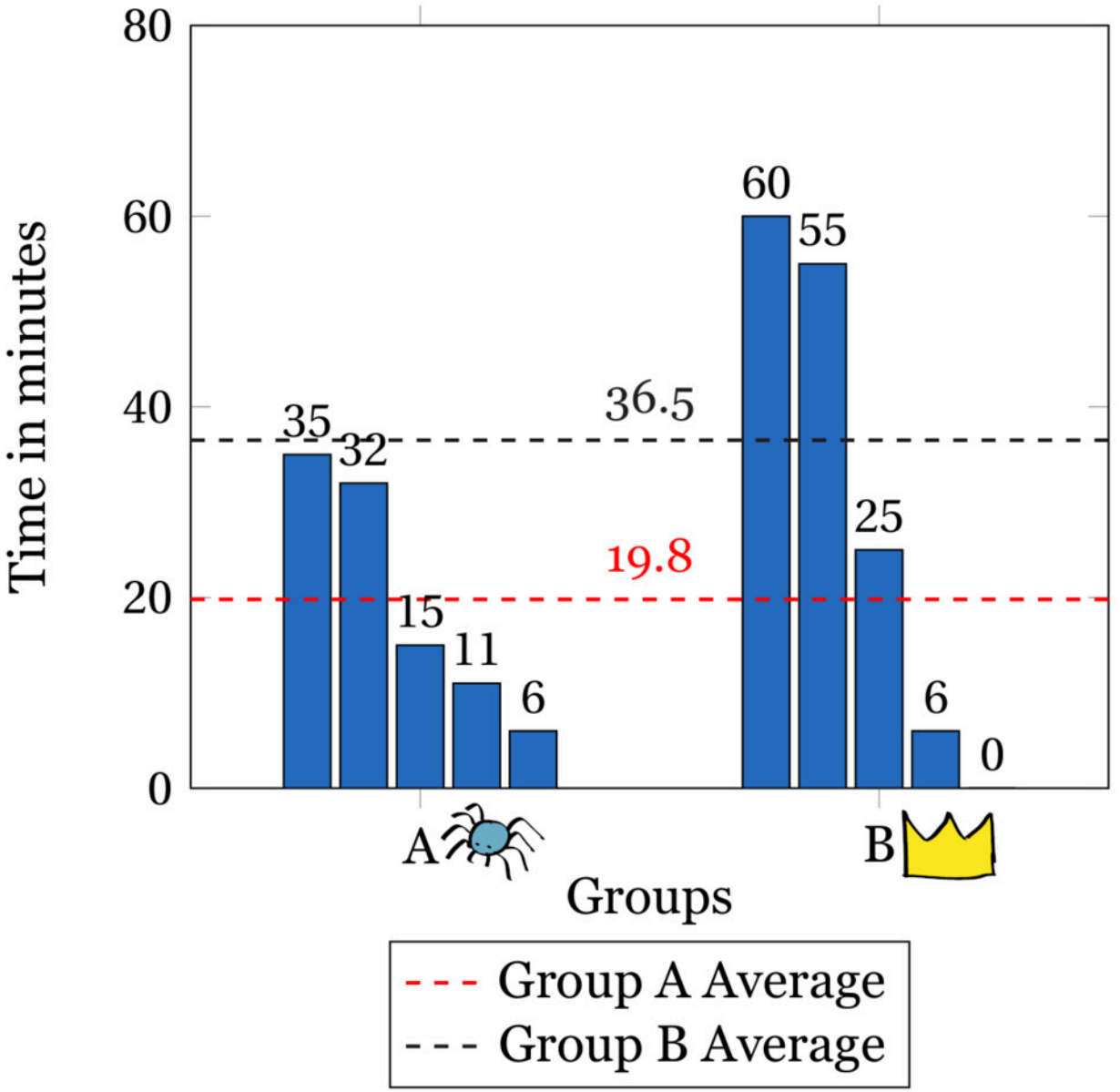


Figure 5.13: Second Run of the Experiment: Time used by participants on assignment 3



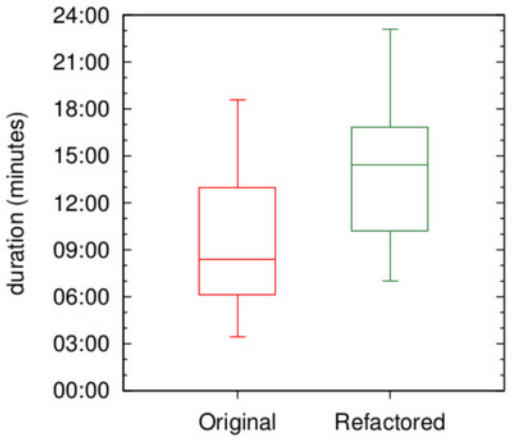
CONCLUSION

Despite our expectations, the results from the experiment show that we were wrong […] there seems to be no immediate benefit of Clean Code in form of understandability.

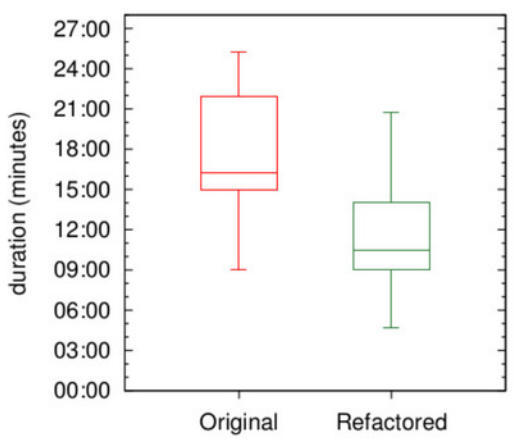
OLD HABITS DIE HARD: WHY REFACTORING FOR UNDERSTANDABILITY DOES NOT GIVE IMMEDIATE BENEFITS

Ammerlaan, Erik and Veninga, Wim and Zaidman, Andy

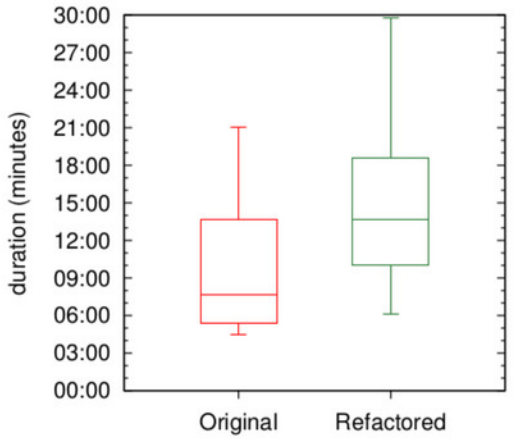
2015 - 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering, SANER 2015 - Proceedings



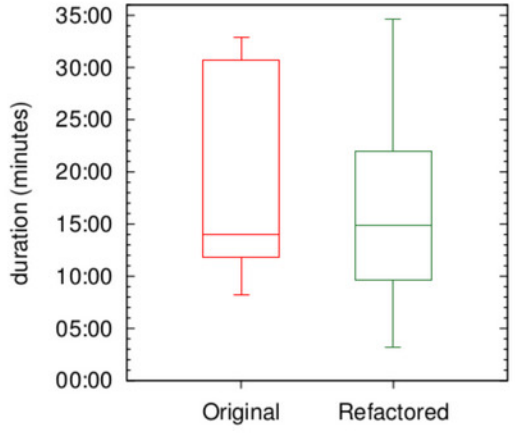
(a) RejectionNotifier (S)



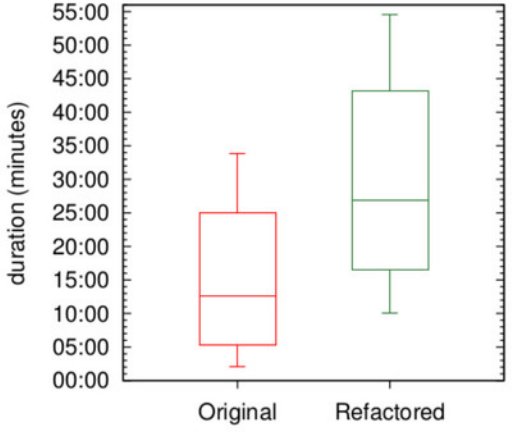
(b) PurchaseOrderTools (S)



(c) ContractProlongation (S)



(d) FinancialEntryTools (M)



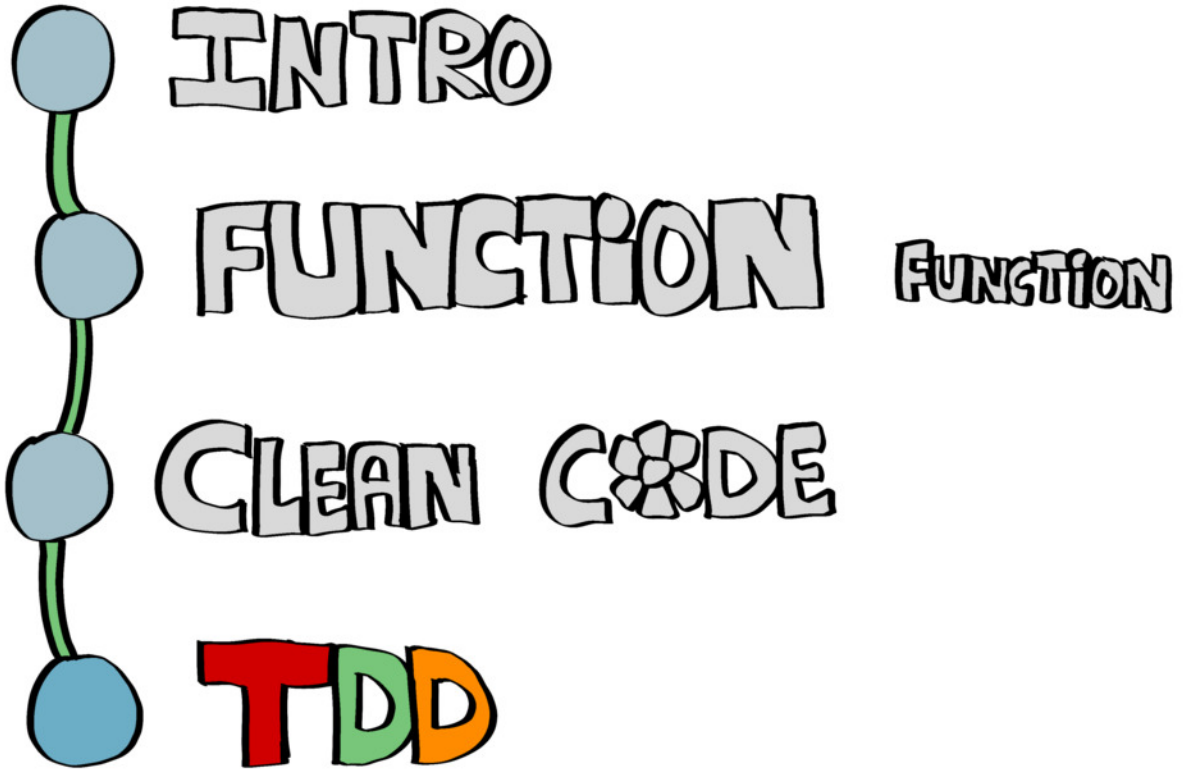
(e) ERDomain (L)

MINCE ALORS !

Les résultats ne sont pas ceux attendus !

Il faut se faire une raison : cette "évidence" n’en est pas une

La réalité est plus compliquée que ça :)


INTRO
FUNCTION
FUNCTION
CLEAN CODE
TDD

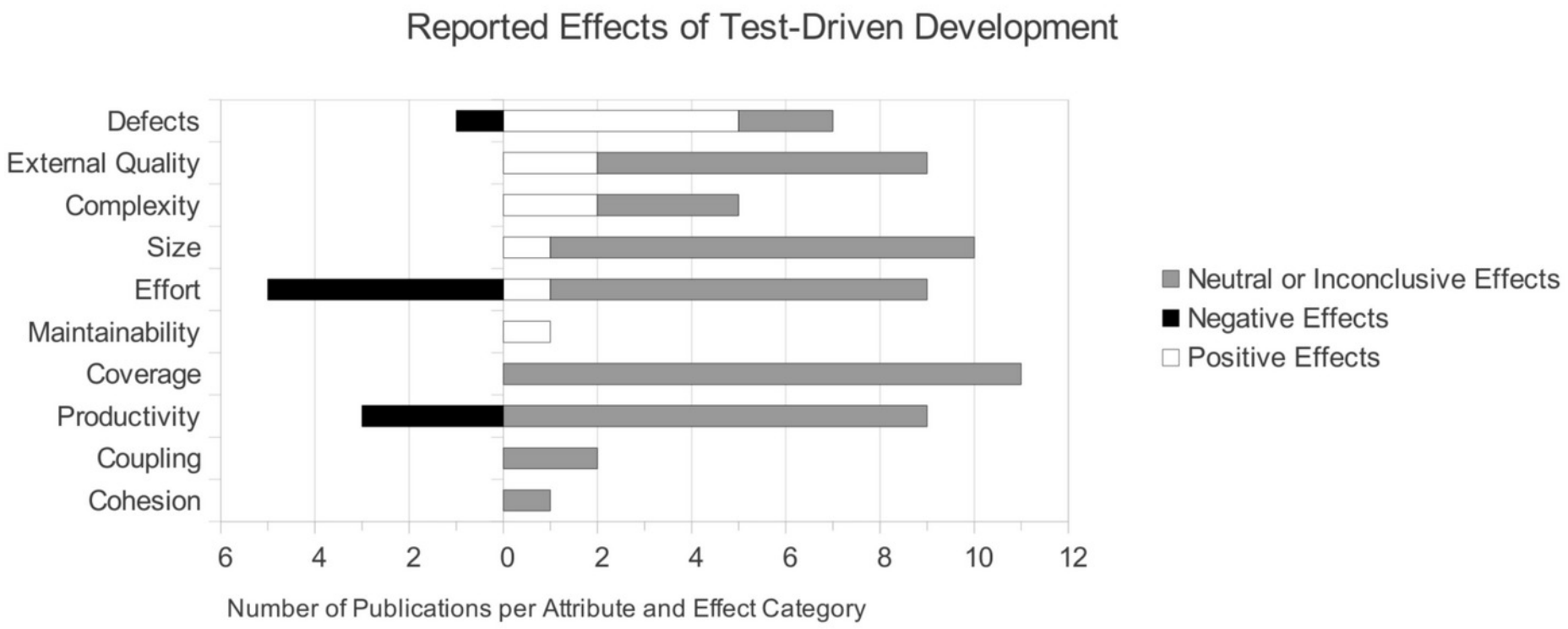
ÉVALUATION DU TEST DRIVEN DEVELOPMENT

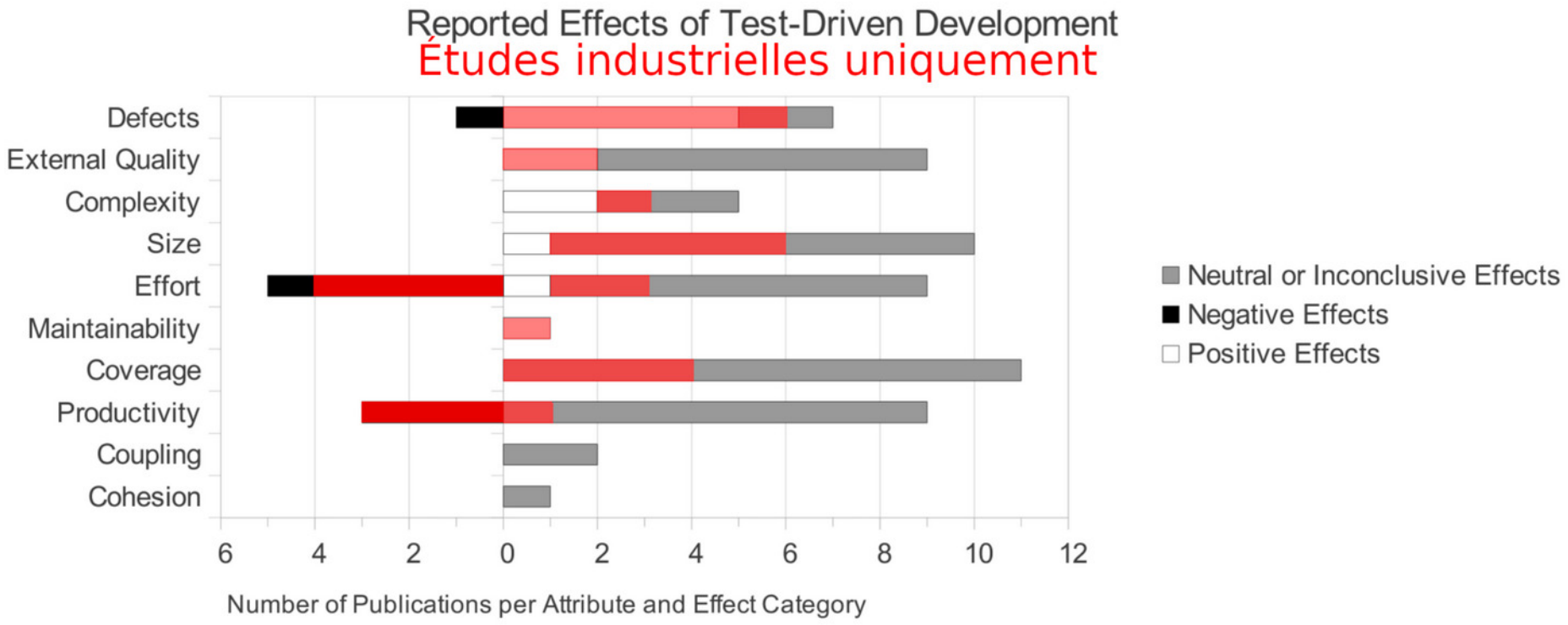
Vu les résultats précédents, que vais-je découvrir sur un sujet complexe ?

