A word about unit test philosophy

SÉBASTIEN VALAT – ATOS/BULL ECHIROLLES – 12/2019

Disclamer

- I see unit testing as a **(philosophical) path**
- We **cannot apply** all **in on week**…
- We, **as a group**, might also **not agree** with **everything**
- **Please be critic**

Be patient,
look the dragon in the eyes

My source of thinking

- One book on **TDD, conference** video, **research papers**
- But **mostly my own** (home/PhD/post-doc) **work**
  - I hardly unit test since **9 years**
- Sample
  - **15** projects
  - **190129** code lines
  - **C++** / **C** / rust / python / NodeJS / Java / GO
  - From **3700** lines to **33173** lines
- Coverage starting from **43%** to **93%**

Plan

1. A little bit of philosophy & motivation
2. Thinking about testing methods
3. A word on my own experience, feelings

4. Quick look on frameworks / tools
5. Infrastructure tricks

6. Mutable testing

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**…

https://commons.wikimedia.org/wiki/File:Hanford_N_Reactor_adjusted.jpg

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

http://www.planetexperts.com/hard-lessons-in-the-us-environmental-protection-agency/

Then there was wastes to handle…

- For **liquids / muds**….
- Solution was to build 177 **tanks**
- **Store** 710,000 m^3

- 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/

What's inside now

- **No inventory** of the **mixture**
  - Acid
  - Little bit of Pu
  - Little bit of Ur
  - Little bit of Actinides
  - Little bit of Sr
- **Hard to pump** and handle
  - Very **corrosive**
  - **Radioactive**

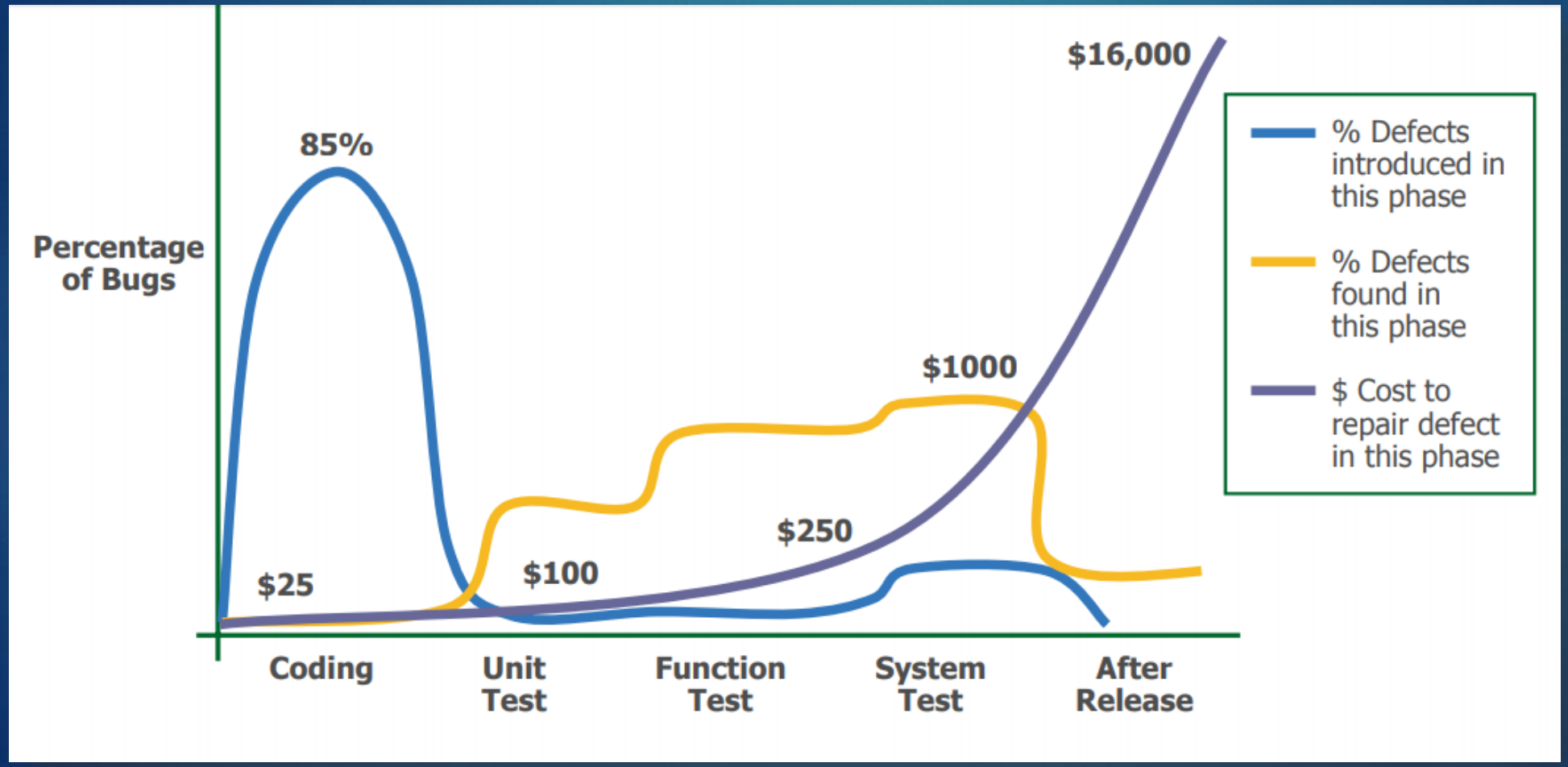
http://large.stanford.edu/courses/2011/ph241/eason2/

