

Unit tests a philosophy and a help face to its own software

Feedback on 13 years of personal
practice

LAPP / CNRS / ANNECY – 19/01/2024

Sébastien Valat

Tests unitaires une philosophie et une aide face à son logiciel

Retour sur 13 ans de pratique
personnelle en HPC

LAPP / CNRS / ANNECY – 19/01/2024

Sébastien Valat

Plan



1. **Why I started**

2. **A little bit of philosophy & motivation**



3. **Thinking about testing methods**

4. **My own experience, feelings**

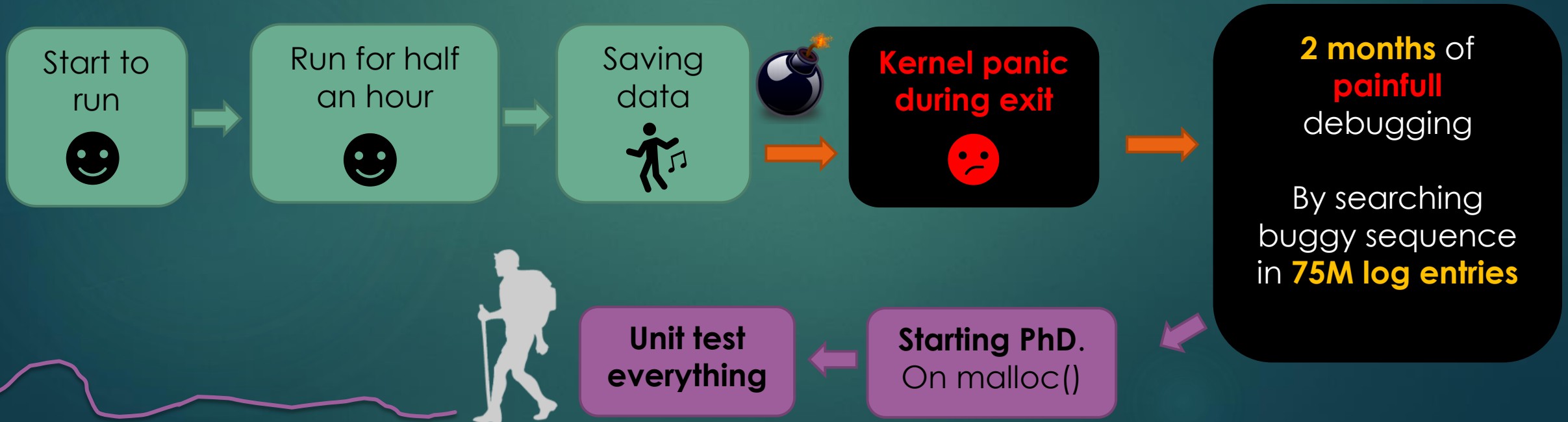


5. **Timings**

Why I started

Once upon a time...

- ▶ Master theses (2009) => **Linux kernel module**
- ▶ **5 months** : module is **working** well on **full KDE session** !
- ▶ Lets try on a real simulation (**1,5 millions C++ lines app & 16 threads**)



My source of thinking

- ▶ **Mostly my own** (home / PhD / post-docs / engineering) **work**
 - ▶ I hardly unit test since **13 years**
 - ▶ 4 years of **scrum** dev in team
- ▶ Sample
 - ▶ **17** projects
 - ▶ **190129** code lines
 - ▶ **C++ / C / rust / python / NodeJS / Java / GO**
 - ▶ From **3700** lines to **33173** lines
 - ▶ Code coverage starting from **43%** to **93%**
- ▶ **Some projects without unit tests !**
 - ▶ **150 000 lines project & 50 devs**

A little bit of philosophy & motivation

How much mistakes costs later .. ?

- ▶ **Manhattan** project, 1945, Hanford
- ▶ There was a **nuclear reactor**
- ▶ For **plutonium** production
- ▶ **Takes** water in
- ▶ **Cooled** the reactor
- ▶and **dump** the water **out**...



Then there was wastes to handle...

- ▶ **Easy** and **quick** and **cheap** solution
- ▶ Make a **hole**,
- ▶ **Dump** everything in
- ▶ **Cover** with sand.

- ▶ **Costs** estimation.... ~12 mens,
- ▶ An excavator
- ▶ A truck

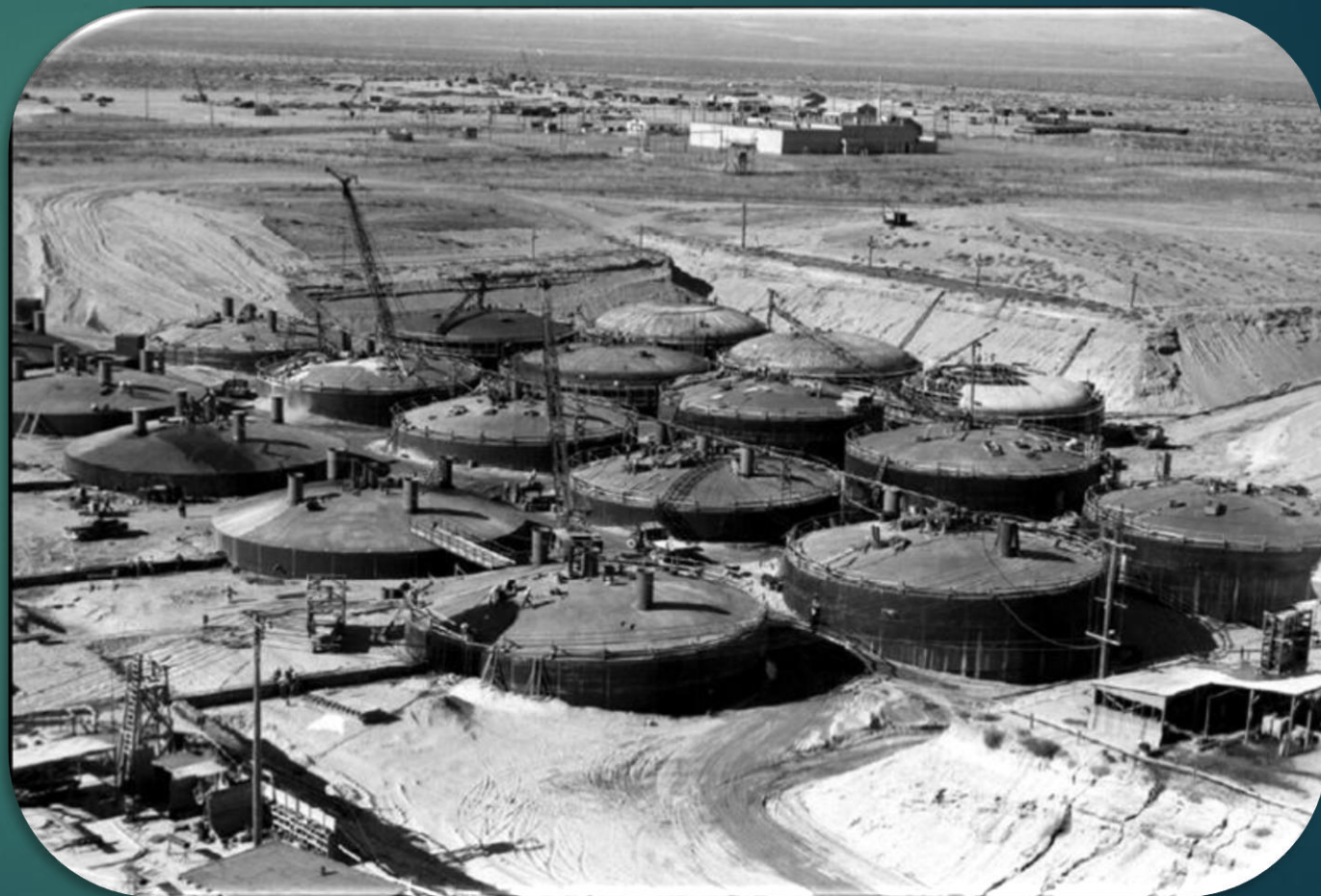


Then there was wastes to handle...

- ▶ For **liquids / muds**....
- ▶ Solution was to build 177 **tanks**
- ▶ **Store** 710,000 m³

- ▶ In the **desert**,
- ▶ Dump wastes in
- ▶ And **cover with sand**....

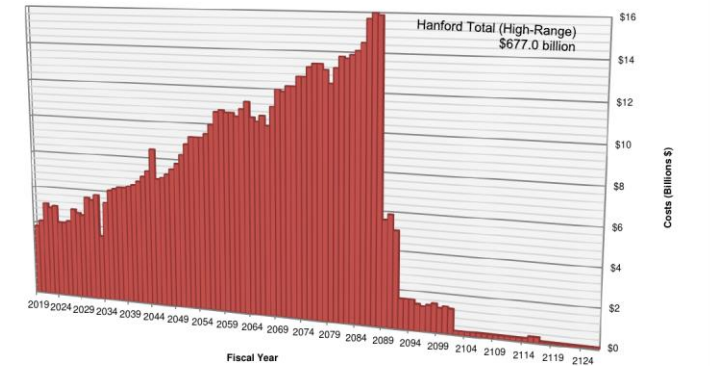
- ▶ Now, **55 years** later....
- ▶ They now (2010) **start to leak**...