EFFECTS OF TEST-DRIVEN DEVELOPMENT: A COMPARATIVE ANALYSIS OF EMPIRICAL STUDIES

Makinen, Simo and Münch, Jürgen

2014 - Lecture Notes in Business Information Processing

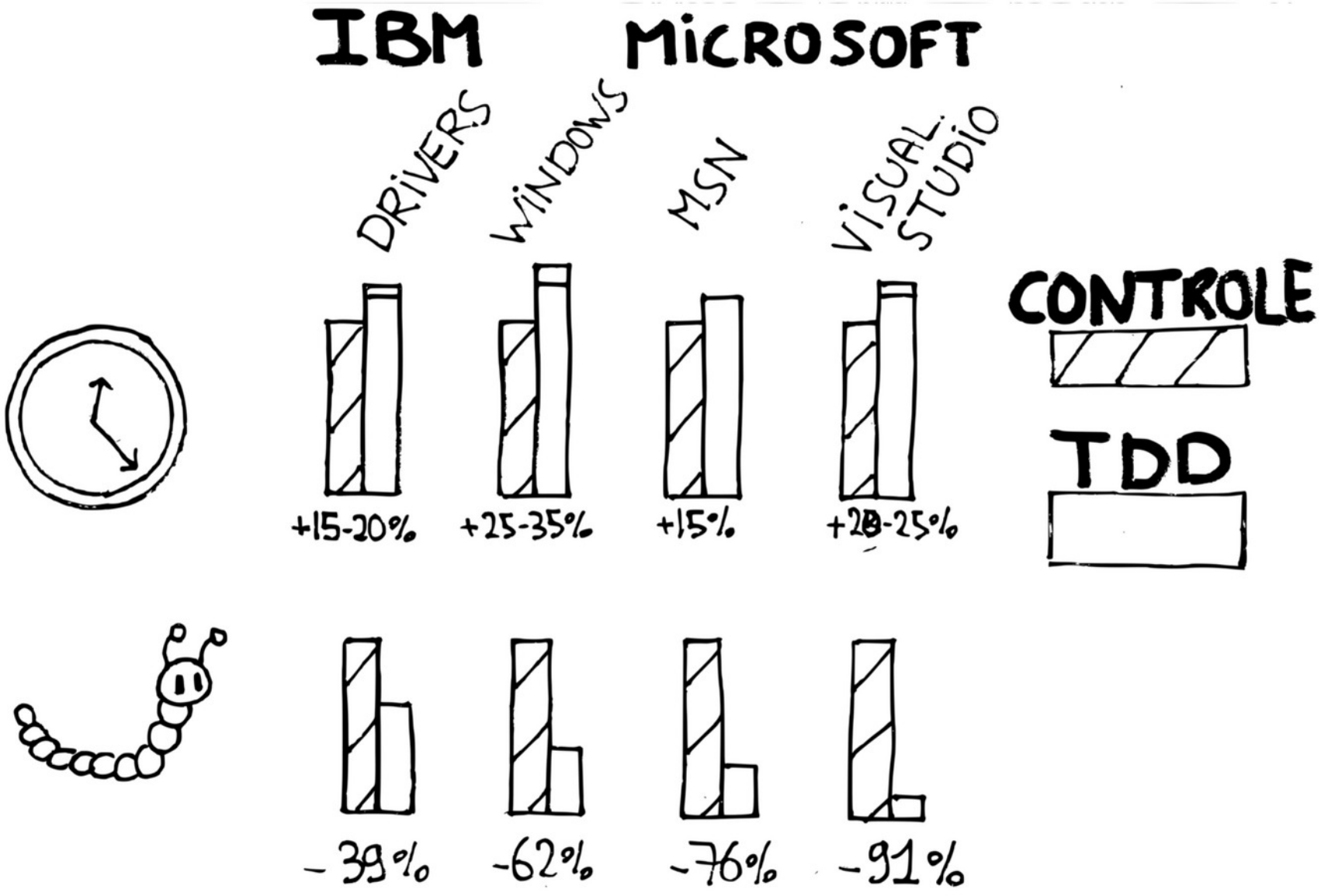

Reported Effects of Test-Driven Development
Defects
External Quality
Complexity
Size
Effort
Maintainability
Coverage
Productivity
Coupling
Cohesion
6
4
2
0
2
4
6
8
10
12
Number of Publications per Attribute and Effect Category
Neutral or Inconclusive Effects
Negative Effects
Positive Effects

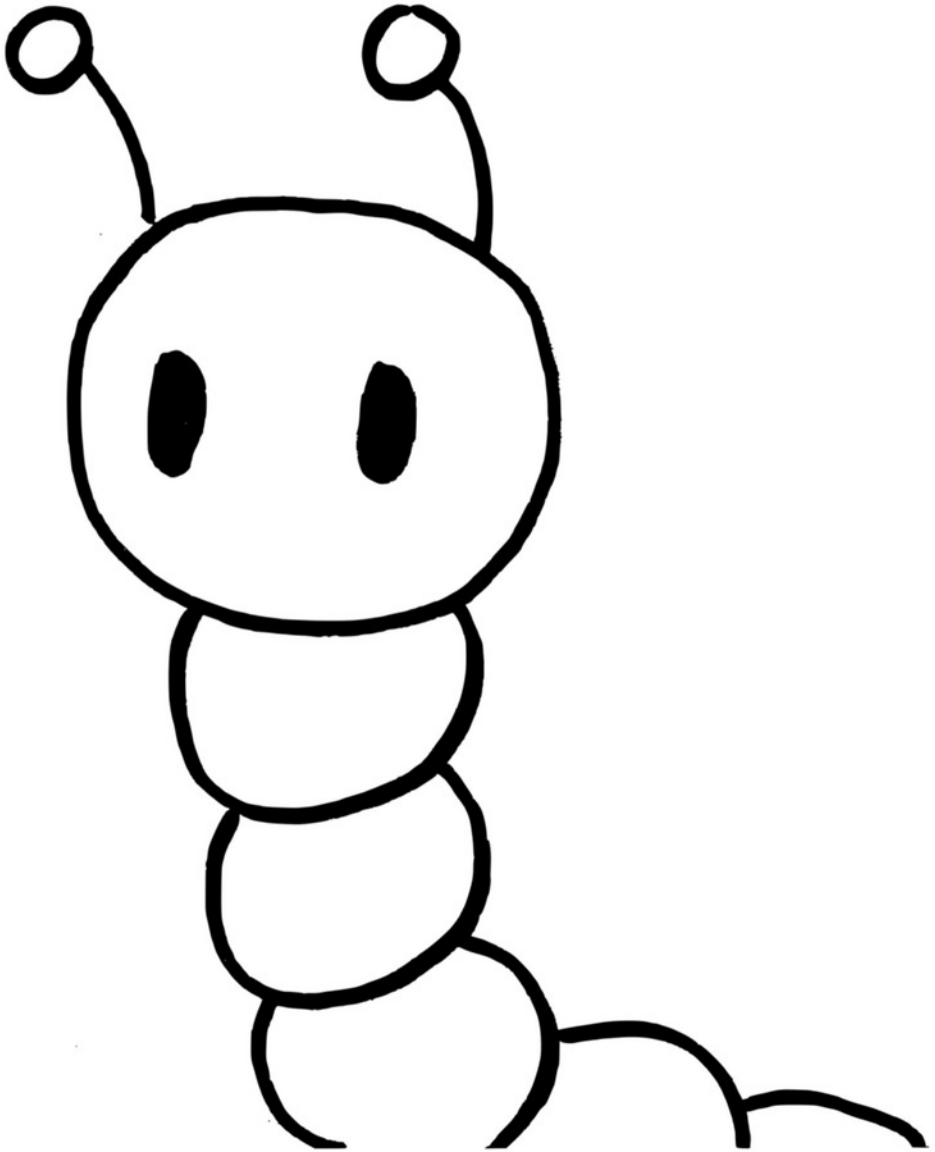

Reported Effects of Test-Driven Development
Études industrielles uniquement
Defects
External Quality
Complexity
Size
Effort
Maintainability
Coverage
Productivity
Coupling
Cohesion
6
4
2
0
2
4
6
8
10
12
Number of Publications per Attribute and Effect Category
Neutral or Inconclusive Effects
Negative Effects
Positive Effects

REALIZING QUALITY IMPROVEMENT THROUGH TEST DRIVEN DEVELOPMENT: RESULTS AND EXPERIENCES OF FOUR INDUSTRIAL TEAMS

Nagappan, Nachiappan and Maximilien, E. and Bhat, Thirumalesh and Williams, Laurie

2008 - Empirical Software Engineering

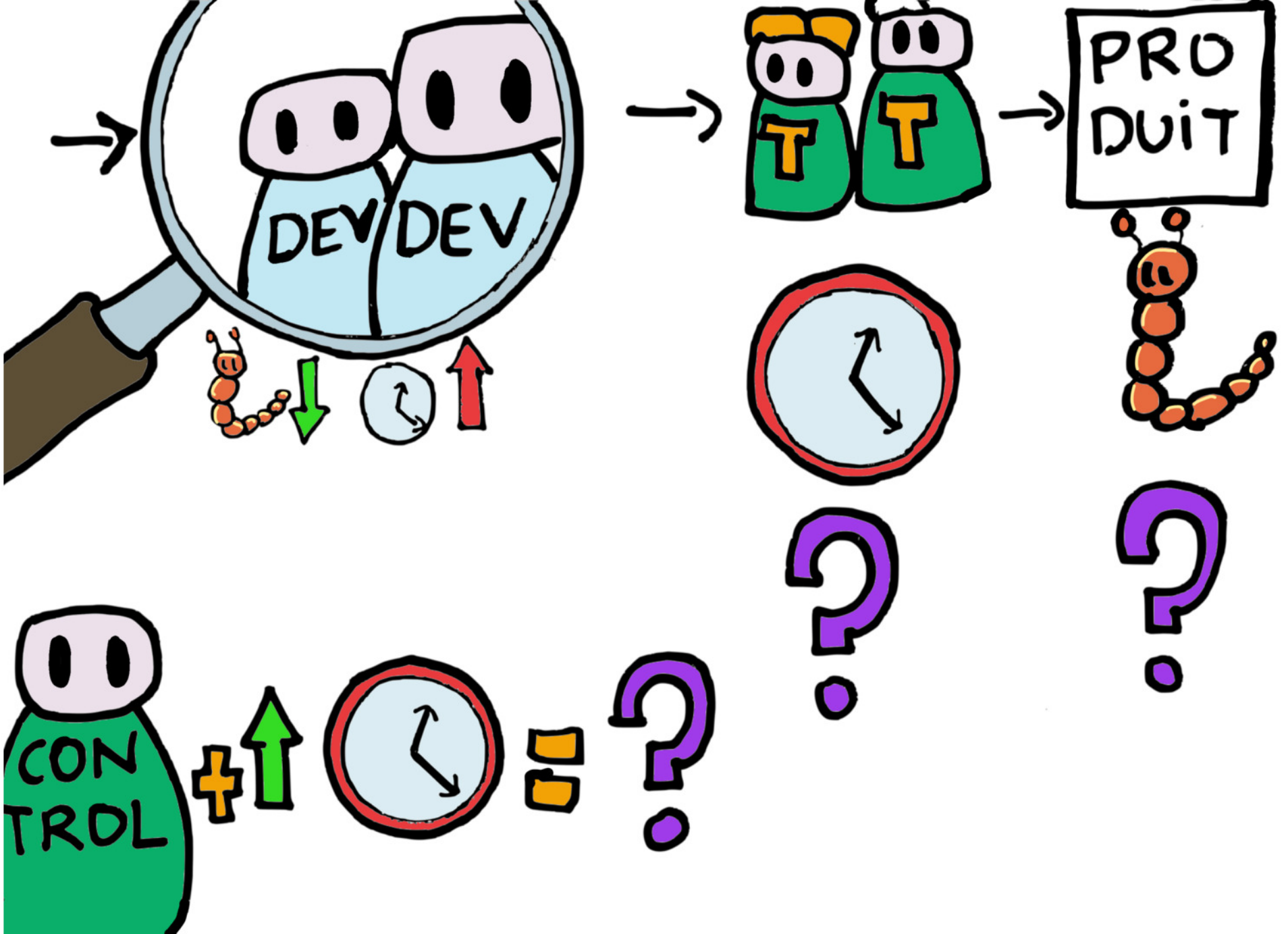

IBM
MICROSOFT
DRIVERS
WINDOWS
MSN
VISUAL STUDIO
CONTROLE
TDD
+15-20%
+25-35%
+15%
+20-25%
-39%
-62%
-76%
-91%



CONCLUSION

L’étude en question ne porte pas de conclusion sur l’efficacité de la pratique du TDD. Elle ouvre de nouvelles pistes pour continuer l’évaluation.

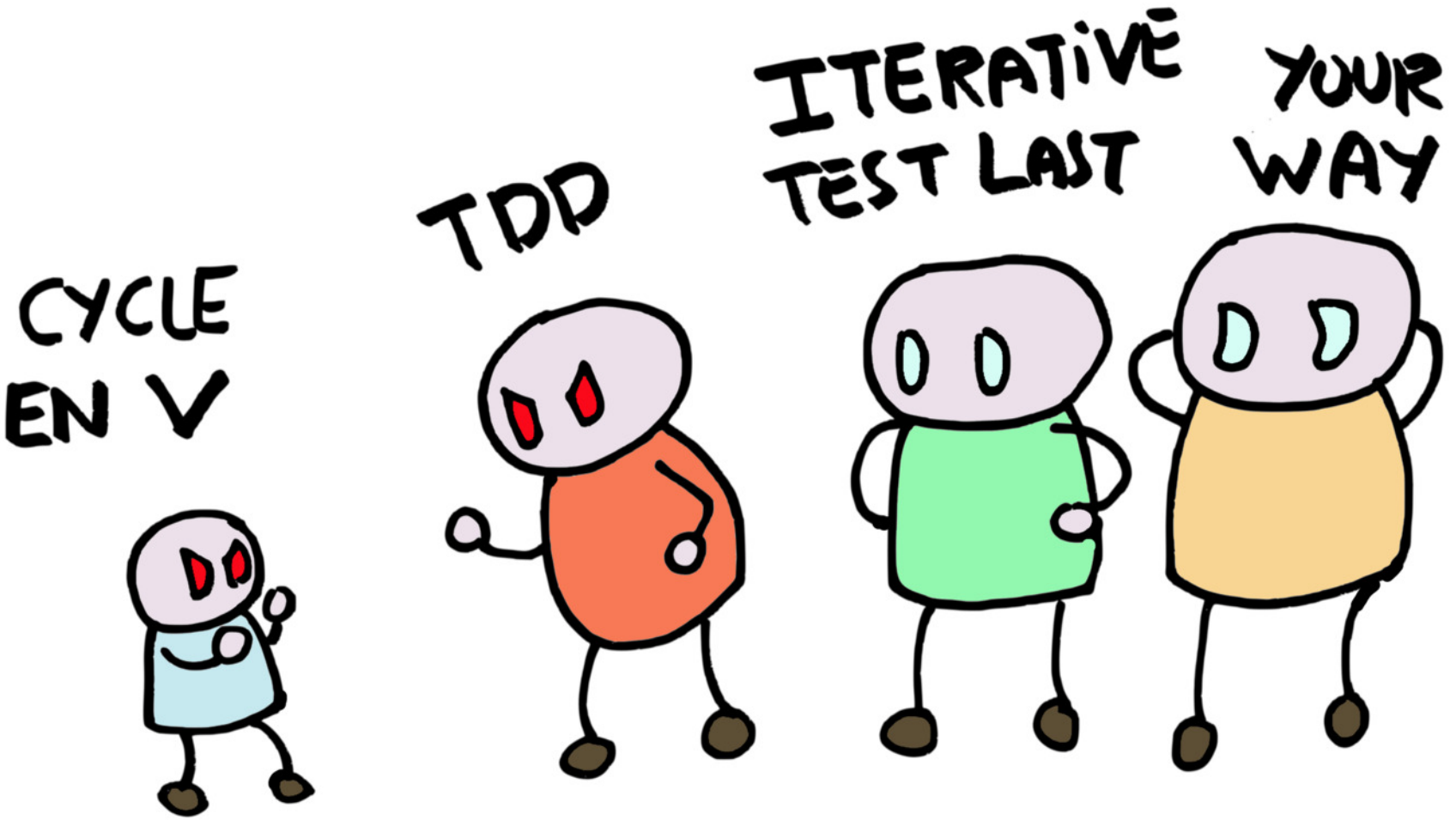
Pourquoi ?