Today

- Hanford site now costs **2-3 billion $** every **year** to US. DOE
- Now currently **11000 people** working on site to **cleanup**
- **Since** roughly **35 years** and for **up to (at least) 2046**
- Total : **~120 billion $**
- Everybody **would like to hide** this big mistake

Came back to software….

Capers Jones, 1996



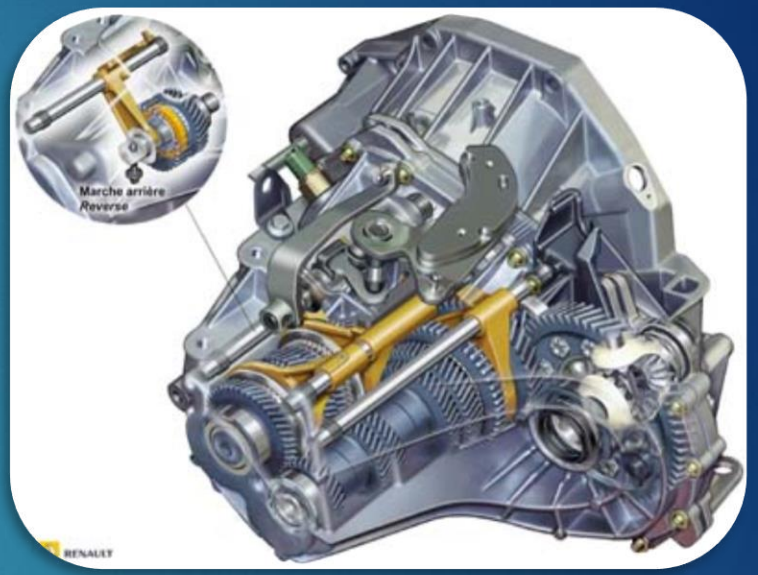
Source: Applied Software Measurement, Capers Jones, 1996

2011 - ref

675 compa.
35 gov/mili.
13500 proj.
24 countr.