<https://tlarremore.wordpress.com/2016/02/28/uncontrolled-spread-of-contamination-nuclear-waste-material-hanford-nuclear-reservation-usa/>

Today: that's **technical debt**

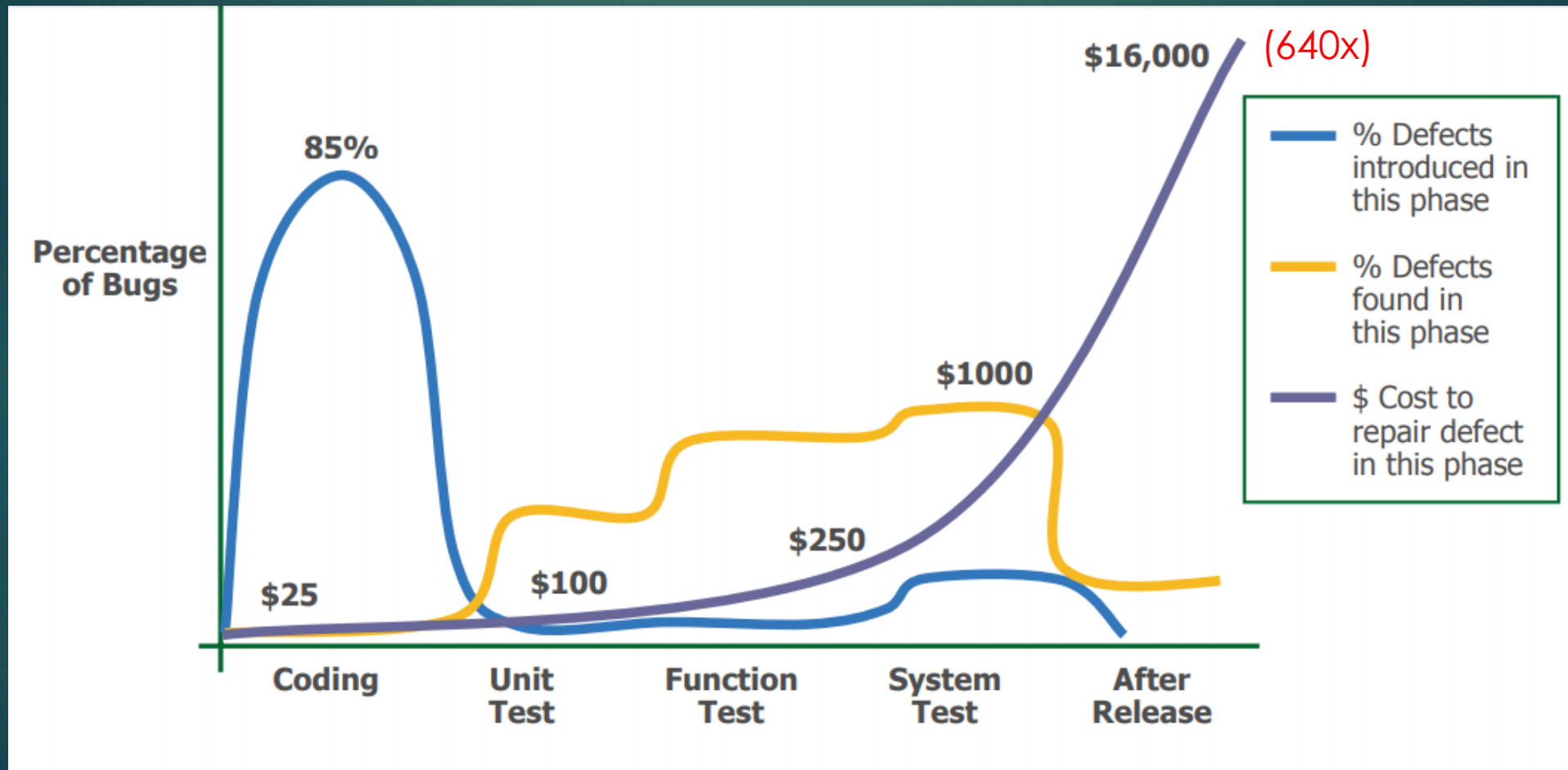
Cleanup until **2090**
And estimated
~300-600 billions \$.



See Appendix D for risk methodology and results.
Figure ES-2. Hanford Site Remaining Estimated Cleanup Costs (High-Range) by Fiscal Year (includes both RL and ORP).

Came back to software....

Capers Jones, 1996



Source: Applied Software Measurement, Capers Jones, 1996

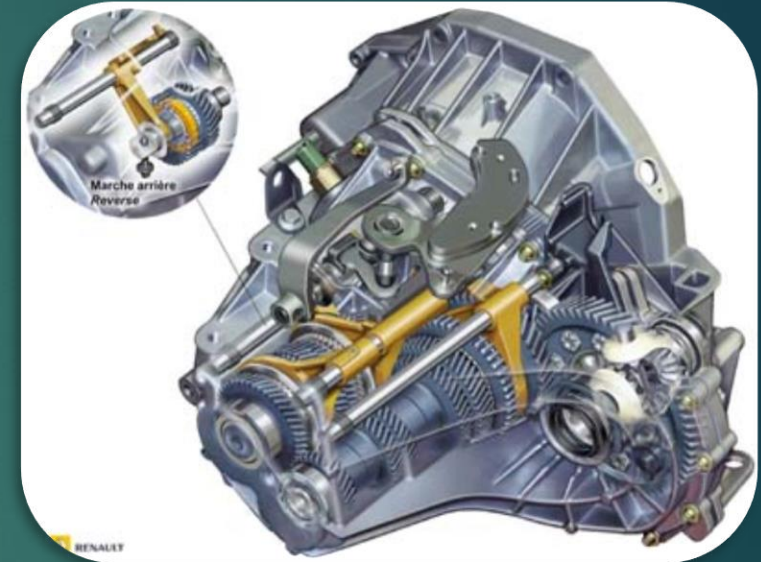
Thinking about testing

Lets think you are a car engineer

14



https://www.motonews.pl/renault/zdjecie-ren-ren_clio_2005_012.html



http://www.auto-innovations.com/site/images8b/Renault_scenic_TL4.jpg

- ▶ You work for Renault (we are French... :D)
- ▶ You want to **build a car**
- ▶ You work on the **gear box**

You make no test...

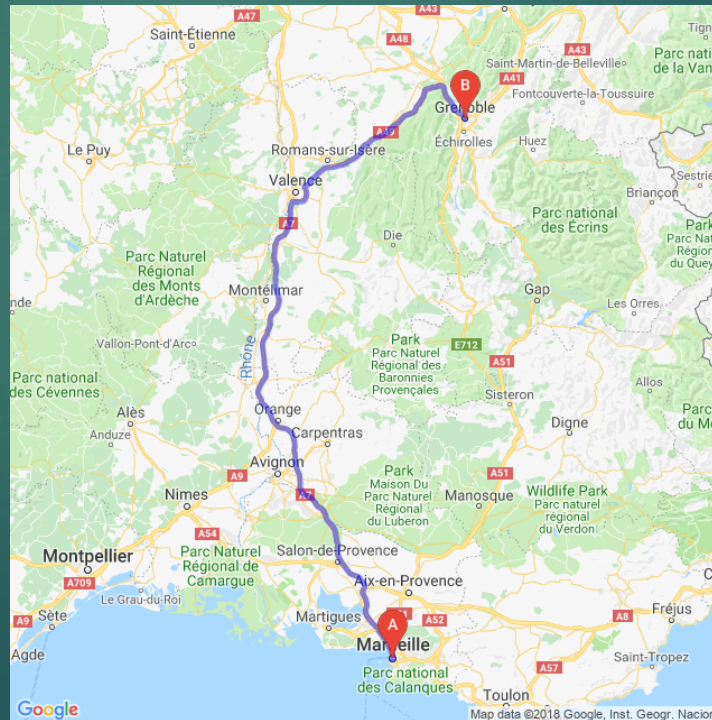
15

- ▶ **Sell** the car **directly** to **customer** and **see**
- ▶ **Would you by ?**



Method 1 : manual test

- ▶ Way to test a **new gear** we added
- ▶ Make a Grenoble – Marseille



Méthode 2 : manual testing

17

- ▶ A bit better : **in controlled environment**
- ▶ **Test circuit**
- ▶ Get a precise **list of tests to perform**
- ▶ We need to define
“a test plan”



Method 3 : automated integration tests

- ▶ We **build** a **prototype** and we run the **tests**
- ▶ **Each time** you change a gear in the gear box ?