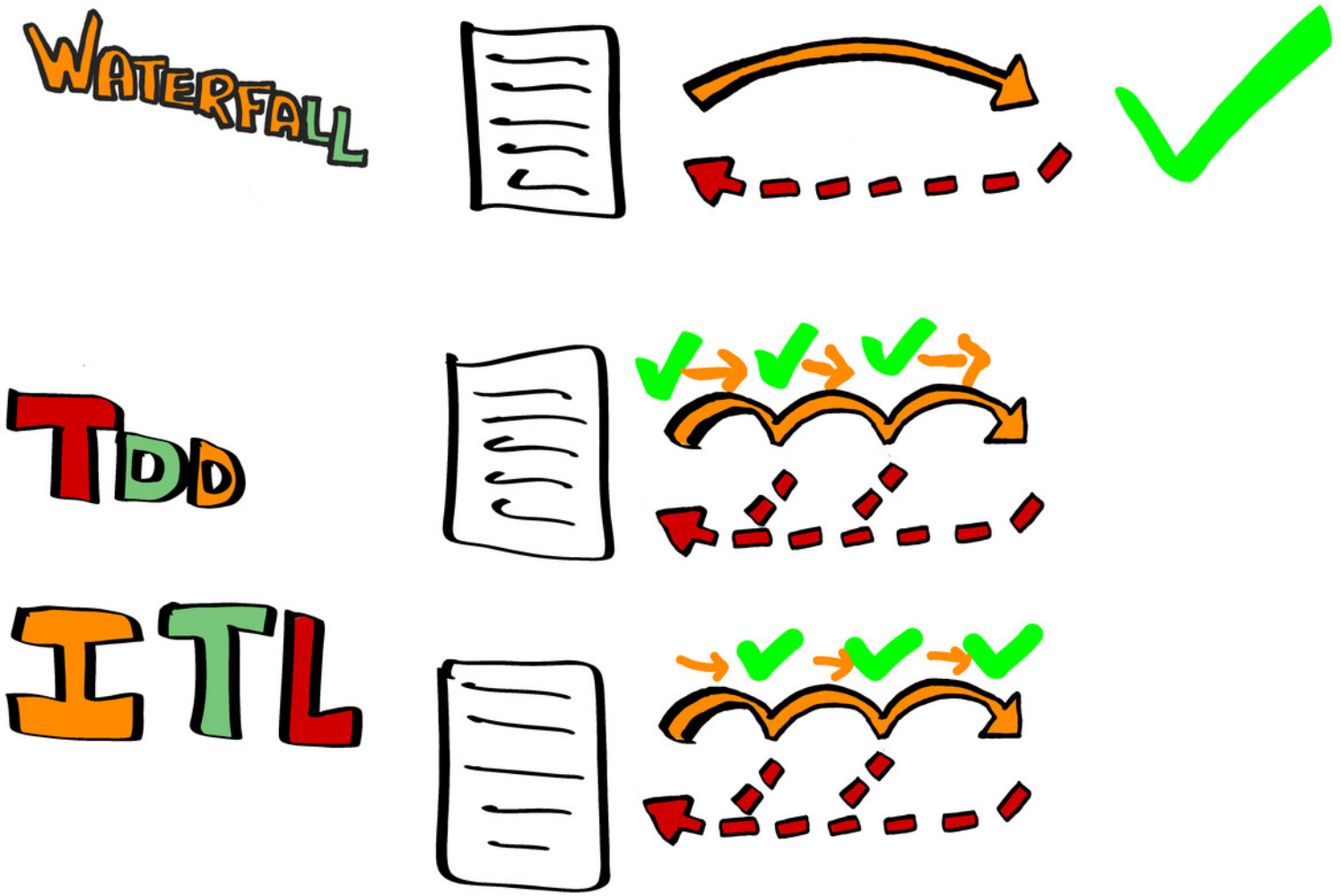

DEV
DEV
T
T
PRO
DUIT
CON
TROL

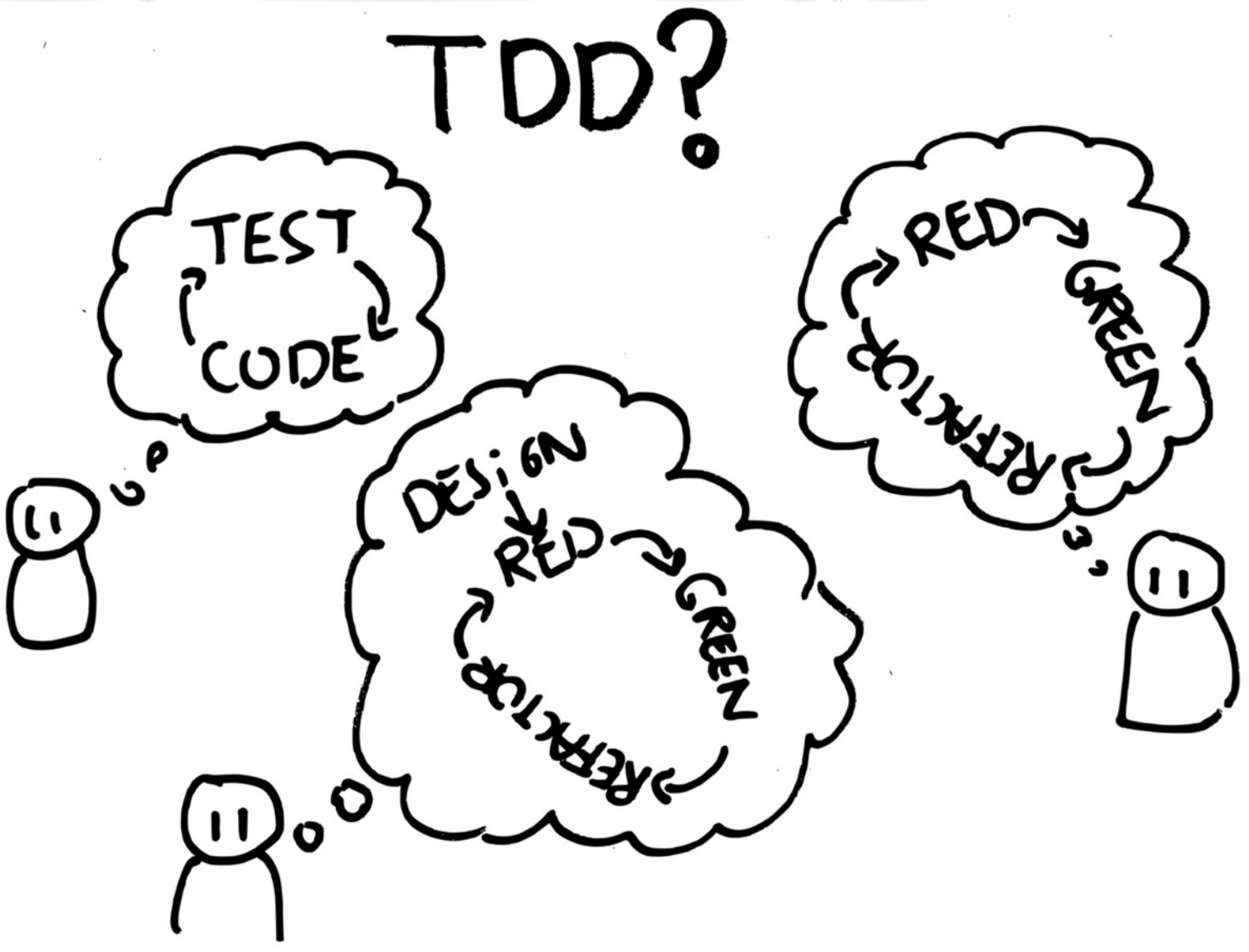
WHY RESEARCH ON TEST-DRIVEN DEVELOPMENT IS INCONCLUSIVE?

Ghafari, Mohammad and Gross, Timm and Fucci, Davide and Felderer, Michael

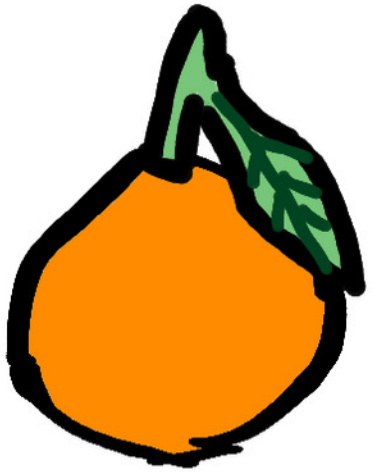
2020

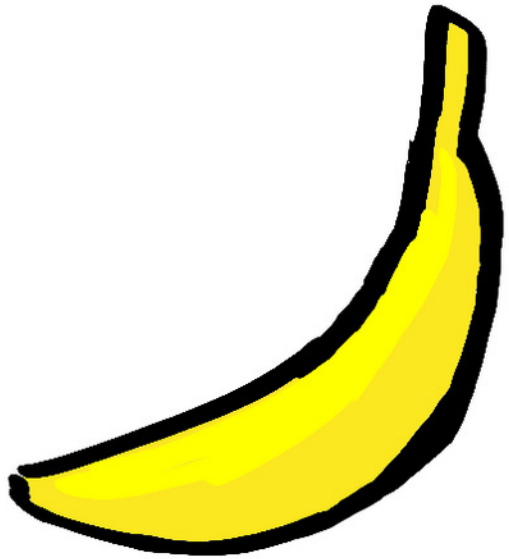

CYCLE EN V
TDD
ITERATIVE TEST LAST
YOUR WAY


WATERFALL
TDD
ITL
ITERATIVE TEST LAST


TDD?
TEST
CODE
DESIGN
RED
GREEN
REFACTOR
RED
GREEN
REFACTOR

ETAT DE L'ART



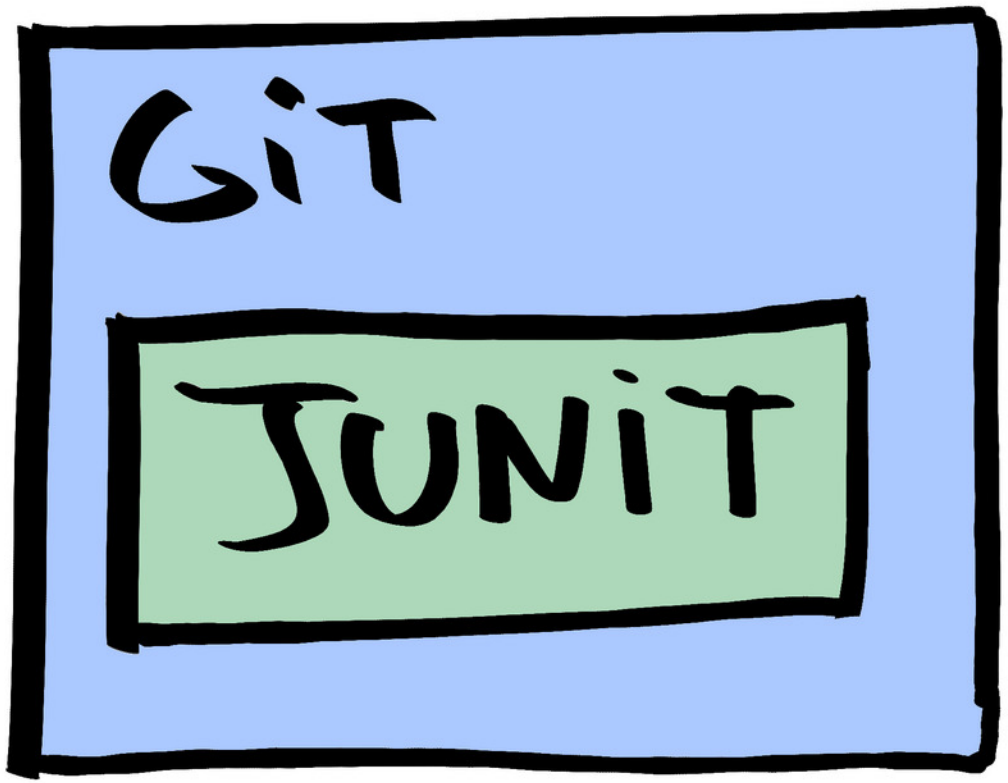


ANALYZING THE EFFECTS OF TEST DRIVEN DEVELOPMENT IN GITHUB

Borle, Neil and Feghhi, Meysam and Stroulia, Eleni and Greiner, Russ and Hindle, Abram

2018 - Empirical Software Engineering

256 572



1954

TDD

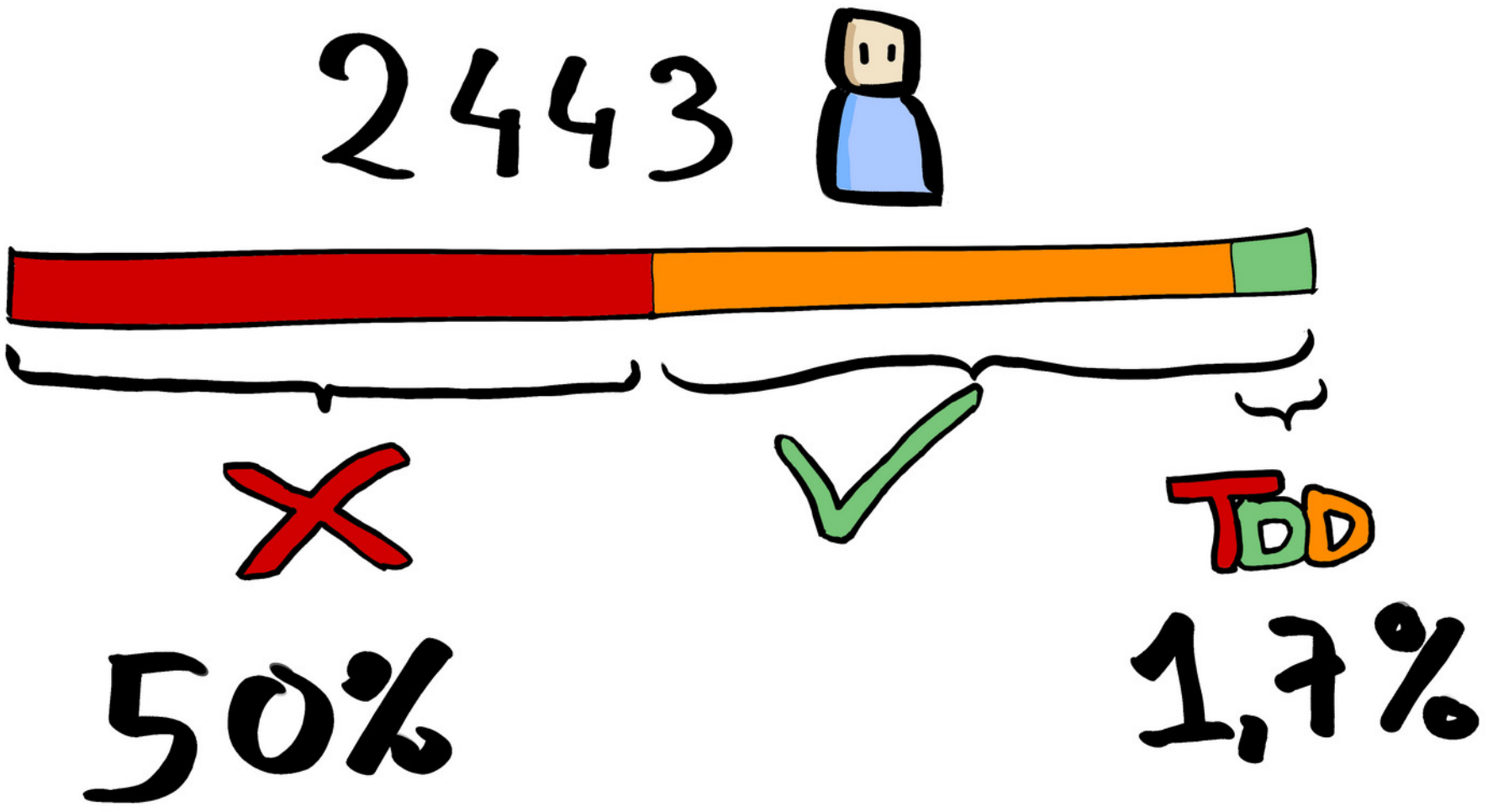
0,8 %

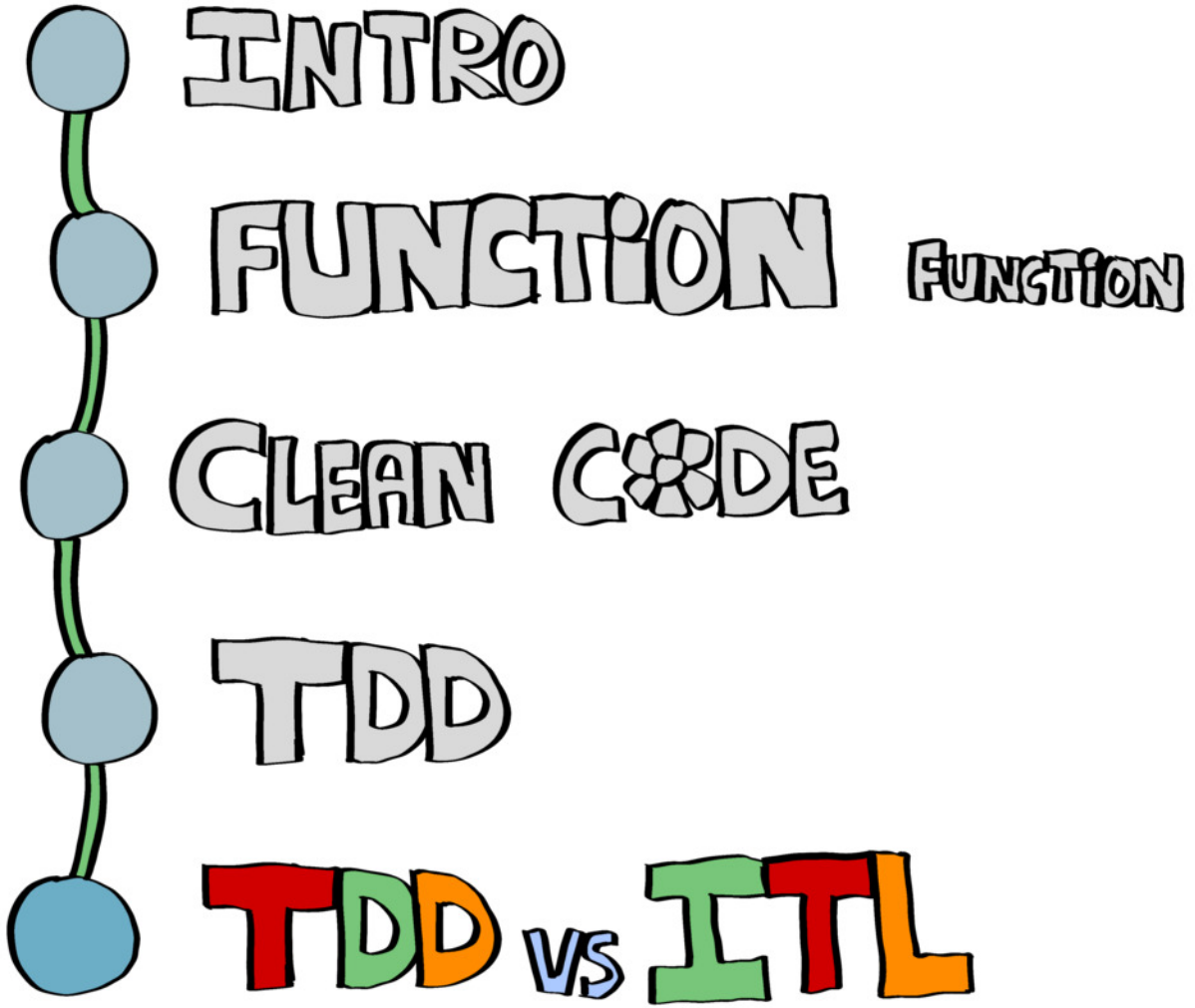
DEVELOPER TESTING IN THE IDE: PATTERNS, BELIEFS, AND BEHAVIOR

Beller, Moritz and Gousios, Georgios and Panichella, Annibale and Proksch, Sebastian and Amann, Sven and Zaidman, Andy

2017 - IEEE Transactions on Software Engineering

QUI FAIT DU TDD ?