Thinking about testing

Lets think you are a car engineer

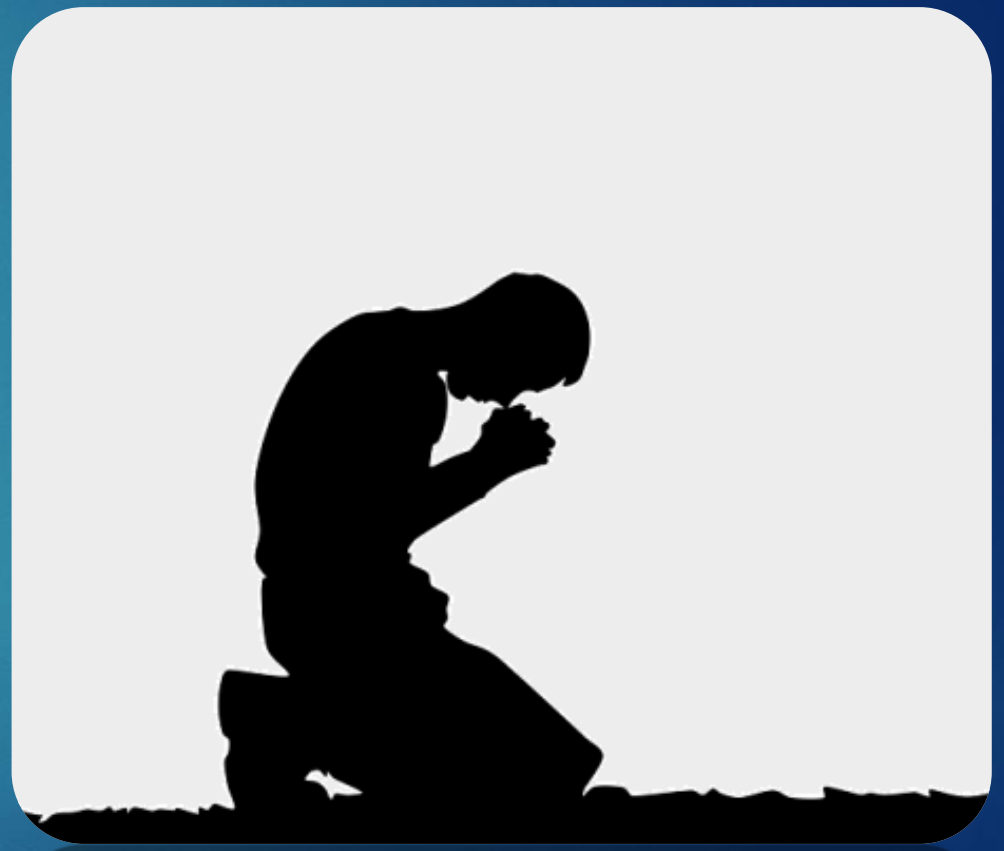


http://www.auto-innovations.com/site/images8b/Renault_scenic_TL4.jpg

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

You make no test…

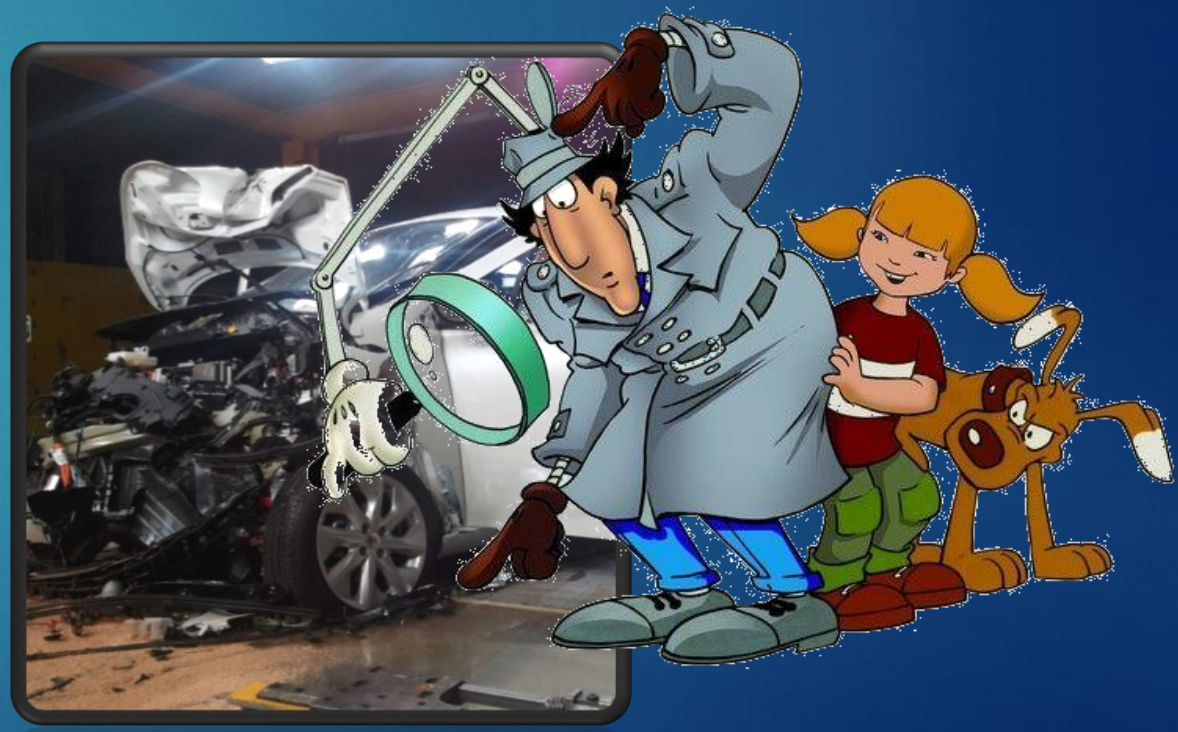
- **Sell** the car **directly to customer** and **see**
- **Would you by ?**



Method 1 : **integration** tests

- You **build** a **prototype car** and make **a crash tests**
- **Every time** you **change a gear shape** 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 2 : **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**

https://www.techbriefs.com/component/content/article/tb/features/application-briefs/13978

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

Some word on my own experience, feelings

When trying to push in teams…. [integration]

- **Integration test**
  - Mostly **everybody agree**
  - Not exactly on the way to do it….
  - Seems easier at first look

- **Quickly cost a lot**
  - Eg. CEA MPC 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