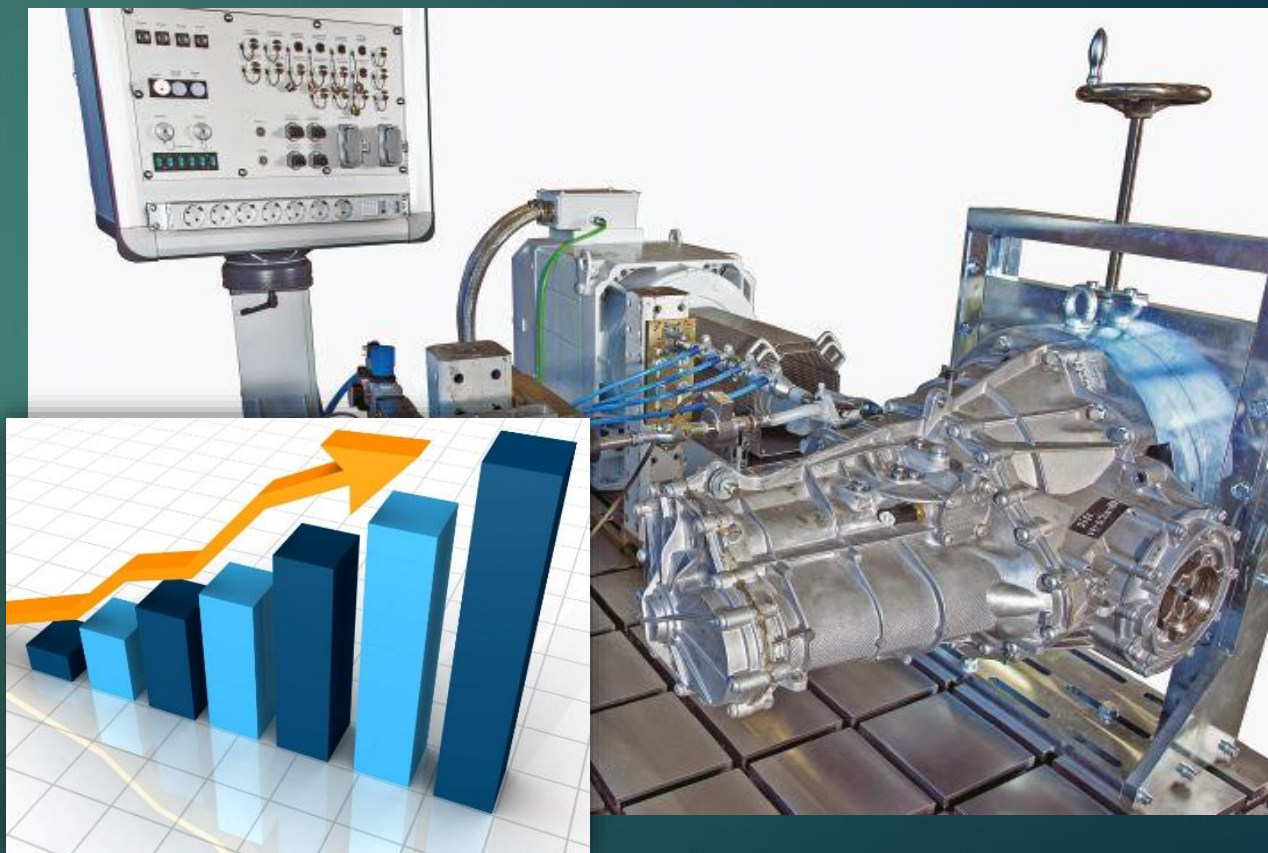
<http://www.thedetroitbureau.com/wp-content/uploads/2016/05/IIHS-Camaro-Crash-Test.jpg>



<https://www.automobile-propre.com/crash-test-renault-zoe-securite/>
<http://maguy69.m.a.pic.centerblog.net/o/969011b4.jpg>

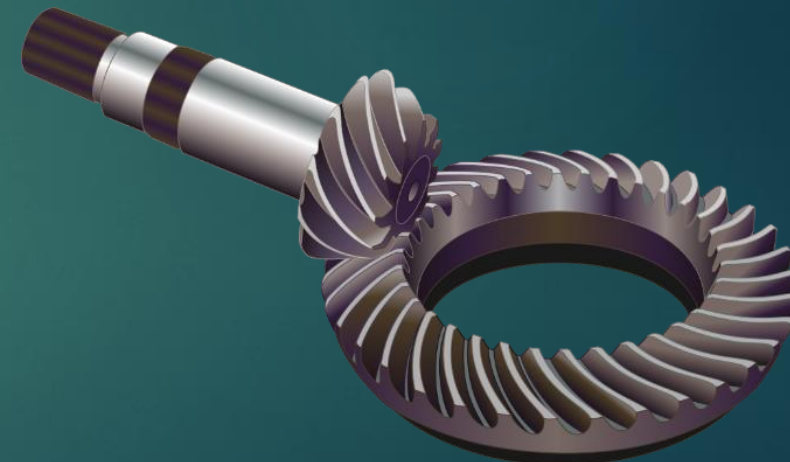
Method 4 : **unit** test

- ▶ You use **a test bench**
- ▶ Test **only** the **gear box**
- ▶ In **controlled situation**
- ▶ Can:
 - ▶ put **infrared camera**
 - ▶ **Probes** to see temperature.
 - ▶ **Vibration measurement**



Notice contiguous transition....

- ▶ There is **unit test**
 - ▶ Test **one gear**
- ▶ A little bit more, still unit test
 - ▶ Test **two gears**
- ▶ ...
- ▶ A little bit more, **integration** test
 - ▶ Test the **gear box**
- ▶ **End to end**, now **test in the car**.



<https://www.indiamart.com/proddetail/automotive-spur-gear-19598784273.html>
https://en.wikipedia.org/wiki/Spiral_bevel_gear#/media/File:Gear-kegelzahnrad.svg

Run example - OK

```
sebv@sebv6:~/2022-01-unit-test$ █
```

But how it looks ?

- ▶ The simplest test in python :

```
def test_abs_value(self):  
    assert abs_value(-10) == 10  
    assert abs_value(10) == 10
```

What is a unit test in python ?

```
from unittest import TestCase

class TestParticle(TestCase):
    def test_move(self):
        # build a particle
        particle = Particle(0)

        # test the initial position
        self.assertEqual(particle.get_x(), 0)

        # move
        particle.move(10)

        # test the final position
        self.assertEqual(particle.get_x(), 10)
```

A bit more advanced one

```
from unittest import TestCase

class TestParticle(TestCase):
    def test_collide(self):
        # build two particles
        particle1 = Particle( 0,5, -1.5)
        particle2 = Particle(-0,5, 1.5)

        # collide particles
        dt = 1.0
        collide = Physics.elastic_collide(particle1, particle2, dt)
        self.assertTrue(collide)

        # checks
        self.assertEqual(particle1.get_vx(), 1.5)
        self.assertEqual(particle2.get_vx(), -1.5)
```

Most unit test frameworks
relies on:
assert keywords

Run example - failure

25

```
sebv@sebv6:~/2022-01-unit-test$ █
```

A realistic case

```
sebv@sebv6:~/Projects/iocatcher/build$ █
```

My my own experience, feelings

When trying to push in teams.... [integration]

▶ Integration test

- ▶ Mostly **everybody agree**
- ▶ Not exactly on the way to do it...
- ▶ One dev. already made a **dirty bash script** !
- ▶ Seems easier at first look

▶ Quickly cost a lot

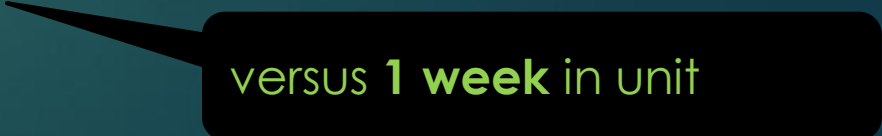
- ▶ Eg. PhD. team project, **10 000** MPI tests, **5000 fails**...
- ▶ **One week** to run everything
- ▶ **Depressing**
- ▶ Harder to debug
- ▶ **Nobody looked** on results except me and another one

When trying to push in teams.... [integration]

▶ Another integration case (costs):

- ▶ Eg. in another team (**scrum**)
- ▶ Only integration test
- ▶ Test suite time : **40 minutes**  versus **9.42 seconds** in unit
- ▶ Complex to maintain : **1.5 dev fully dedicated** to it (over 15)

▶ If your CI env is not stable:

- ▶ **Lots of issues** to maintain the env running
- ▶ Lots of **non code related issues** (**timeout**, ...)
- ▶ Company **migrated the CI env** : **~5 months consumed to migrate**  versus **1 week** in unit

When trying to push in teams....