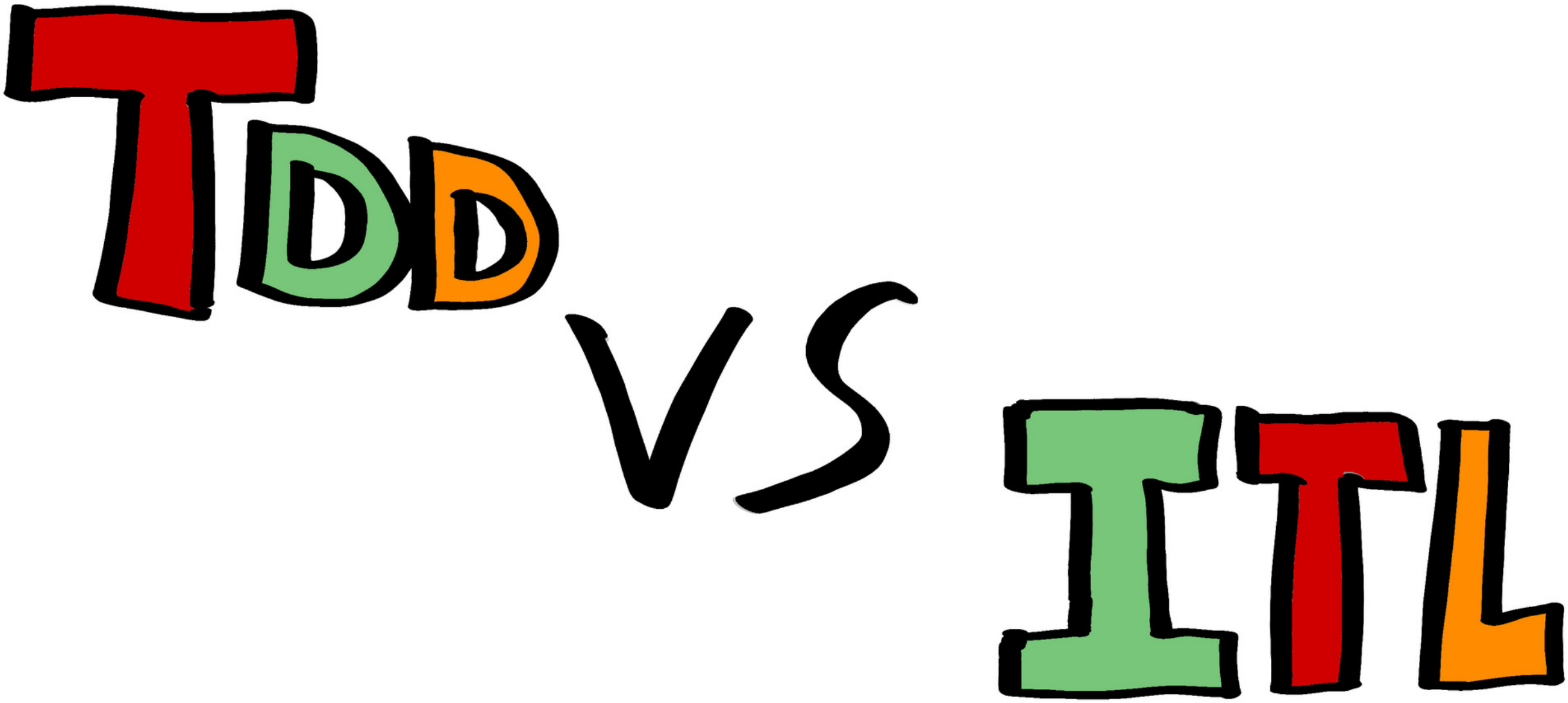

INTRO
FUNCTION
FUNCTION
CLEAN CODE
TDD
TDD VS ITL

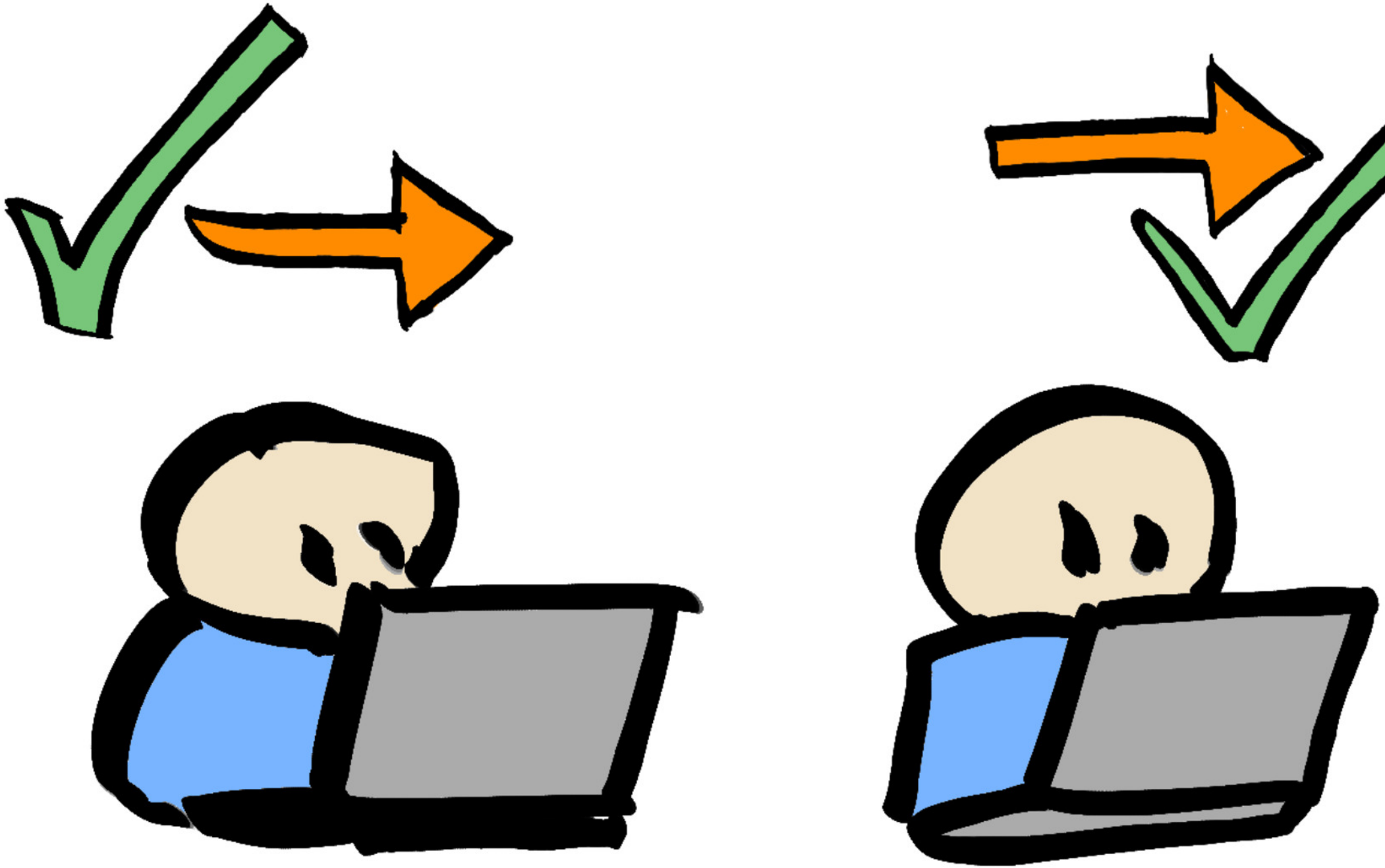
A FAMILY OF EXPERIMENTS ON TDD

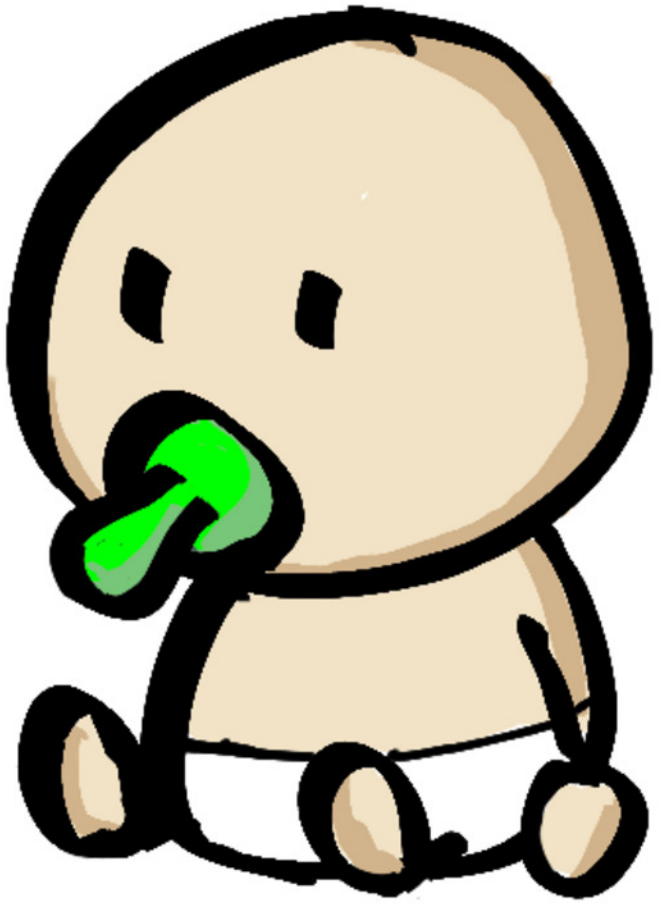
A FAMILY OF EXPERIMENTS ON TEST-DRIVEN DEVELOPMENT

Santos, Vegas, Dieste Tubío, Uyaguari, Tosun, Fucci, Turhan, Scanniello, Romano, Karac, Kuhrmann, Mandić, Ramač, Pfahl, Engblom, Kyykka, Rungi, Palomeque, Spisak, Juristo, Natalia