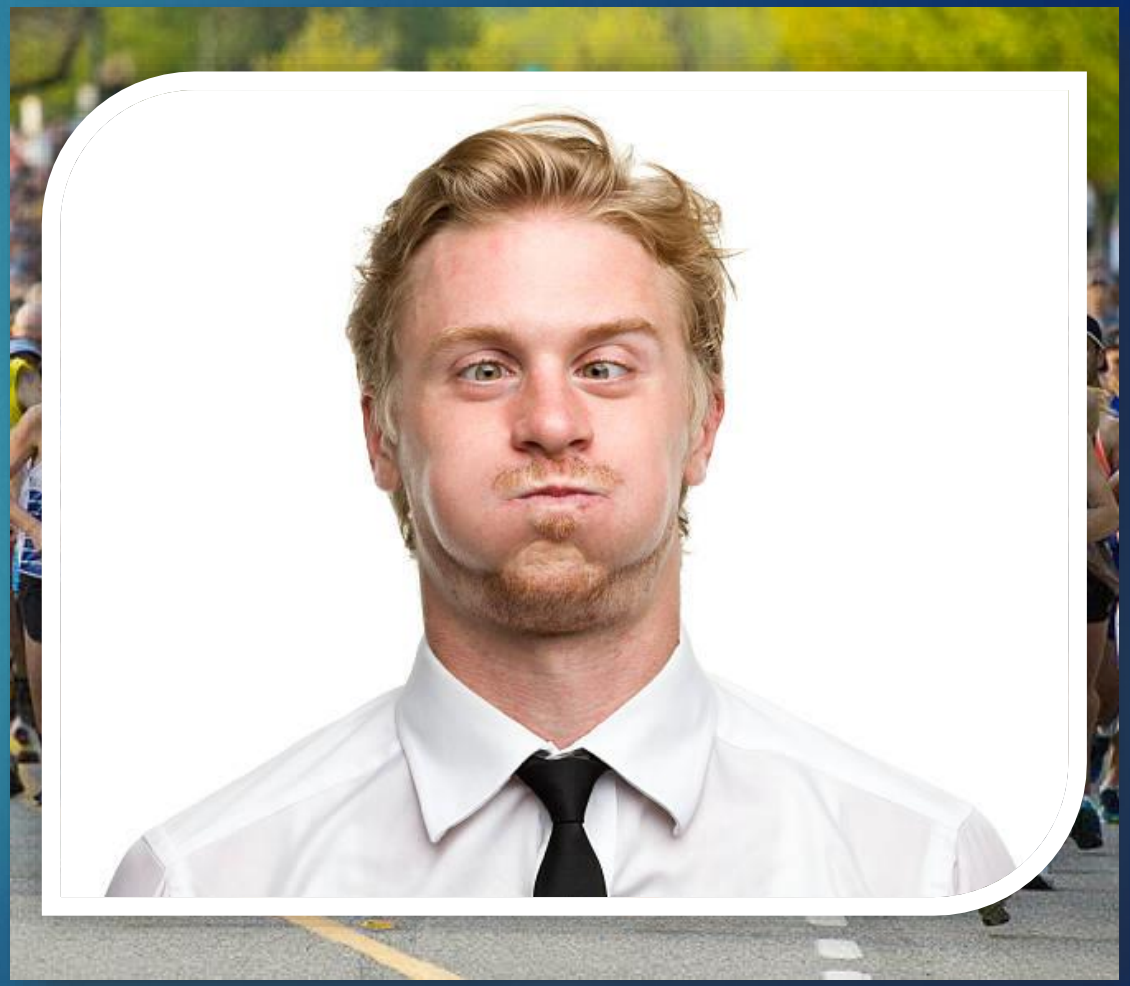
When trying to push in teams…. [unit tests]

- **Unit tests**
  - Required an investment
  - Initial effort
  - We are slower to start
  - **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**" ✖