[unit tests]

▶ Unit tests

- ▶ Required an investment
- ▶ Initial effort
- ▶ We are slower to start
- ▶ Hard to convince devs who never made unit tests
- ▶ **Hard to introduce in pre-existing software**

▶ Common first kill :

- ▶ “This one is **too hard** to test” ❌
- ▶ “This one **call many others**” ❌
- ▶ “I’m sure of this function, it is **so simple**” ❌
- ▶ “Hola, **do not touch this part of the code !**” ❌

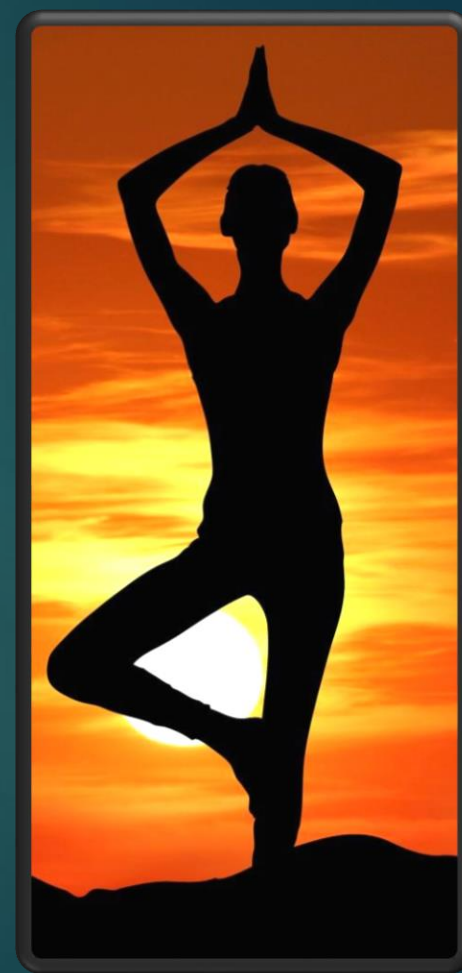
First time I made unit tests

- ▶ I was **not convinced**
 - ▶ But **I tried**
- ▶ Had the impression to **loose my time**
- ▶ It **was hard**
- ▶ I **didn't see the benefits**
- ▶ I **already had most of my codes**
 - ▶ Painfull to unit test for weeks



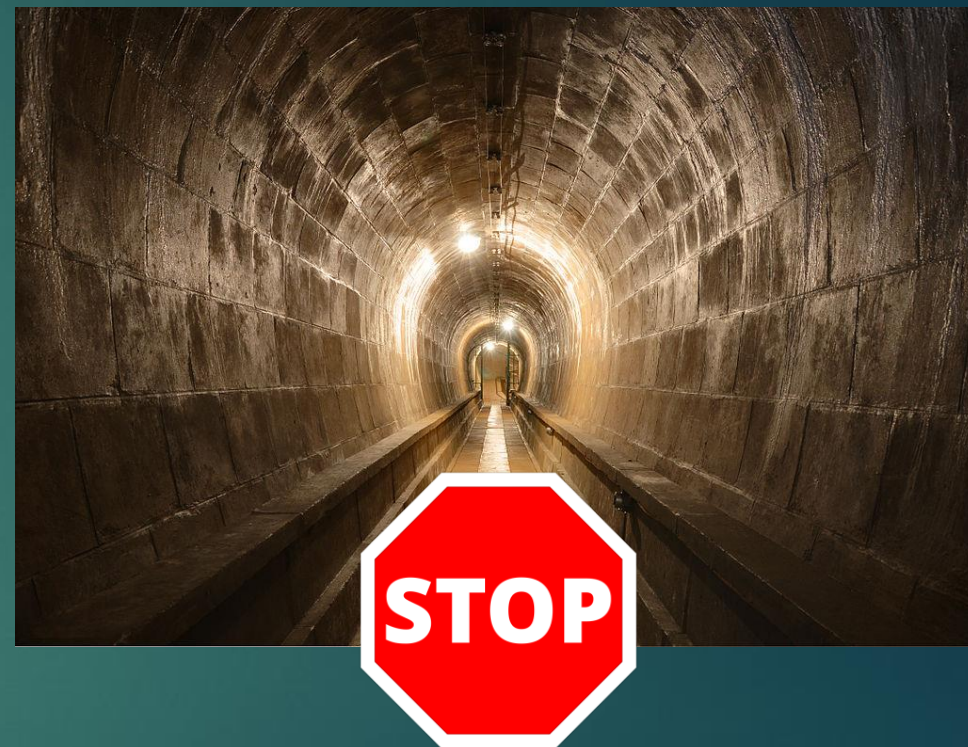
Day to day methodology : **discipline**

- ▶ “This is a POC.... I will make my tests later” ❌
- ▶ You will never do them later
 - ▶ Because your *design* will **not permit**
 - ▶ Because you will **want to move** to **other stuff**
 - ▶ **Nobody** will be **happy** to write unit **tests for ~4 weeks**
 - ▶ **Your boss/commercial manager already sold it to clients....**
- ▶ You already **loosed half the benefits** of unit tests
 - ▶ **Become a more or less useless investment**



Benefits of unit test

- ▶ **That's not only testing (~20%)**
- ▶ It forces you to **think your design**
- ▶ Forbids **global variables**
- ▶ Make **spec**, also for **internal APIs**
- ▶ Open easy door for **refactoring / rewriting**
- ▶ **New developers** are more confident (**you in 6 months...**)



A safety for QA guy

34

Sébastien Volat

- ▶ Quality loss and **rush warnings**.
- ▶ Noticed **via a technical channel** not through **quality exigent guy** !



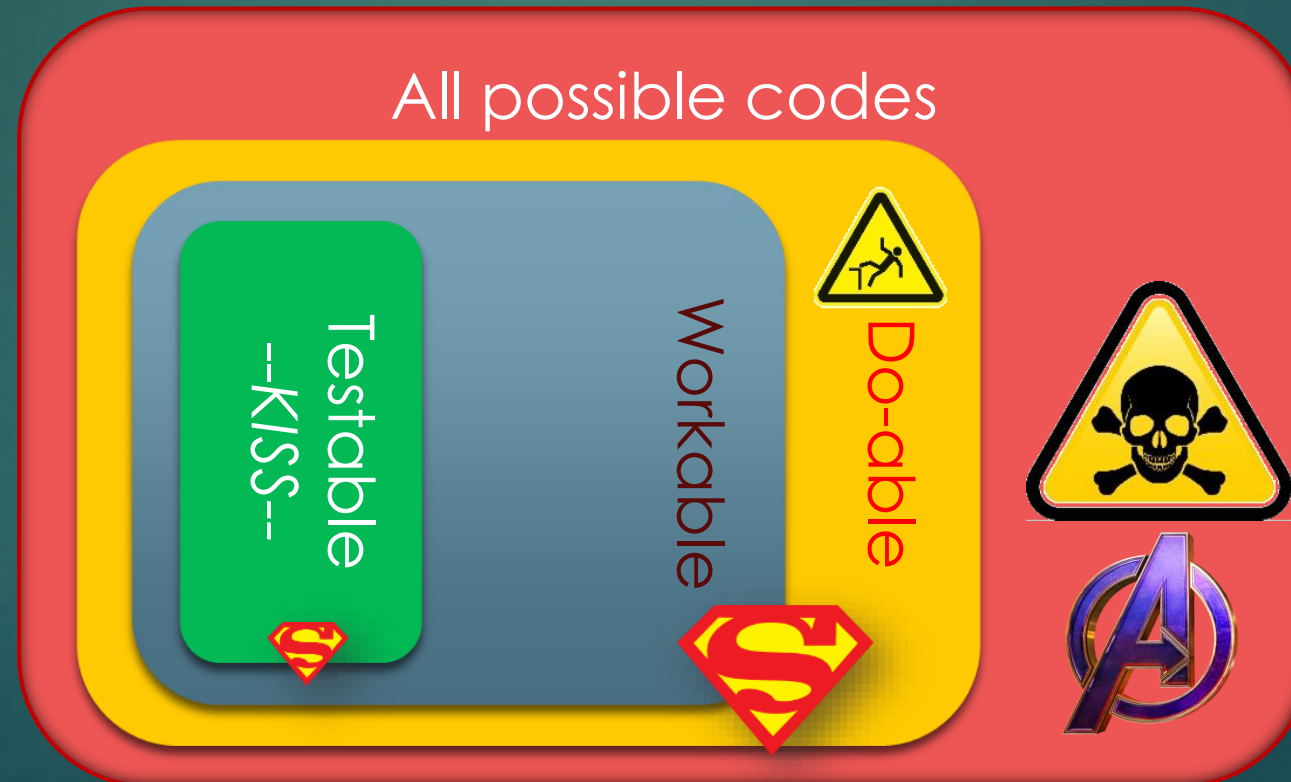
That's a discreet **teacher** !