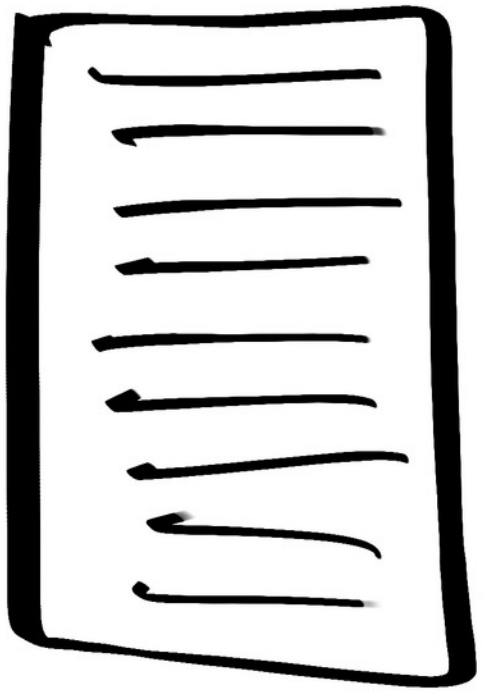
2021 - Empirical Software Engineering

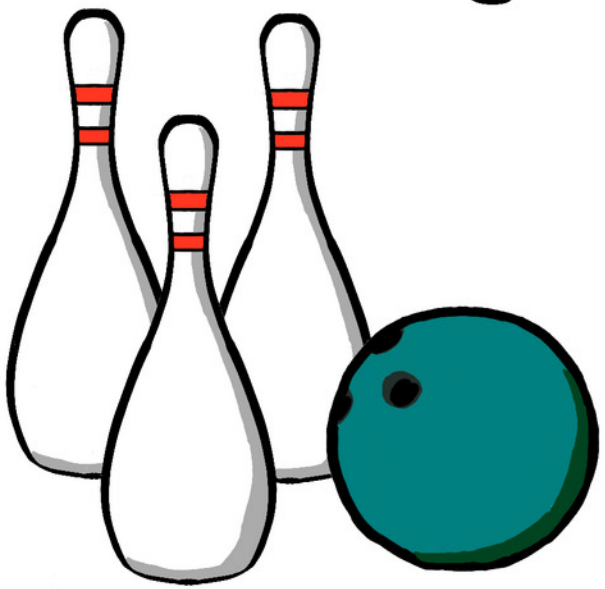

TDD
VS
ITL



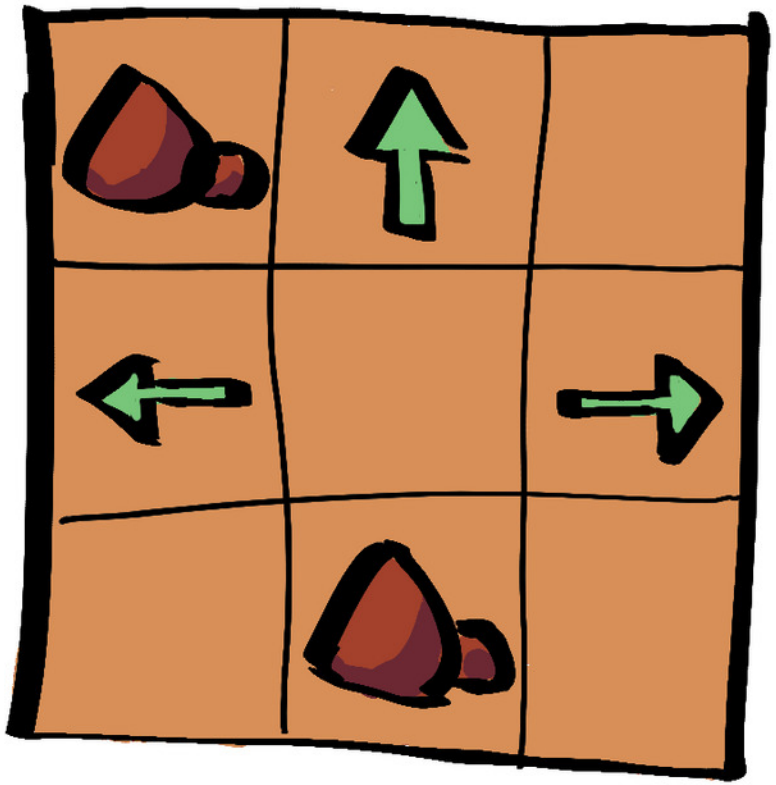


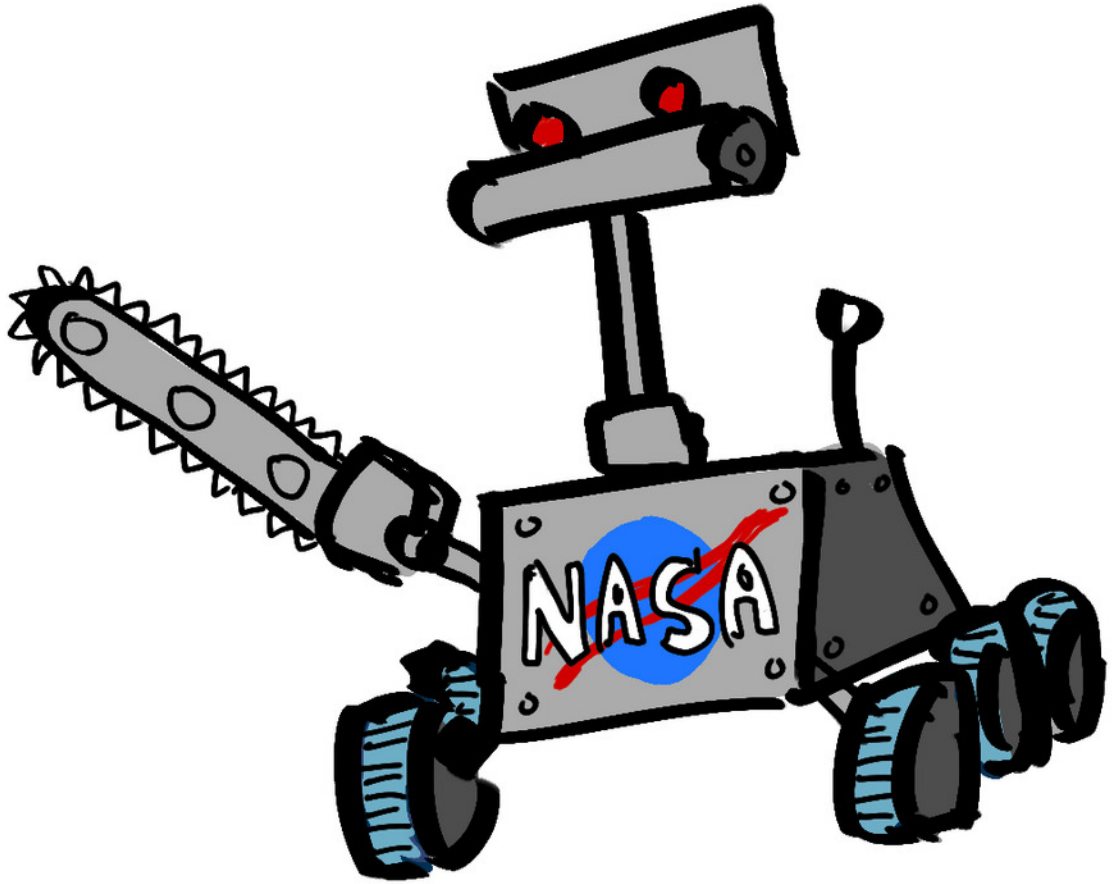
Bowling Kata



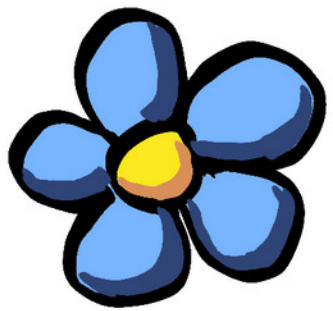


Mars Rover





QUALITÉ

☒ INTERNE 

☑ EXTERNE 

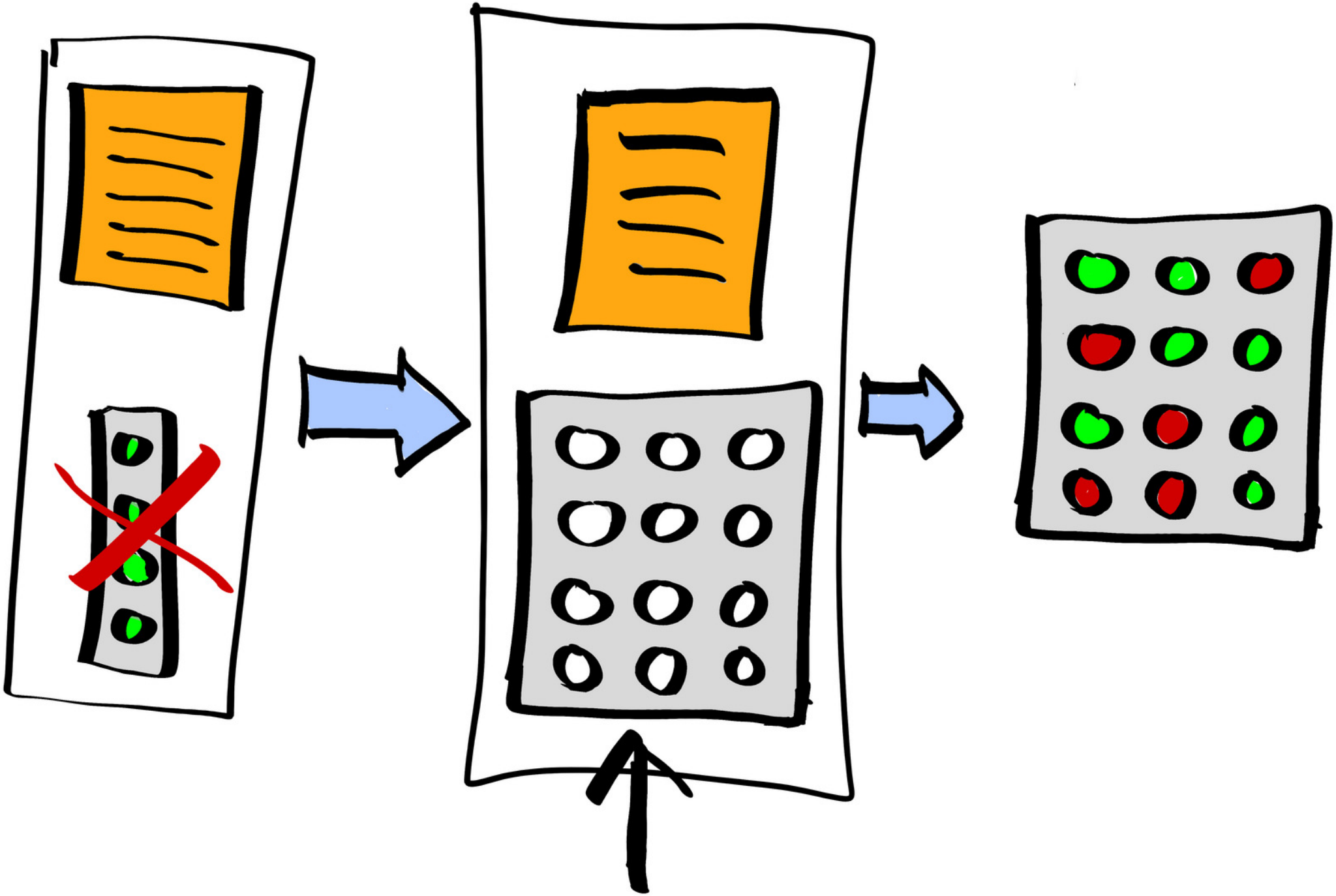


Fig. 4 Forest plot: TDD vs. ITL mean effects.

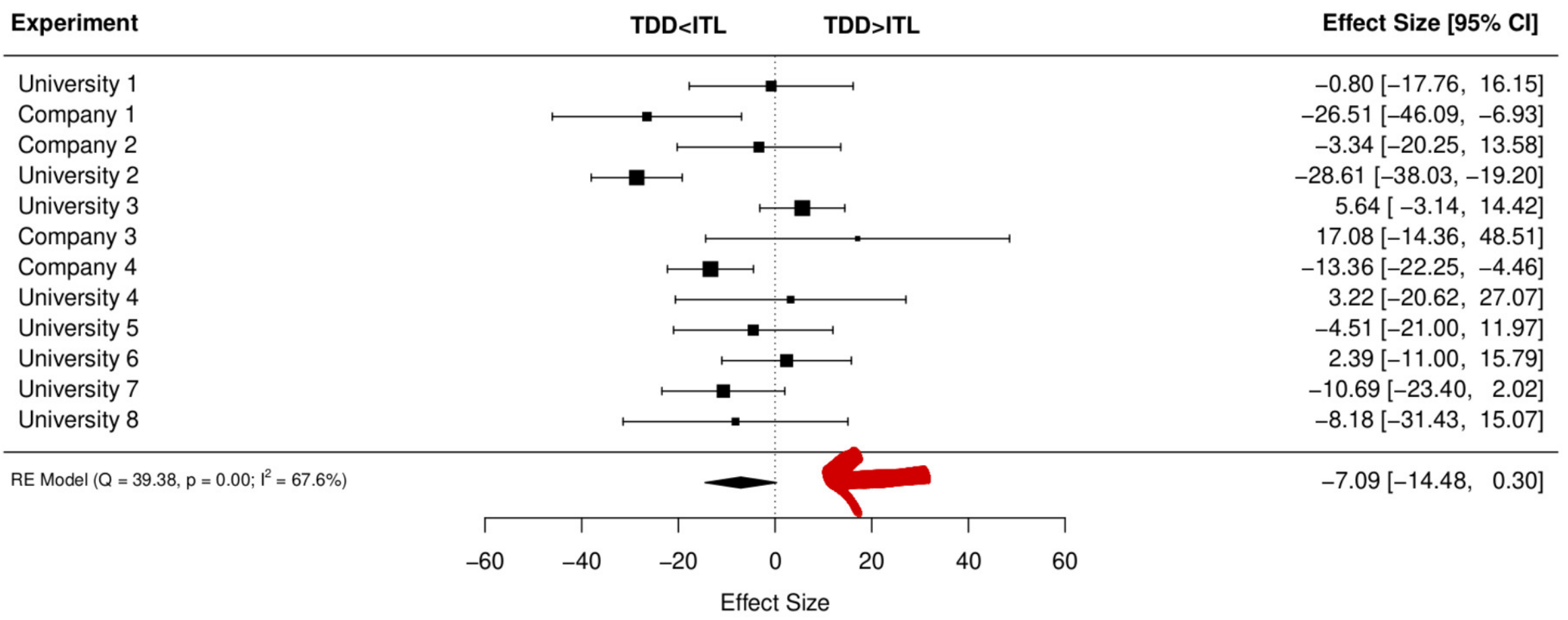


Table 9 Marginal means for quality: task by treatment.

Task	Treatment	Estimate
MR	ITL	29.36
	TDD	24.95
BSK	ITL	49.22
	TDD	46.46

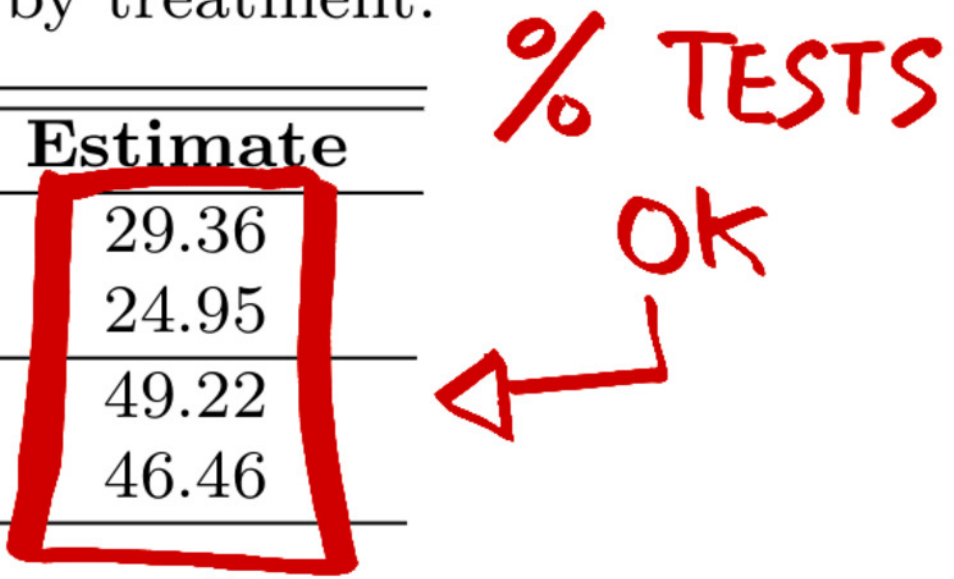


Table 13 Marginal means for quality: programming environment by treatment.

Treatment	Group	Estimate
ITL	Java	43.83
	C#/C++	41.25
TDD	Java	35.77
	C#/C++	37.55

Fig. 5 Forest plot: students vs. professionals

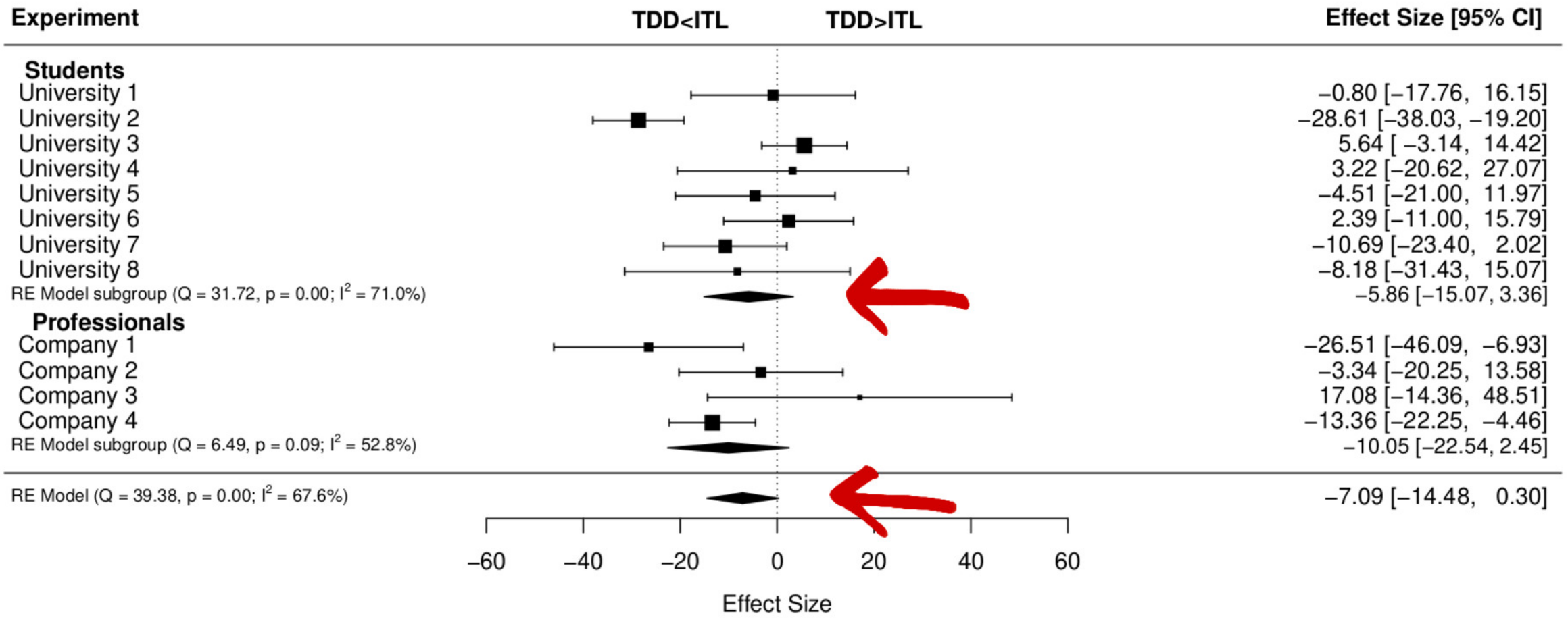


Table 11 Marginal means for quality: participant type by treatment.

Treatment	Group	Estimate
ITL	Students	40.47
	Professionals	49.53
TDD	Students	35.61
	Professionals	38.26

Fig. 10 Profile plot: mean difference in quality (TDD-ITL) for professionals per experience level in Questionnaire 2.