First time I made unit tests

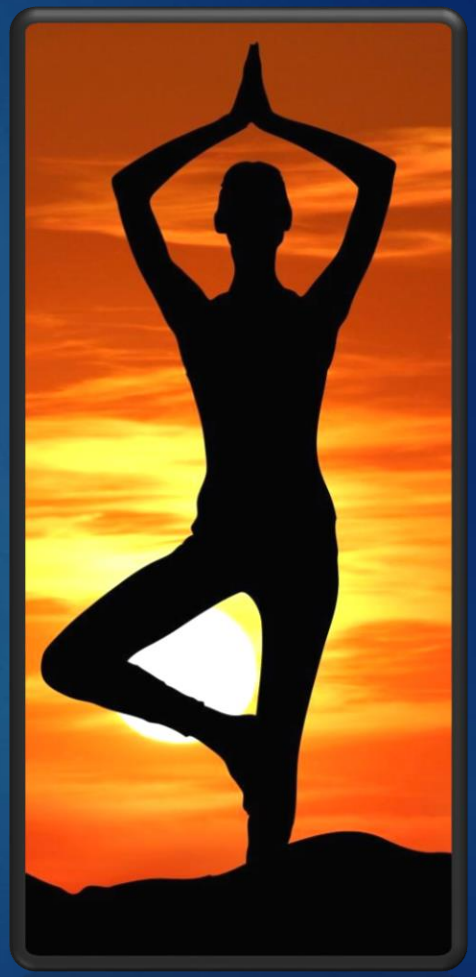
- I was **not convinced**
  - But **I tried**
- Had the impression to **loose my time**
- It **was hard**
- I **didn't see the benefits**
- **I already had most of my codes**
  - Painfull to unit test for weeks



That's also adequate tools and ways to work

Day to day methodology : **discipline**

- **"This is a POC…. I will make my tests later"** ✖
- **<u>You will never do them later</u>**
  - 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 cost**

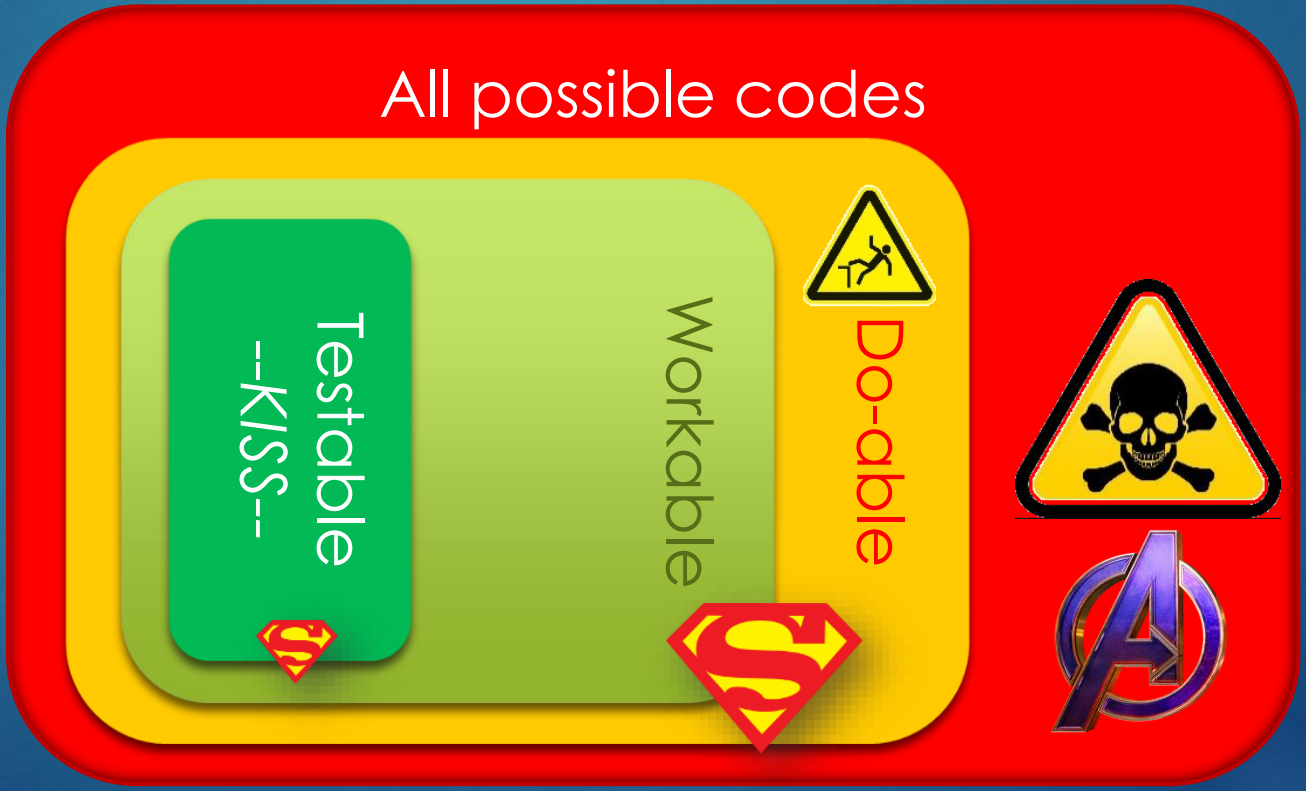


Benefits of unit test

- **That's not only testing**
- 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**…)
- Quality loss and **rush warnings**. Not via a human channel through quality exigent guy !