- ▶ You get **feedback by yourself**
- ▶ **No** need to get **critics from someone else**
- ▶ If you **don't know how to write** your test :
 - ▶ **Your internal API is badly designed !**



That's also constraints

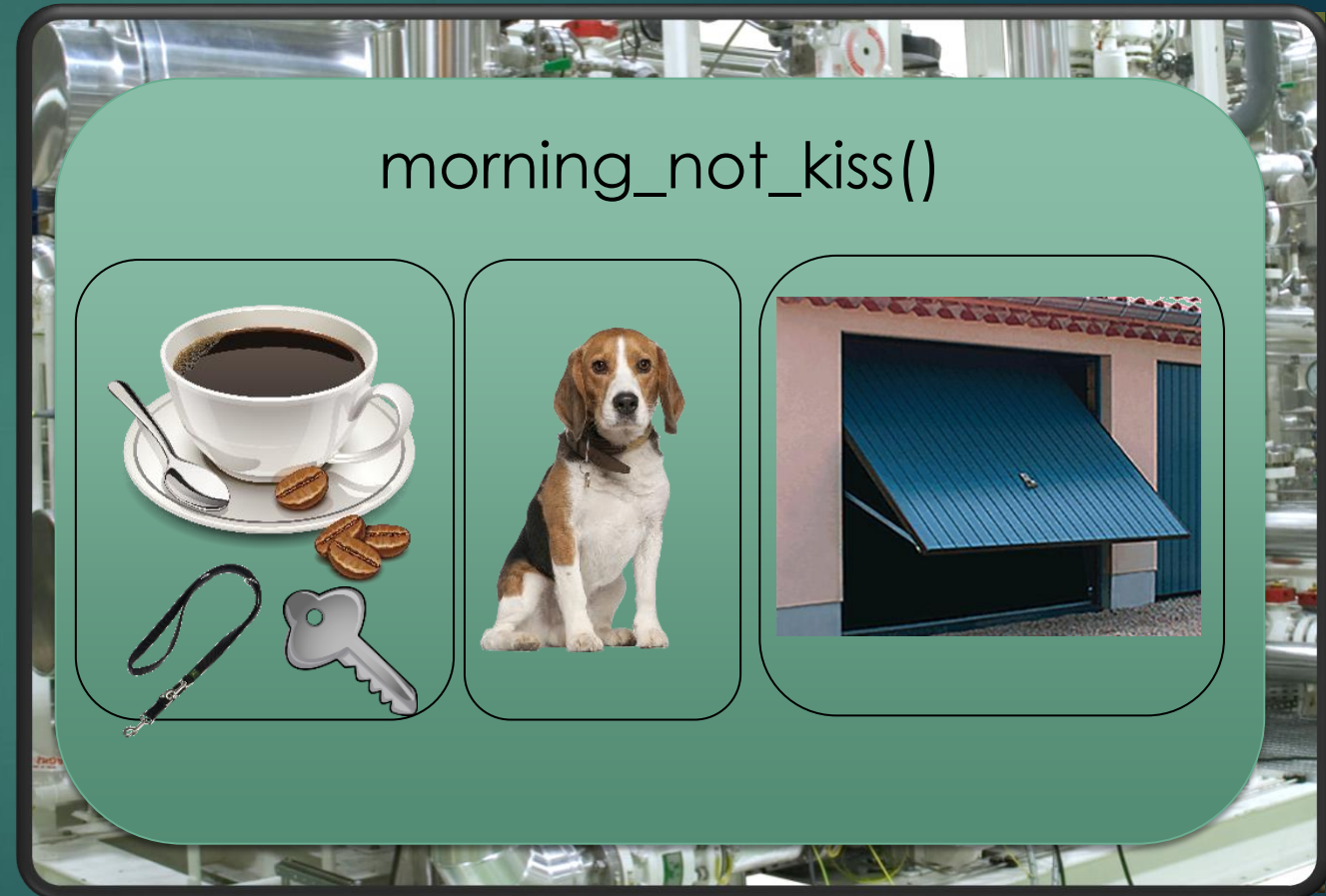
- ▶ **Not all** codes are **unit test-able**



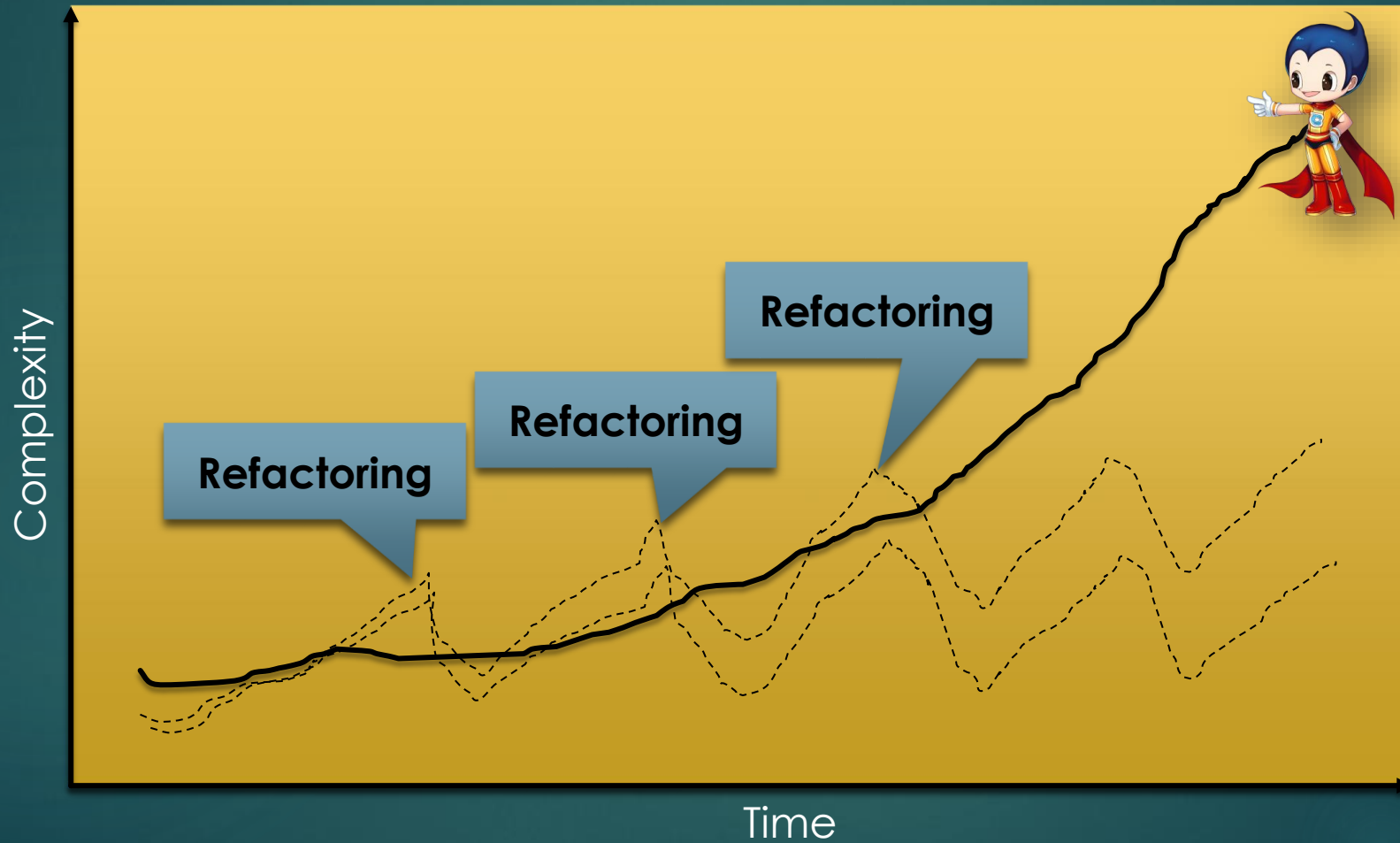
Test a gas machine

37

- ▶ If your **test** become **too complex**
- ▶ You are **certainly** on the **wrong way**
- ▶ **Stop**, **think** and **KISS**



Facing the entropy !