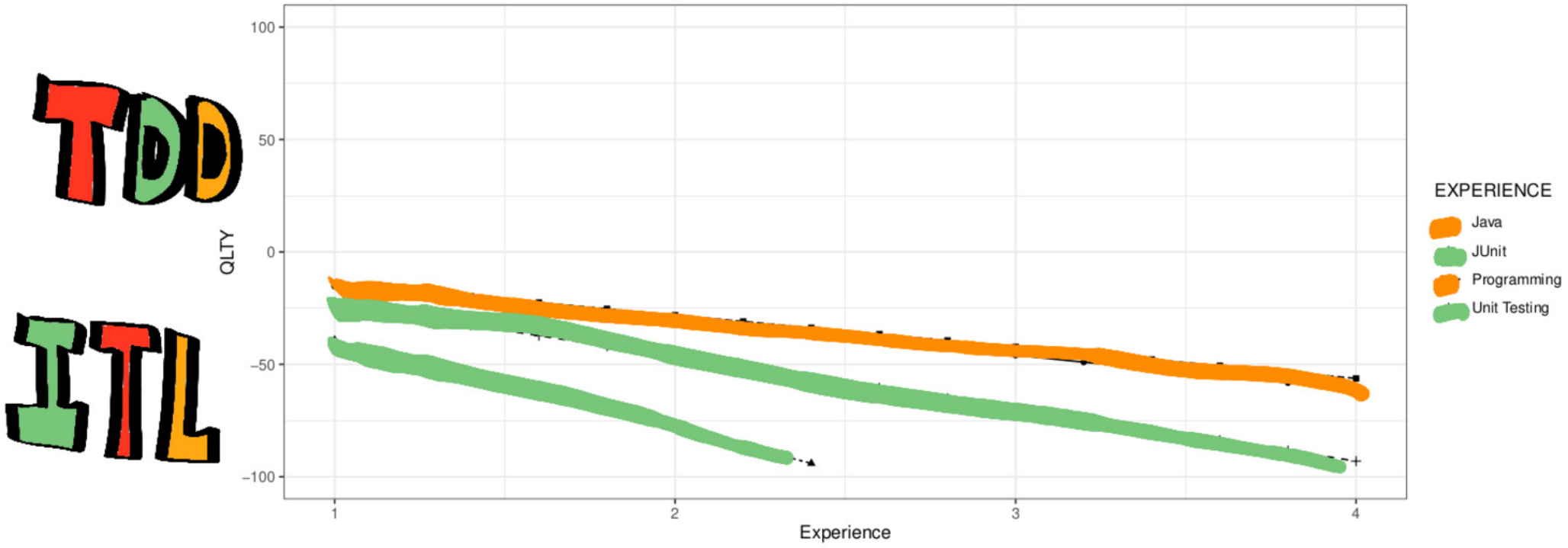
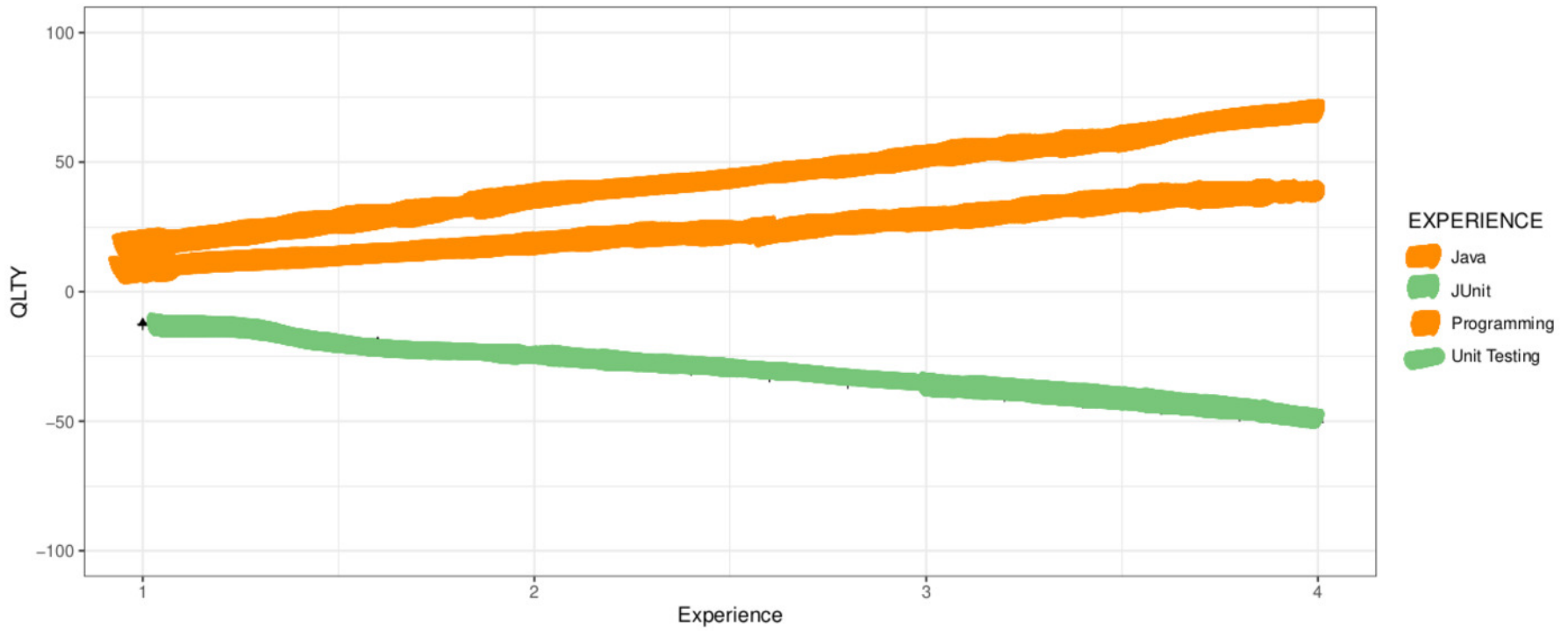
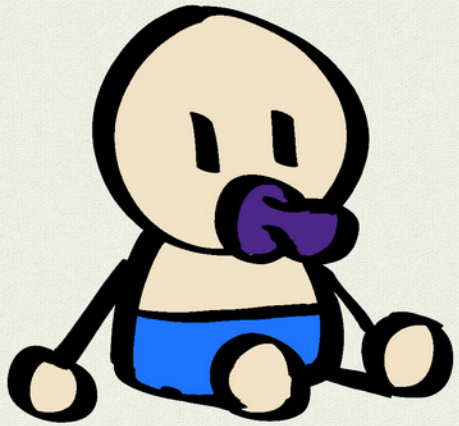


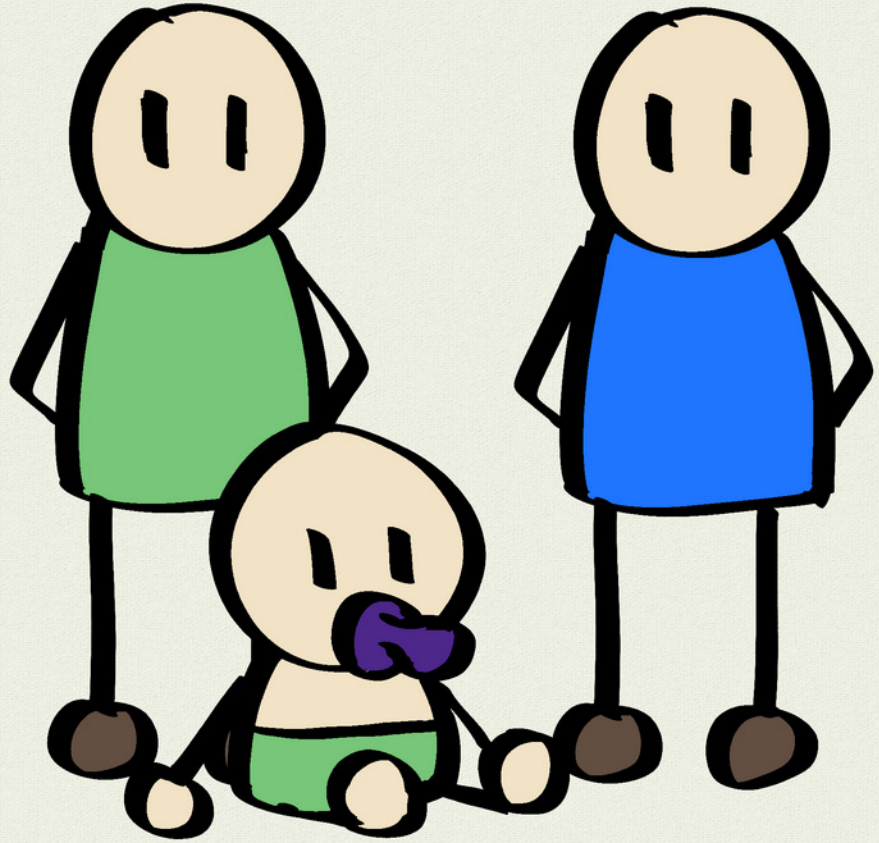
Fig. 11 Profile plot: mean difference in quality (TDD-ITL) for students per experience level in Questionnaire 1.