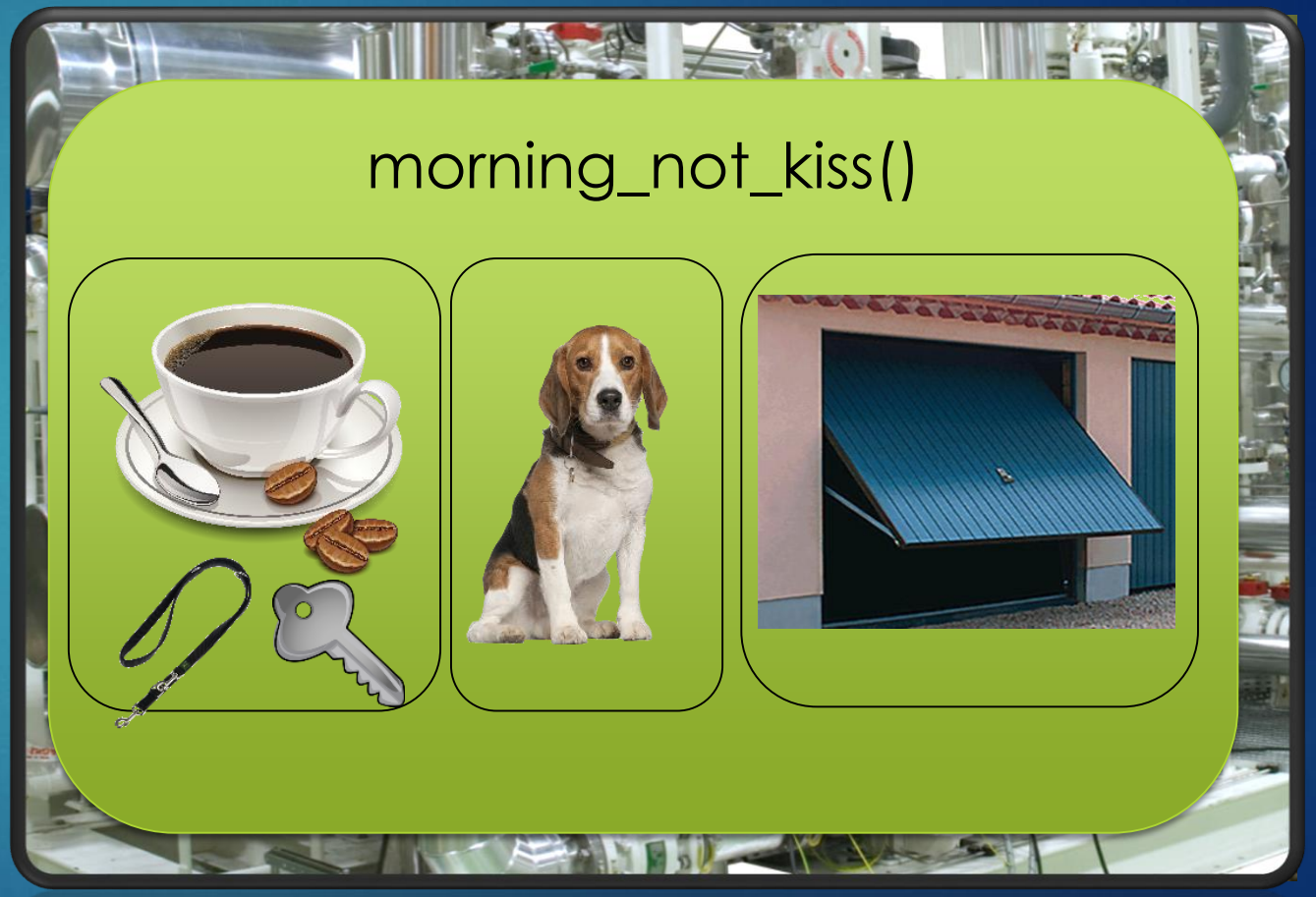
That's also constraints

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



Test a gas machine

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

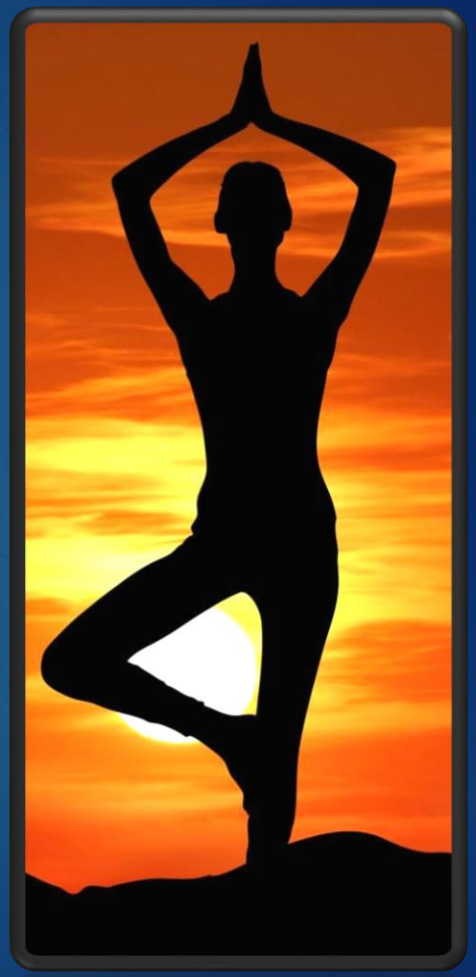


Unit tests should stay simple


```
TEST(TestProject, loadContent_fail_minimum_required)
{
    FileLines content;
    content.push_back("[cdeps_minimum_required 2000.4.3]");

    SpecFile file;
    file.loadContent(content,"none.none");

    Options options;
    Project project(&options);
    EXPECT_EXIT(project.loadSpec(file),
       ::testing::ExitedWithCode(1, "version is too old");
}
```

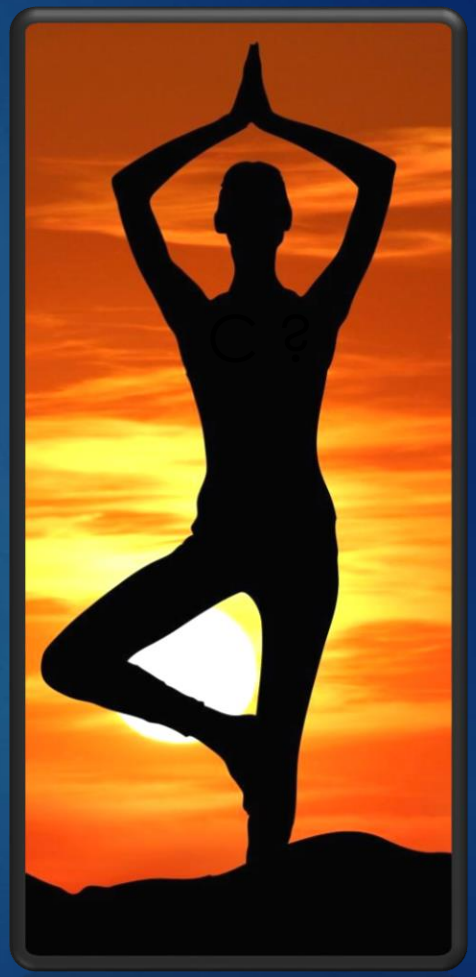


Example : client / server


```
//server
MasterOptions masterOptions;