Team dynamic - silos

39

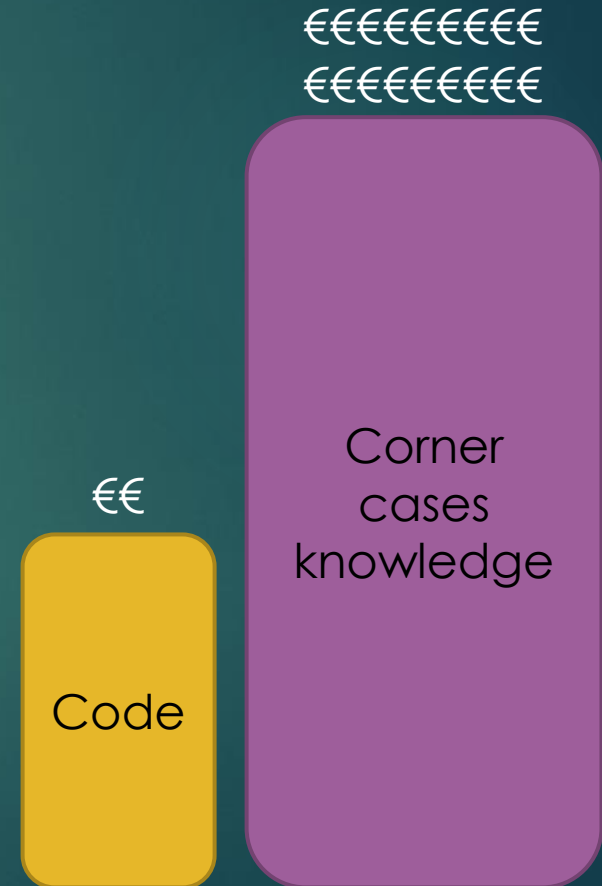
Sébastien Volat

- ▶ **Without test** it is **hard to go** on a **part we do not know** !
- ▶ **Especially in HPC !**
- ▶ So **each dev** will have **his part**
- ▶ It **reduces** the **discussions** in the team
- ▶ **Favor heroes**



Keep the knowledge !

- ▶ The hard **corner cases** are **encoded into the tests**
- ▶ Useful:
 - ▶ On **turnover** or **retirement**
 - ▶ Very **usefull** in case of **rewriting a V2**
 - ▶ To **translate** in another language
- ▶ Eg: **porting my memory allocator**:
 - ▶ **C original** implementation : **~1 year**
 - ▶ **C++ translation** + new algo : **1 month**
 - ▶ **Rust translation** : **2.5 weeks** for the biggest part



A basement of agile methods



Never do **AGILE** or **SCRUM**
without **unit test** !

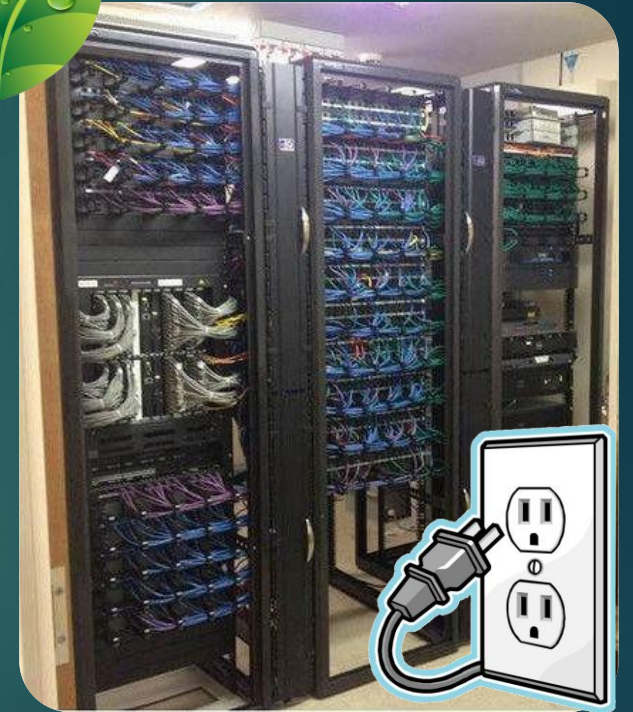
- ▶ That's a REQUIREMENT for the method, **not an option**
 - ▶ For the **technical validity** of the method
 - ▶ For the **dynamic of the team**
- ▶ In agile you **didn't plan**...
 - ▶ If you **cannot refactor** => you are **scrued** or **get very lucky** !

Ecology argument

42

Sébastien Volat

- ▶ You can make the **whole dev locally** on your **laptop**
- ▶ **No need** of **a large dev cluster**
- ▶ **Once done** and validated with unit tests:
 - ▶ Make real **test on cluster**
 - ▶ **Once a week** or two weeks
- ▶ Not anymore per team cluster
- ▶ **Less debugging at scale so less CPU hours !**



Ecology argument - 2

43

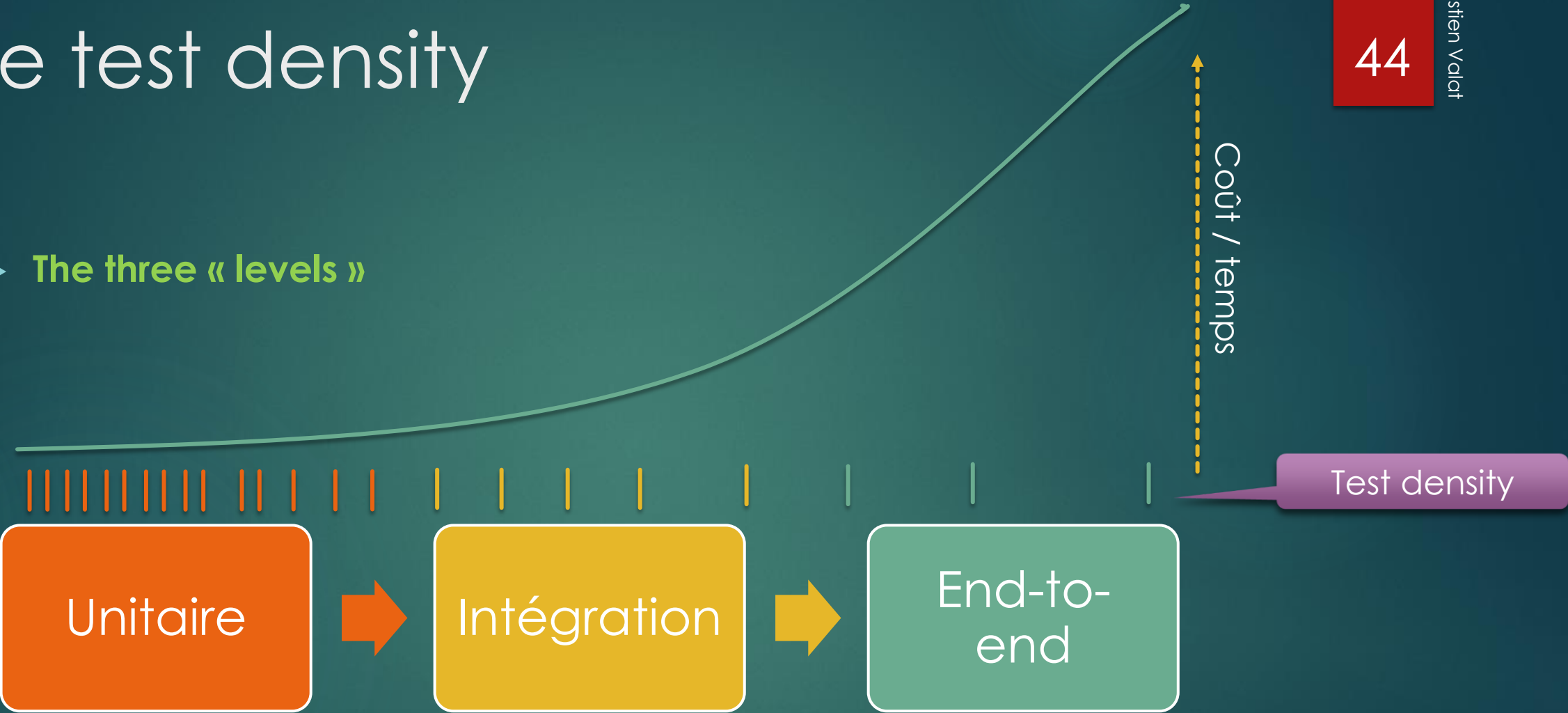
Sébastien Volat

- ▶ CI cycle takes ~ **a minute**
- ▶ Instead of **45 minutes** with only **integration tests**
- ▶ Require **less CI server** resources



The test density

► The three « levels »



► Build (or use) **dedicated tools** for each level

Some framework

Language	Test framework	Mocking
Python	Unittest pytest	unittest.mock
C++	Google test Catch2 Boost test library cppunit ...	Google mock FakeIT
C	Google test Criterion	
Bash	bats	
Rust	[native]	<i>mockall</i>
Go	[native]	<i>gomock</i>