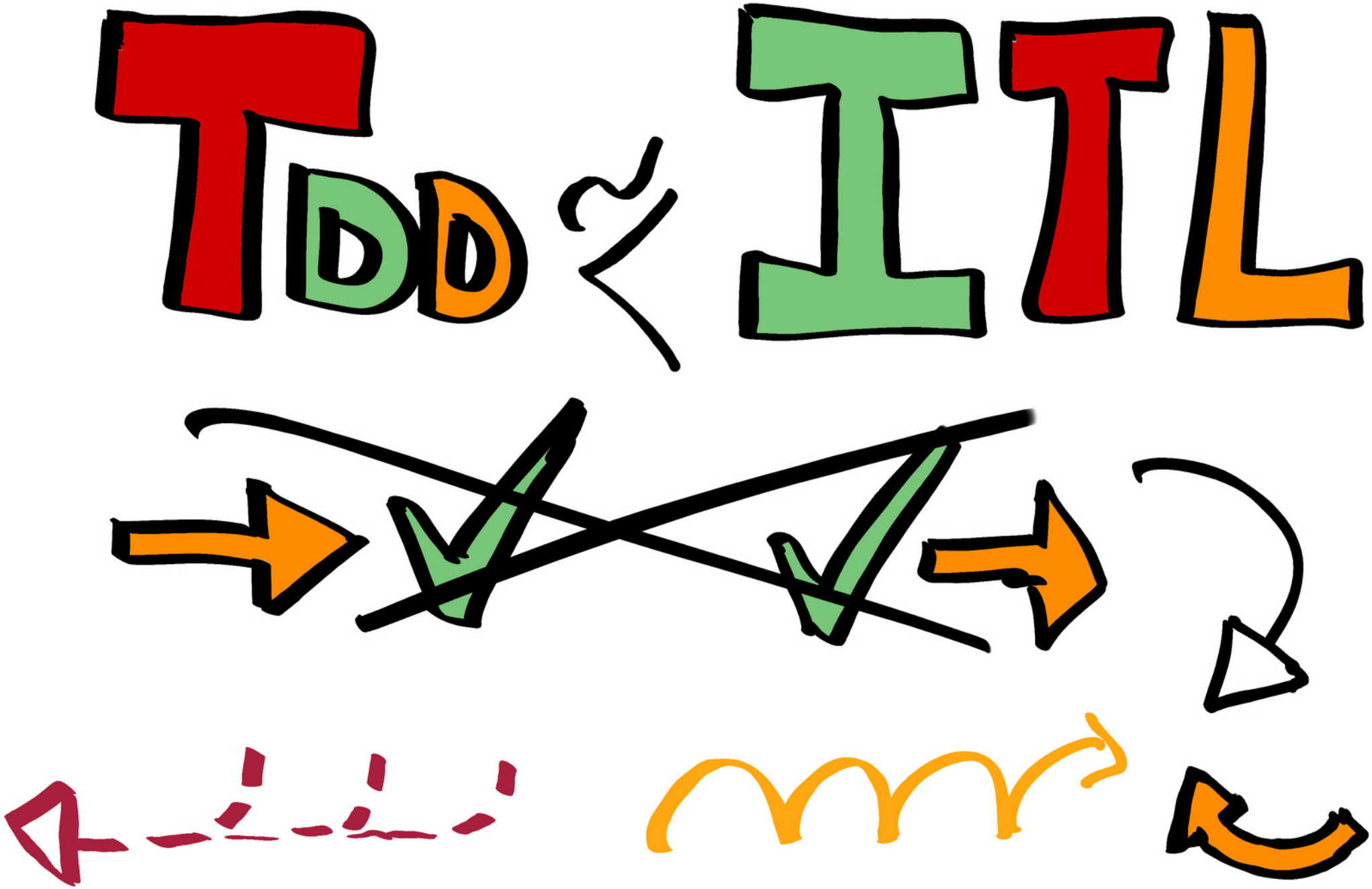


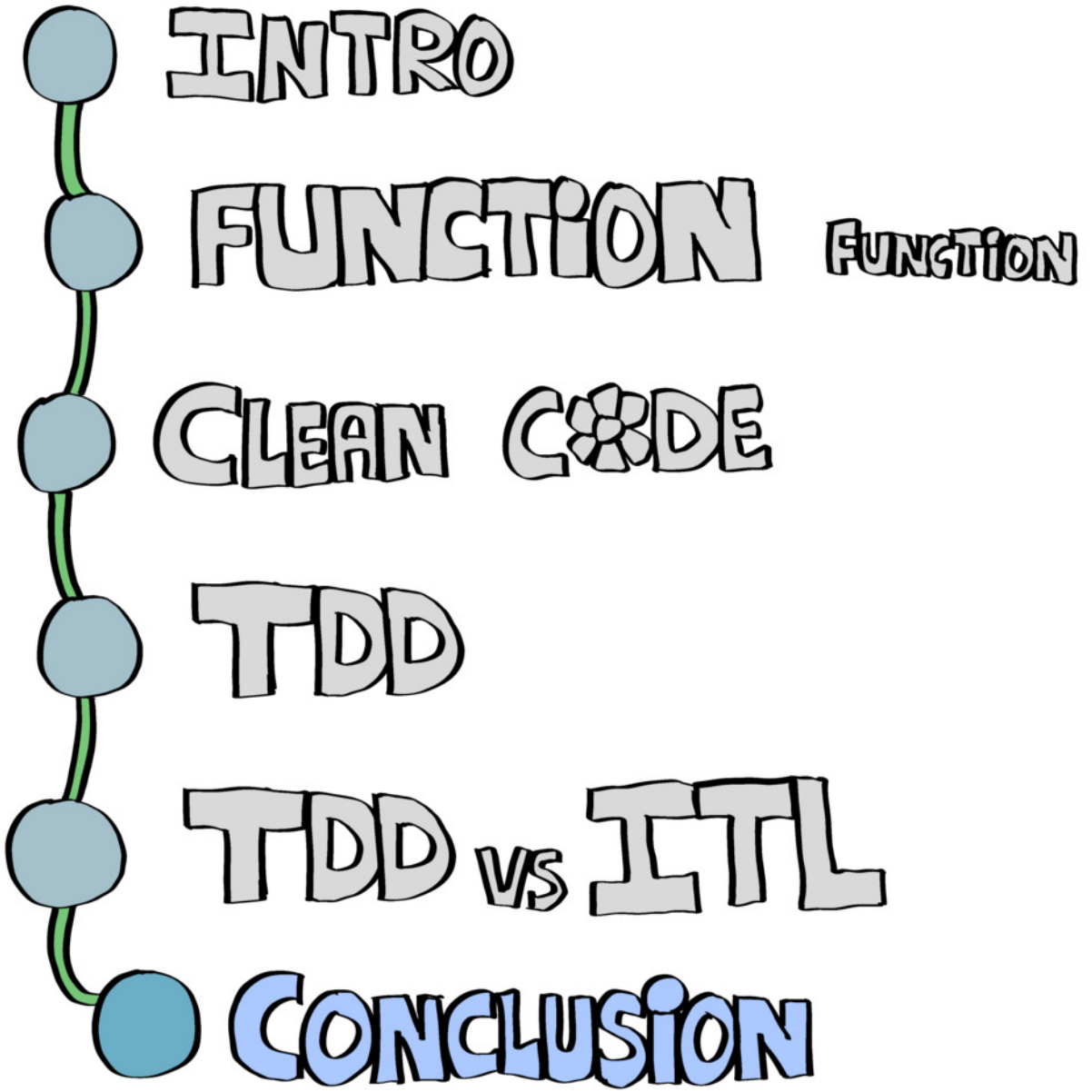
TDD

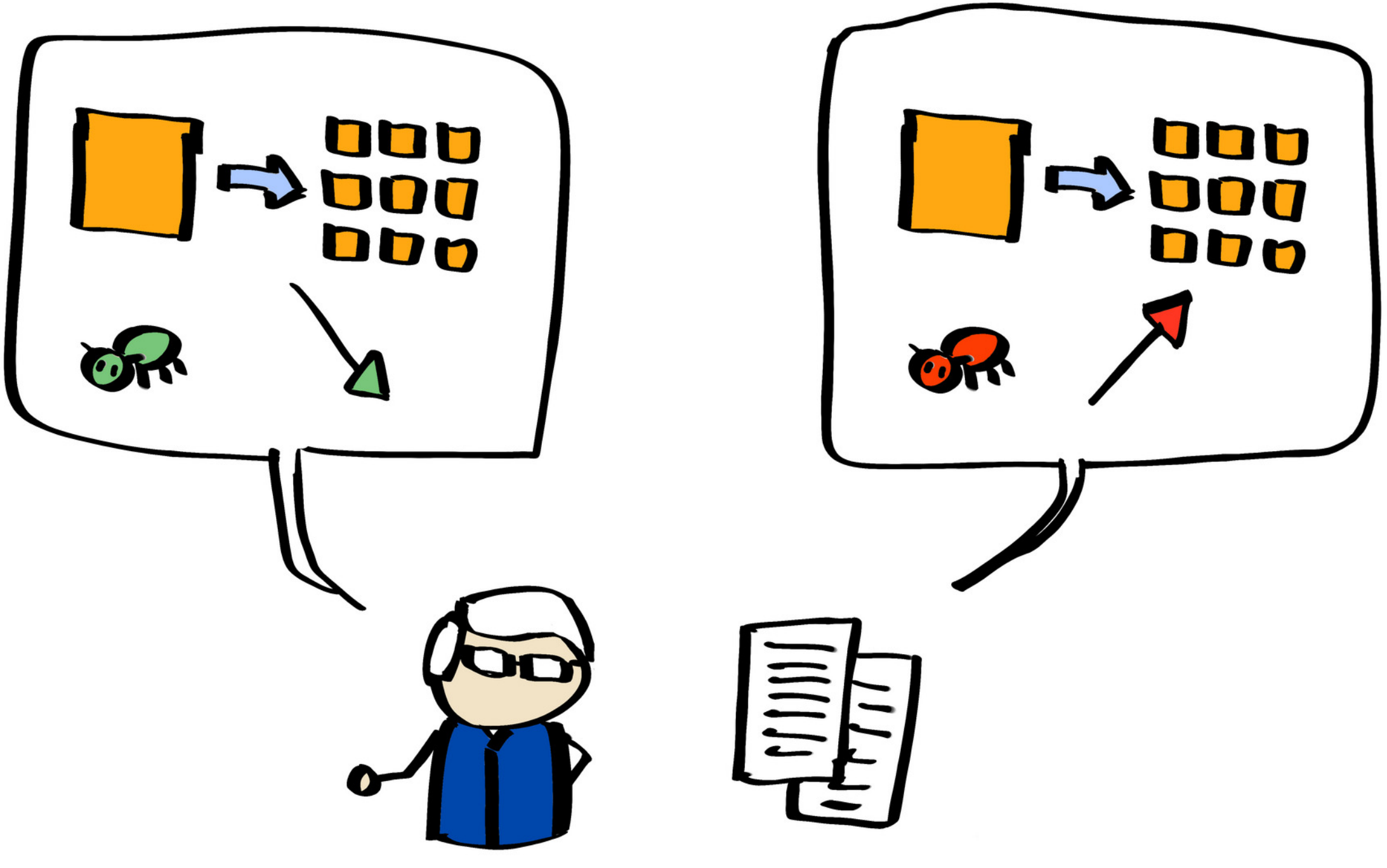


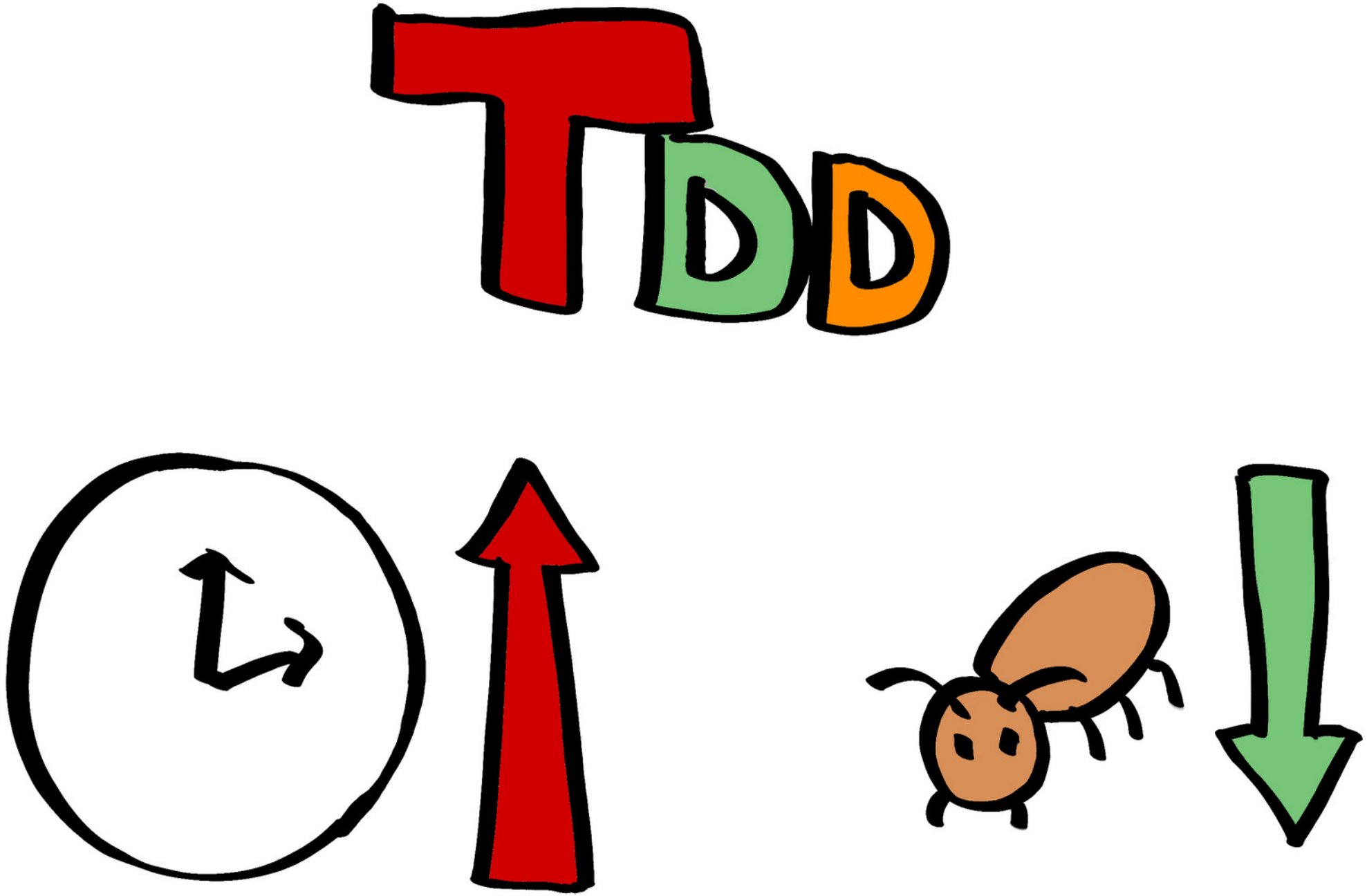
ITL




TDD
ITL


INTRO
FUNCTION
FUNCTION
CLEAN CODE
TDD
TDD VS ITL
CONCLUSION

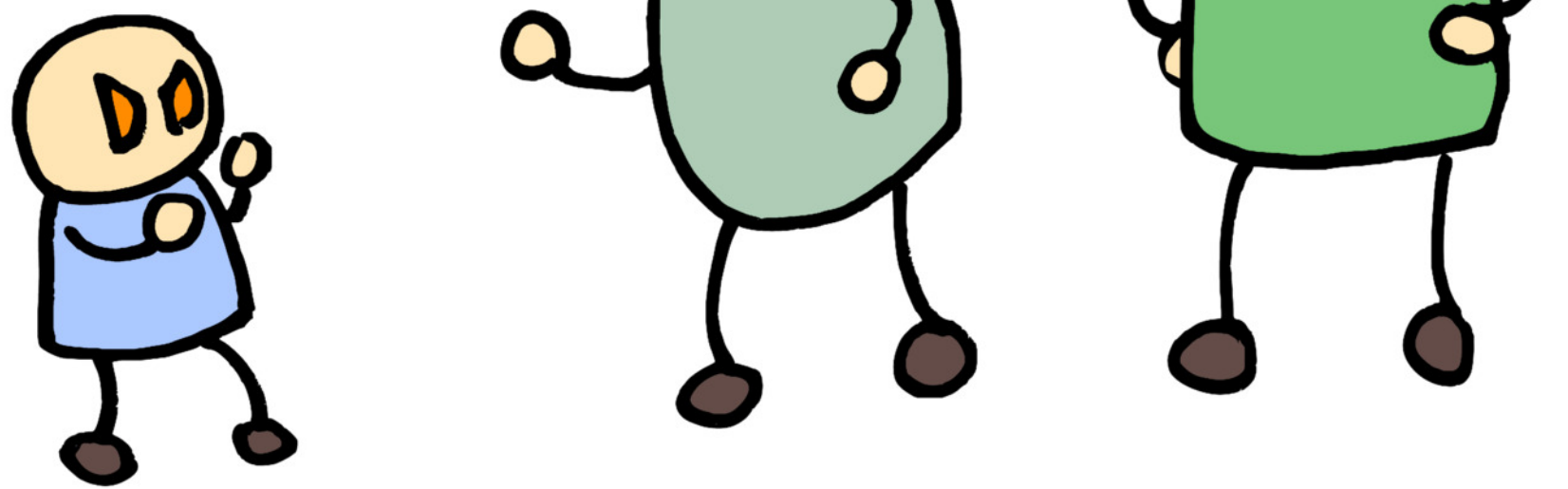


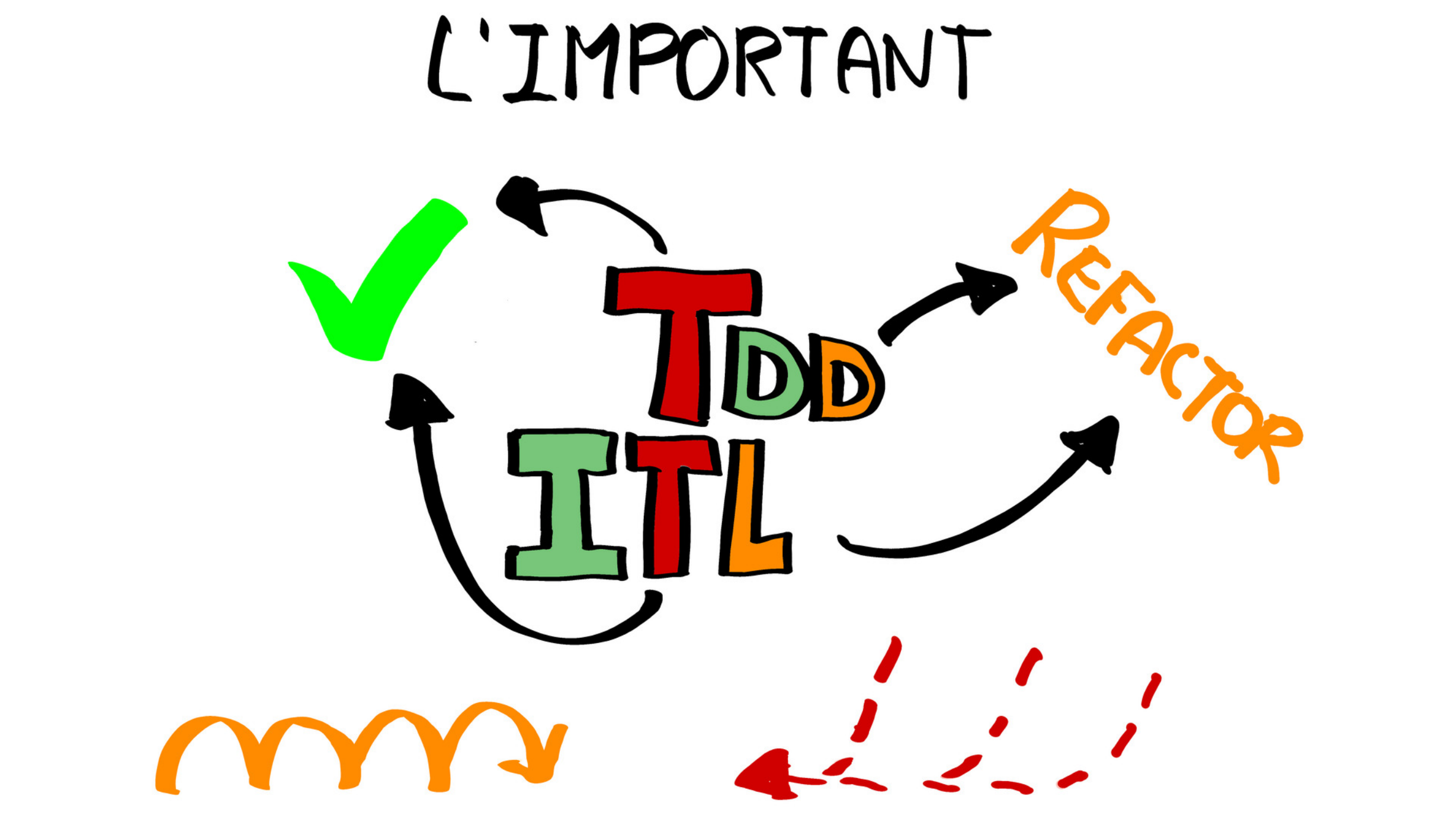

TDD

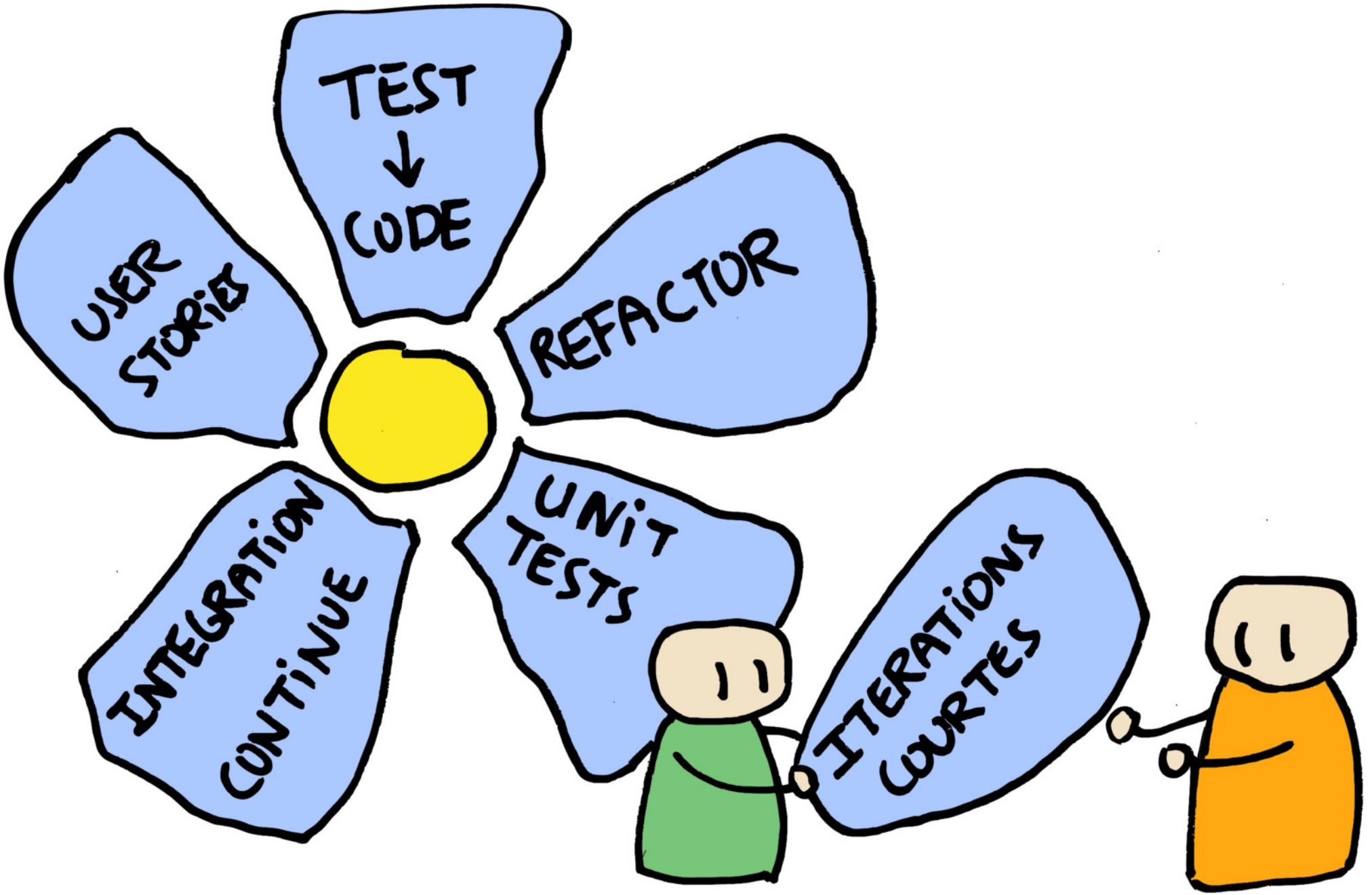
CYCLE EN V

TDD

ITERATIVE TEST LAST



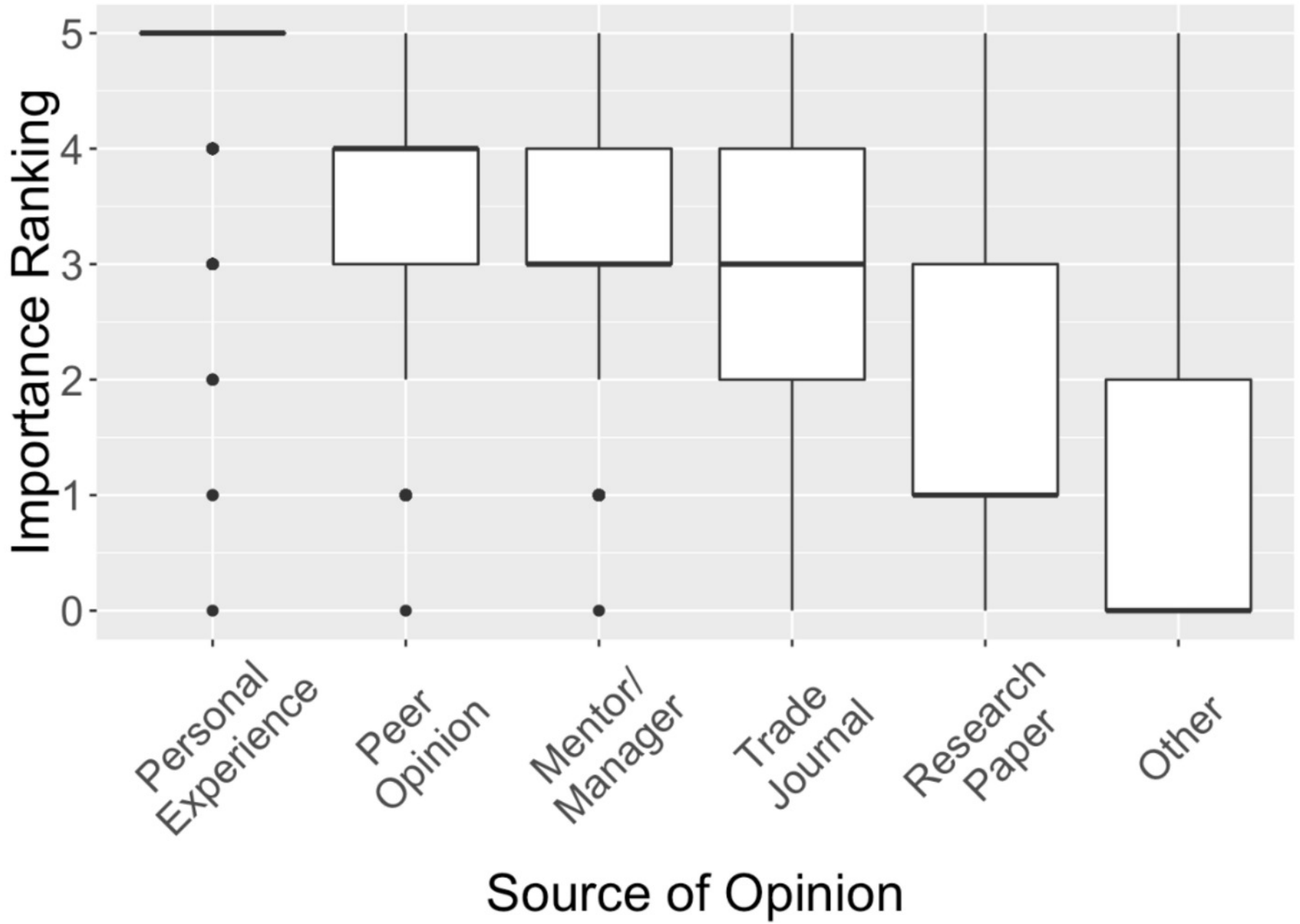

L'IMPORTANT
TDD
ITL
REFACTOR

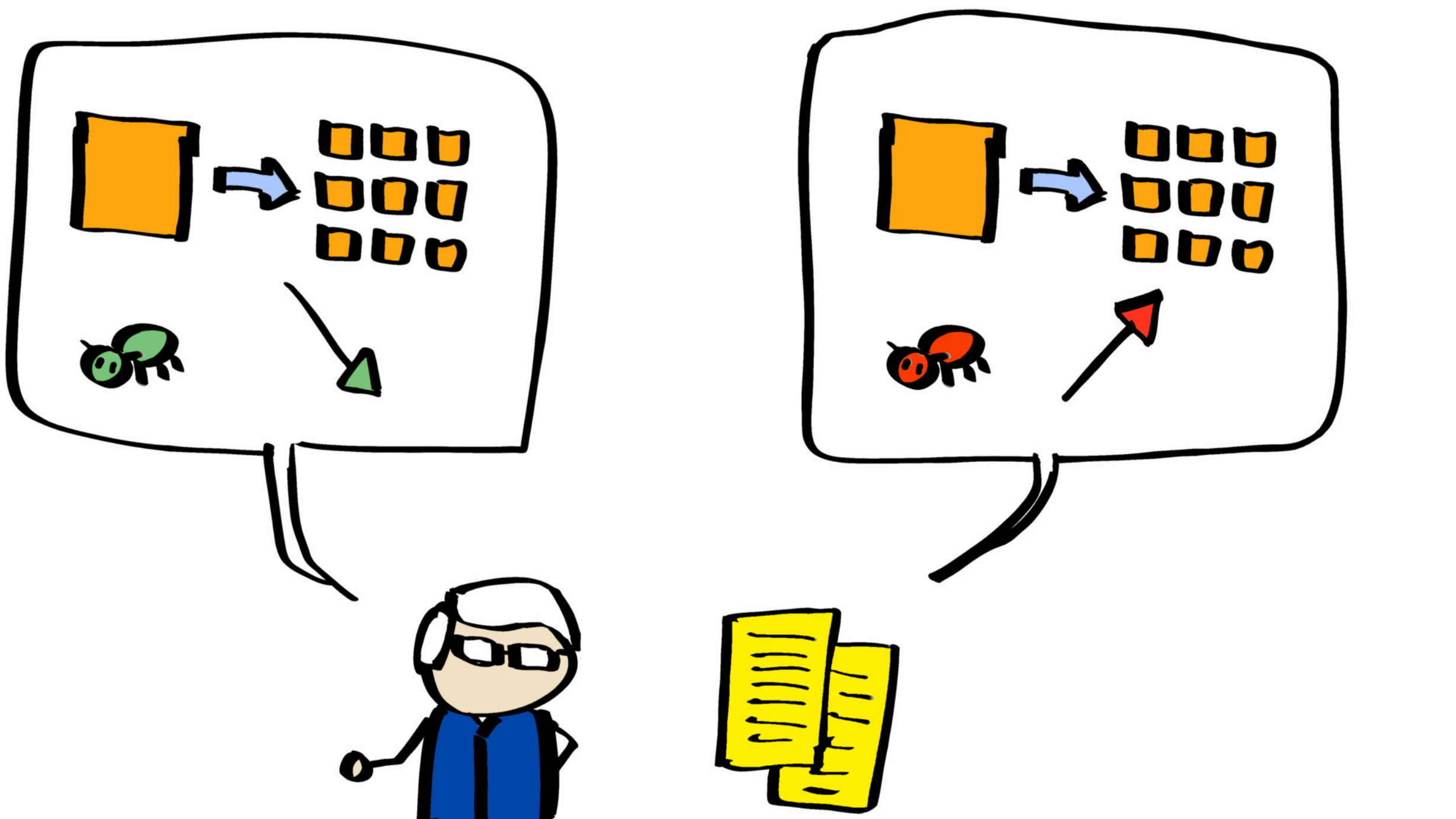

TEST
↓
CODE