masterOptions.setListen(« localhost »,8080);
MockMongoDriver mockMongo;
Master master(masterOptions, mockMongo);
master.run(WAIT_READY);

//client
GatewayOptions gatewayOptions;
gatewayOptions.setServer(« localhost », 8080);
Gateway gateway(gatewayOptions);
gateway.run();

//test
iolib.onRead(4096);
EXPECT_EQ(master.getIoiMetric(PID).readCnt, 100);
```



Example : client / server

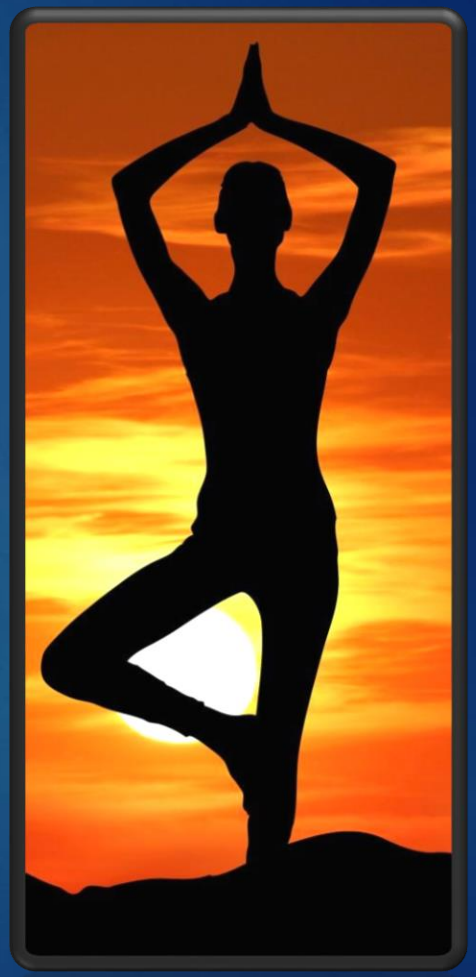

```
MockNetwork mockNetwork;

//server
MasterOptions masterOptions;
MockMongoDriver mockMongo;

Master server(masterOptions, mockNetwork, mockMongo);
server.run();

//client
GatewayOptions gatewayOptions;
Gateway gateway(gatewayOptions, mockNetwork);

//test
……
EXPECT_EQ(mockNetworkd.packetStats(), 1);
```