Timings on 1 examples

COSTS AND EXAMPLES

CERN lhcb-daqpipeline

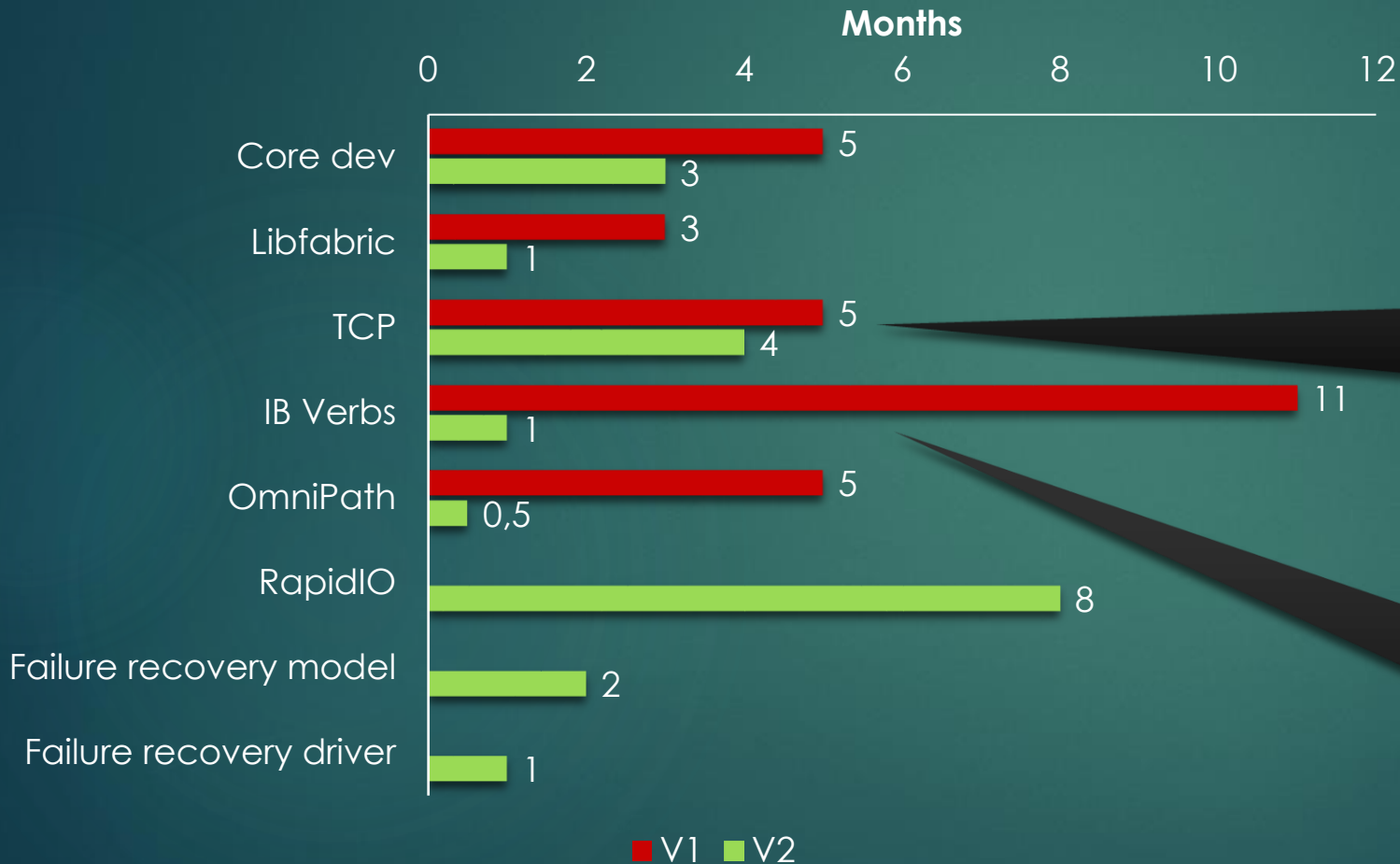
47

- ▶ **LHCB** Acquisition **R&D code** for **scaling studies**
- ▶ Need to handle **40 Tb/s** on
 - ▶ **InfiniBand**
 - ▶ **Omni-Path**
 - ▶ **100G ethernet**
- ▶ Over **500** servers (continuous **80 Gb/s all-to-all**) + send to **~3000**

Compare costs

48

Sébastien Volat



TCP driver :

V1 => network **expert**

V2 => **very basic** C/C++ **knowledge**

IB driver :

V1 => student made an **IB simu.**

V2 => **No** MPI or RDMA **knowledge**

Not gaining only time !

DON'T BE AFRAID BY REAL PROBLEMS

MALT

50

Sébastien Volat

December

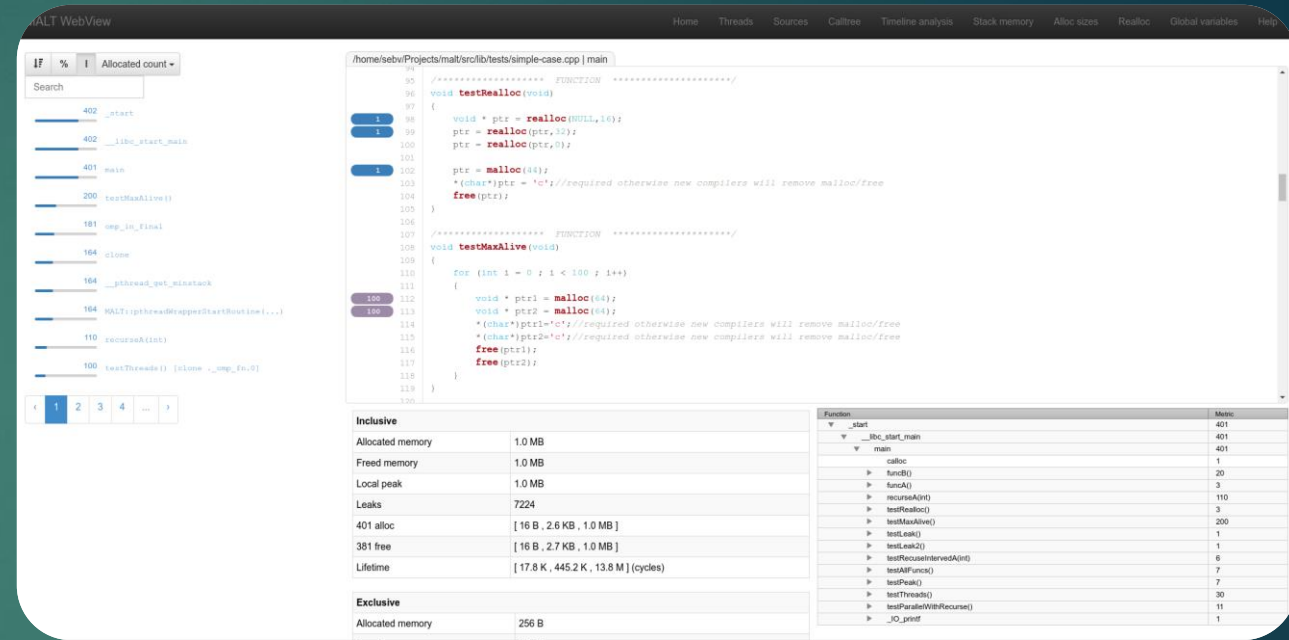
- Working on laptop Core2Duo
- OS: Gentoo / Debian