USER STORIES
REFACTOR
INTEGRATION CONTINUE
UNIT TESTS
ITERATIONS COURTES

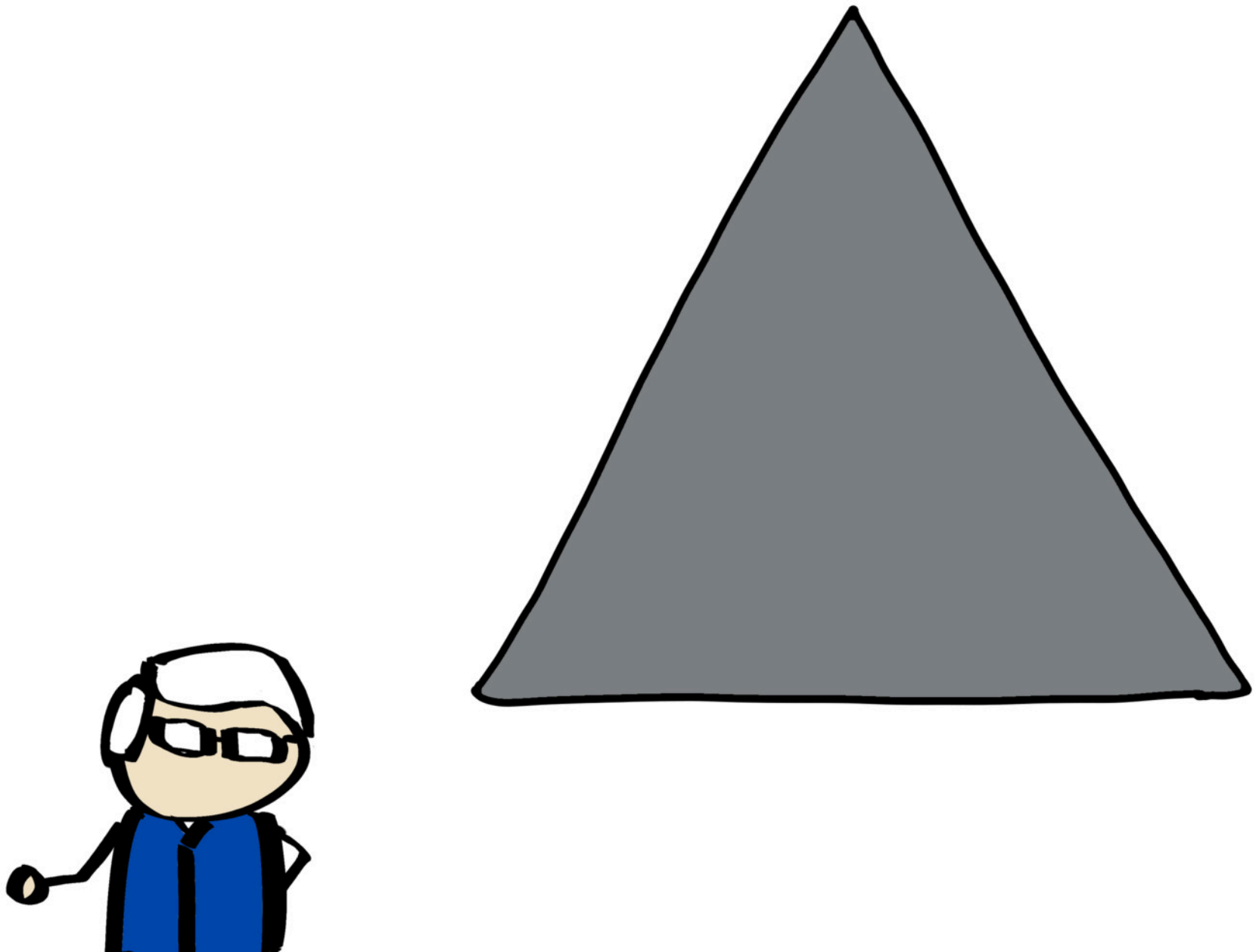
BELIEF & EVIDENCE IN EMPIRICAL SOFTWARE ENGINEERING

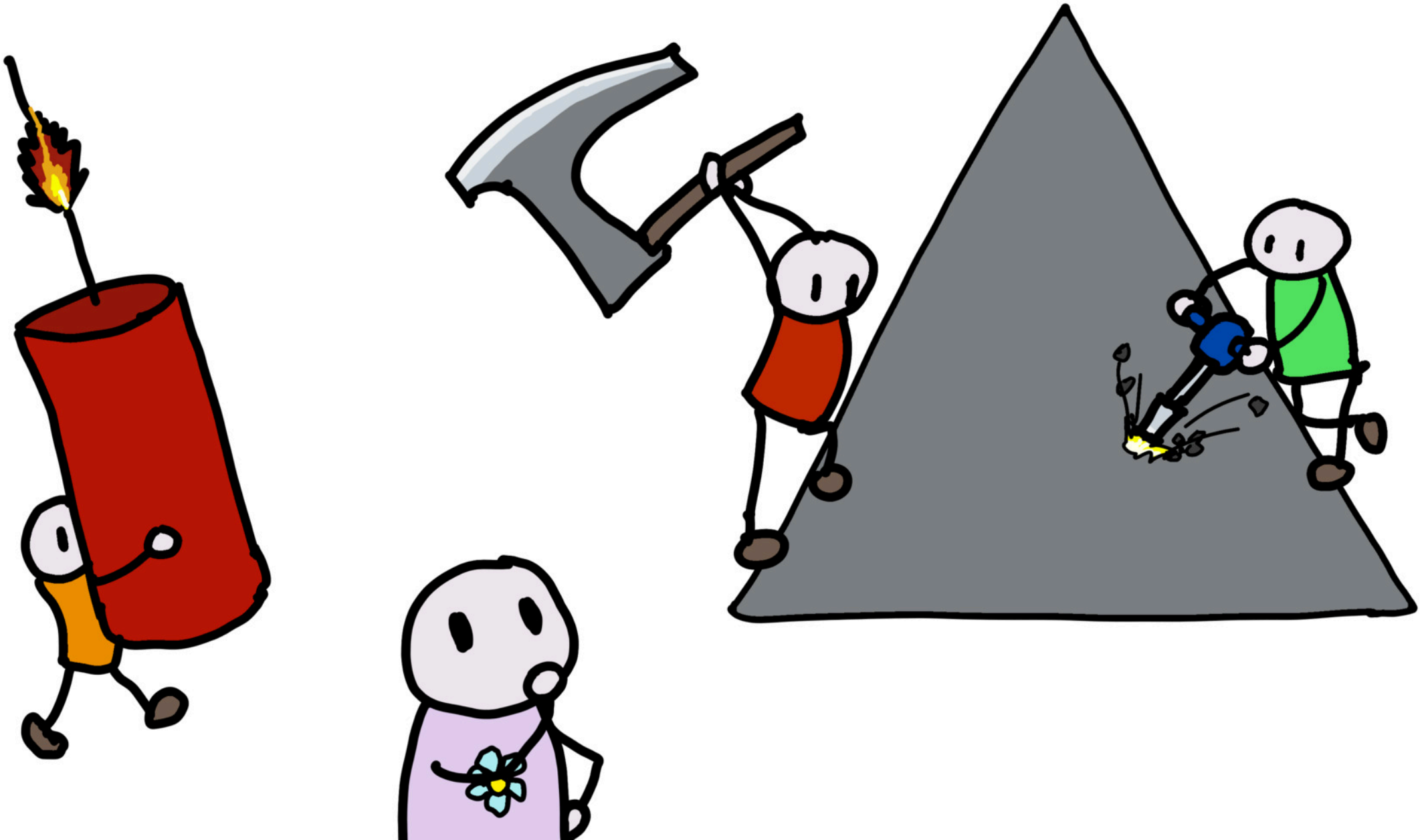
Devanbu, Premkumar and Zimmermann, Thomas and Bird, Christian

2016

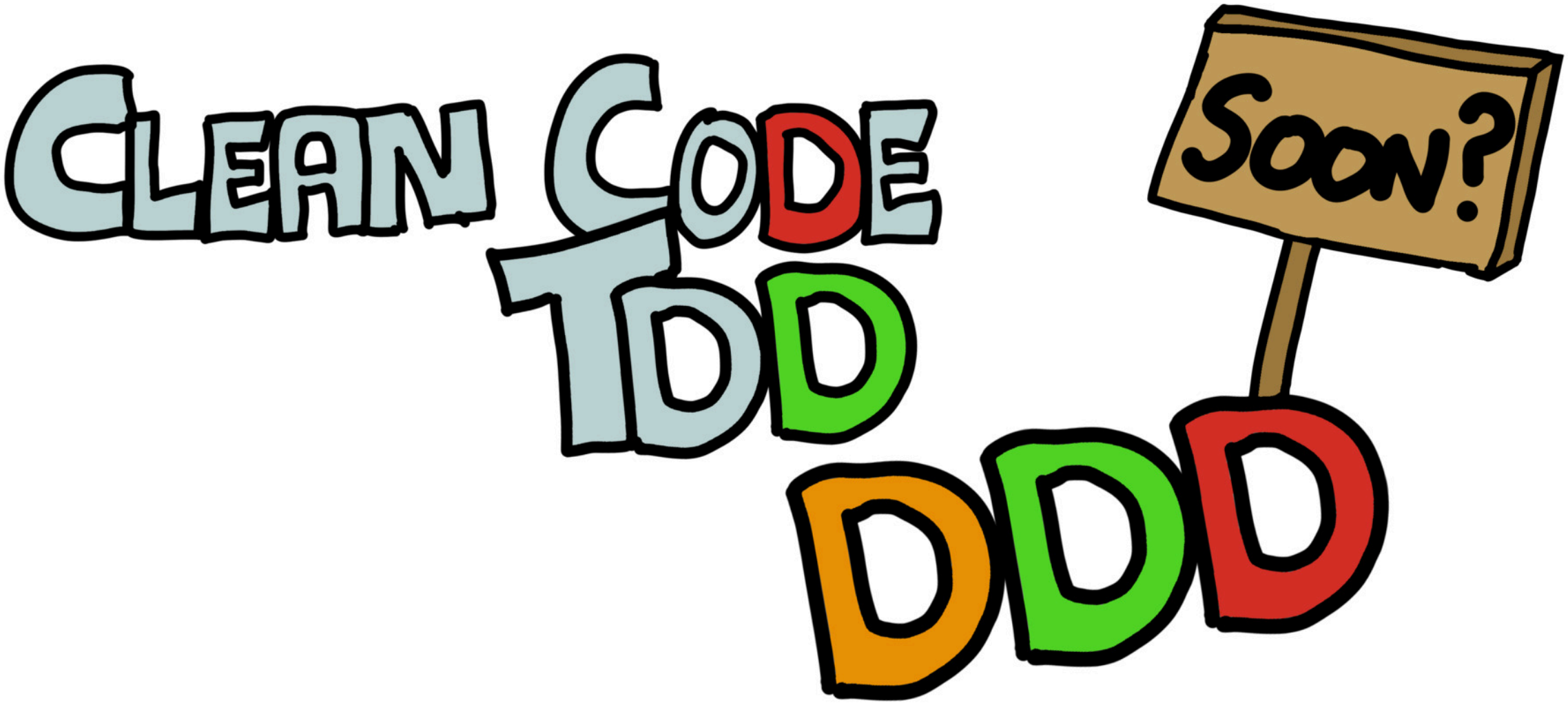

Importance Ranking
5
4
3
2
1
0
Personal Experience
Peer Opinion
Mentor/ Manager
Trade Journal
Research Paper
Other
Source of Opinion








CONTEXT


CLEAN CODE
TDD
DDD
SOON?

MERCI

MERCI

MERCI

AGICAP

BORINGDEV.EU

@VICTORLAMBRET