Mid-Feb

- First run as a POC
- Basic backend + draft GUI

March

- A “real” test at ViHPS
- With a Phd. student stucked with his app

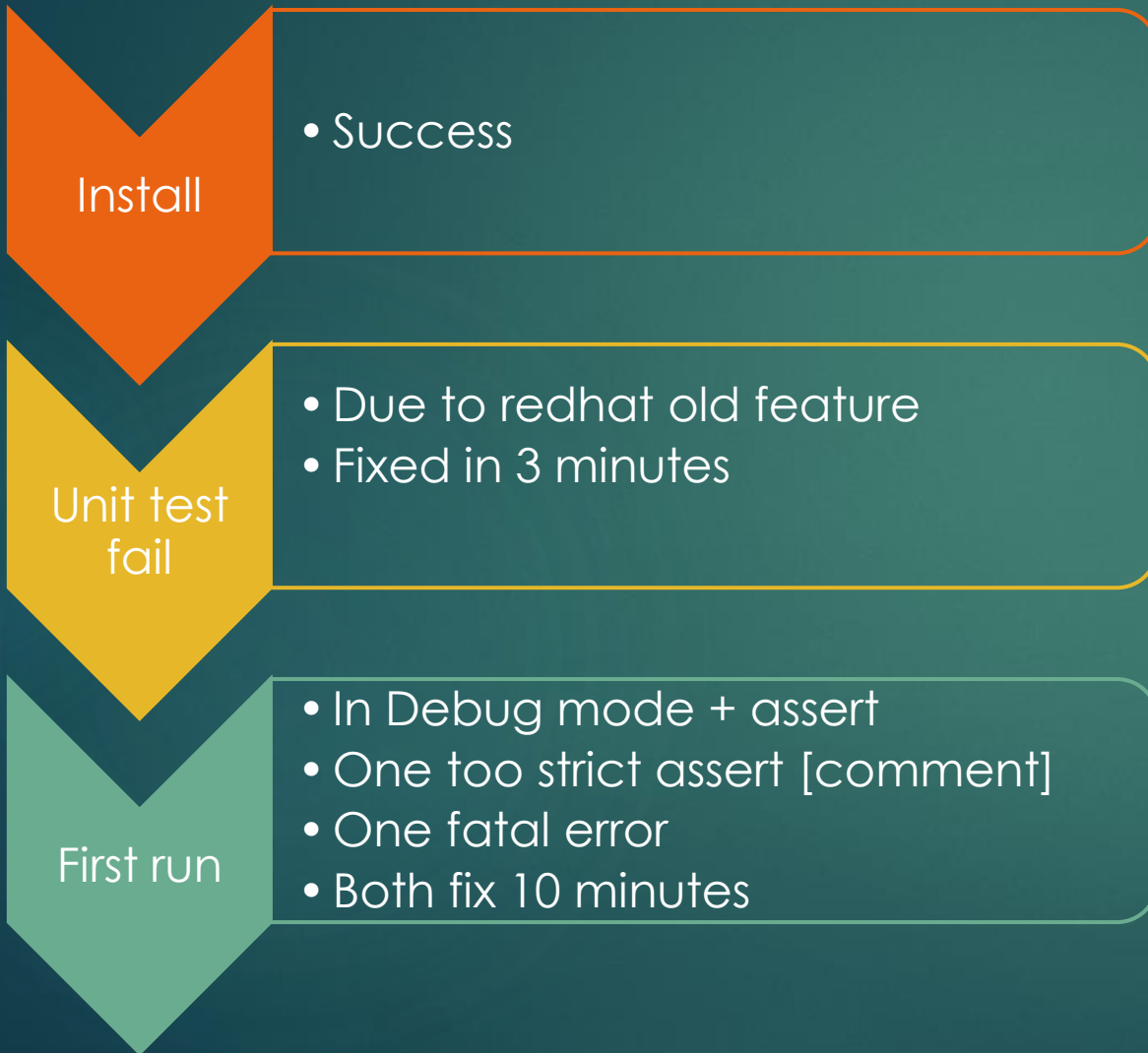


- ▶ The PhD. **student aside me**:
 - ▶ **Stuck** with his code **failing on cluster** due to **out-of-memory**
 - ▶ **“I develop a tool for this, maybe we can test ?”**

- ▶ **“I’m not sure because I started 3 month ago”**
 - ▶ Never tested **MPI**
 - ▶ Biggest (~uniq) code was 1000 lines, C.
 - ▶ His one was **256 tasks, ~30000 Fortran, lfort, Intel MPI**
 - ▶ Cluster OS: **Redhat** (I tested Gentoo)

MALT

52



He forgot global variable

Biggest than what he thought

12-20 GB

MALT

53

Sébastien Valat

- ▶ Total dev : **~8 month at the lab**
- ▶ **1.5 year latter** without touching
 - ▶ Run at CERN on lhcb-daqpipeline (**30000 C++ lines**) => Success
- ▶ Run on **Lhcb framework** (**~2 million lines** + XXXX libraries)
 - ▶ Backend success
 - ▶ NodeJS not loading Json file larger than 600MB => mine 690MB ☹
 - ▶ **~1.5 week data reshaping** and **recursive call** stack **compactation**
 - ▶ File 250MB => display OK

No fear to
quickly expose
to real app !

LHCb HPC Adventure

54

Sébastien Valat

- ▶ Present at CINECA
 - ▶ **Me + my boss** arrived by plane morning
 - ▶ **4 Cineca admins** (~cannot work due to test)
 - ▶ **2 INFN people** implied
- ▶ Cluster status (installation ongoing)
 - ▶ Still **instabilities on OPA** network
 - ▶ **Luster not available**
 - ▶ **PBS not working** since 2 days
- ▶ **Plug my laptop on projector**, then I work !
 - ▶ We didn't pay but I would like to know how much it costs !!!!!!!
 - ▶ **Demo effect ?**



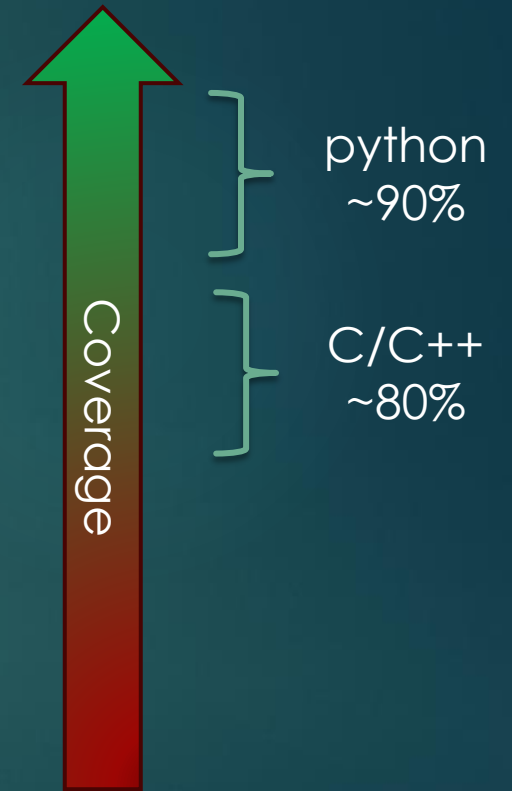
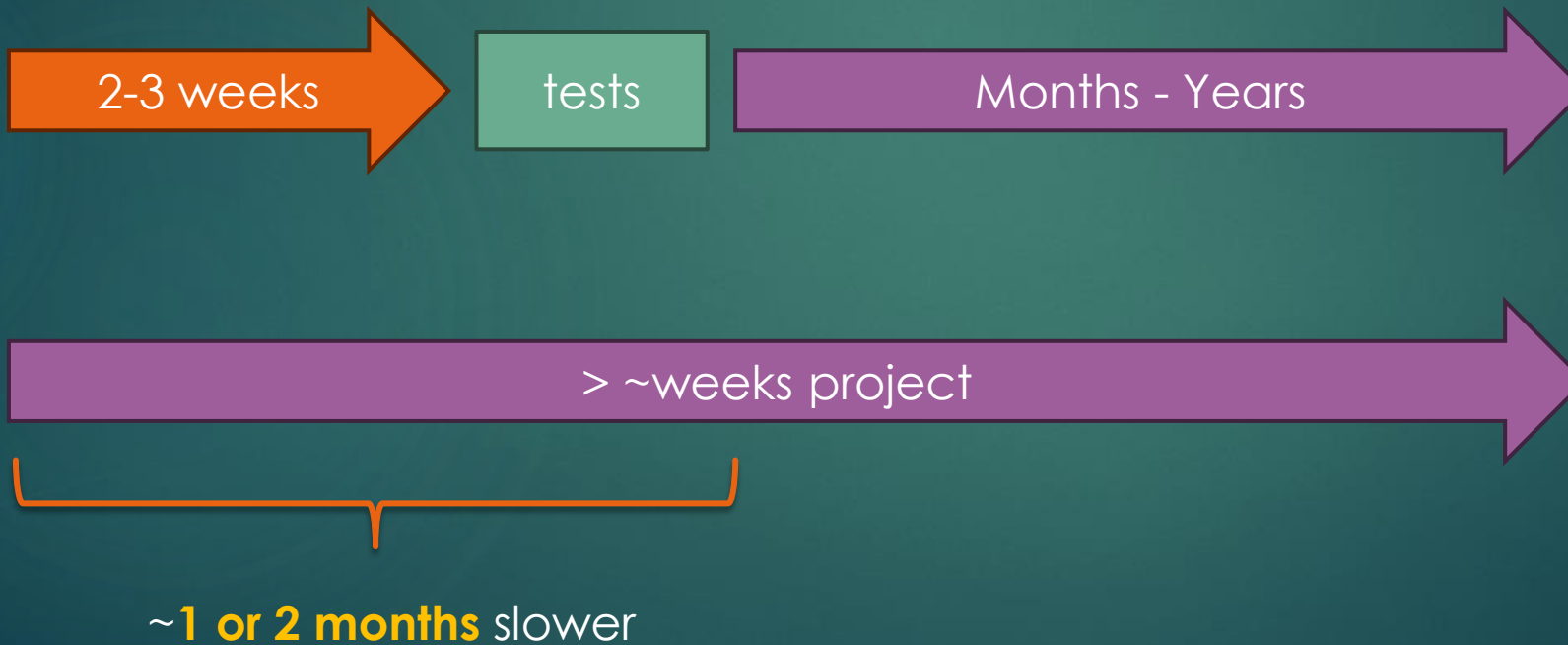
Conclusion

My time rules

56

Sébastien Volat

- ▶ Of course, **depend on** language / objectives / complexity



A least 1 test per function / class

Conclusion

- ▶ Always compare with **real world engineering**
- ▶ We **tend** to **think** because it is **virtual** it **cost nothing**
 - ▶ That's **absolutely wrong** on **long term**
- ▶ In research we **want to explore** algos
 - ▶ We need to **change the code many times**
 - ▶ Hard if we **lose months on debugging**
- ▶ There is a **human aspect**
 - ▶ the more interesting part for me
 - ▶ **In research** => **we let code** to the **next guy** (due to short contracts) !

Be patient, look the dragon in the eyes

58

Sébastien Volat



Thanks