Frameworks, tools

Frameworks

- Unit tests / integration tests
  - C / C++ : Google Test /Google Mock
  - C : ~~Criterion~~
  - Python unit
- Schedule lot of integration tests on HPC clusters via Slurm
  - Jchronoss (schedule 10000 tests MPC test suite on Tera100 : 3000 cores)
    - http://jchronoss.hpcframework.paratools.com/
    - Initiated by Julien Adam during my PhD.
    - If you **need this** you might **have failed unit testing**, care about **high costs** !

Critics about criterion

- **Hard** to run easy in **GDB due to fork** (need gdb-server / connect)

```
#include <criterion/criterion.h>

Test(TestSuite, testCase)
{
    cr_assert_eq(out.bytes_read, 1);
    cr_assert_eq(out.bytes_read, 1, “Bla bla bla %d“  out.bytes_read);
}
```

```
CMakeFiles/          test-libsbb-ioproxy  test-libsbb-metrics  test-pr-pool        test-pt-pool
root@ioi-install:/tmp/test/sbb/build# ./libsbb/tests/test-libsbb-metrics
[----] /tmp/test/sbb/libsbb/tests/test-libsbb-metrics.c:43: Assertion failed: The expression (out.bytes_read) == (1) is false.
[FAIL] libsbb_metrics::init: (0.00s)
[====] root@ioi-install:/tmp/test/sbb/build#
```

Lets look Google Test

- I would **suggest** Google Test
- No auto fork (easy gdb)
- Parallel run
- C++ => **nice auto formatting**

```
#include <googletest/gtest.h>

Test(BuilderUnitData, assign) {
    MachinTruc out;
    out.action();

    ASSERT_EQ(out.bytes_read, 1);
    EXPECT_EQ(out.bytes_read, 1);
    EXPECT_EQ(out.bytes_read, 1) << “My super message !”;
}
```

```
[       OK ] BuilderUnitData.constructor (1 ms)
[ RUN      ] BuilderUnitData.assign
/home/valats/Projects/lhcb-daqpipe-v2/src/units/tes
Value of: data.getCreditId(i)
  Actual: 1
Expected: i+2
Which is: 2
/home/valats/Projects/lhcb-daqpipe-v2/src/units/tes
Value of: data.getCreditId(i)
  Actual: 2
Expected: i+2
Which is: 3
[  FAILED  ] BuilderUnitData.assign (0 ms)
[ RUN      ] BuilderUnitData.unassign
```

Self testing : asserts


```
int reverseDefaultFreeSizes(Size size,const Size * sizeList,int nbLists)
{
    //errors
    assert(sizeList == cstDefaultFreeSizes);
    assert(64 >> 5 == 2);
    assert(sizeList[45] == -1UL);
    assert(size >= 16);

    //implem
    …
}
```

Some infrastructure tricks

On remark… building unit tests

- Do **not repeat yourself**
- Avoid putting again sources files in every binary
- Build **two** libs
  - one for **final target**
  - one **for test**
- Link once

```
#build test for net
#Re-aggregate part of source files ?
add_executable(test-net-tcp test-net-tcp.c
               ../proto-tcp.c
               ../proto-tcp-pools.c
               ../net.c
               #mocking via function erasing v                    ?
               fake-machin.c
               fake-bidule.c
               glue.c)

#link again everything ? What is add a new dependency ?
target_include_directories(test-net-tcp PRIVATE ${ZMQ_INCLUDE_DIRS})
target_include_directories(test-net-tcp PRIVATE ${B1_INCLUDE_DIRS})
target_include_directories(test-net-tcp PRIVATE ${C2_INCLUDE_DIRS})
target_include_directories(test-net-tcp PRIVATE ${D3_INCLUDE_DIRS})
target_link_libraries(test-net-tcp PRIVATE ${ZMQ_LIBRARIES})
target_link_libraries(test-net-tcp PRIVATE ${B1_LIBRARIES})
target_link_libraries(test-net-tcp PRIVATE ${C2_LIBRARIES})
target_link_libraries(test-net-tcp PRIVATE ${D3_LIBRARIES})

#why not ctests 'add_test' simple semantic ?
add_custom_target(run-net-tcp-test ……)
```

On remark… building unit tests

- Do **not repeat yourself**
- Avoid putting again sources files in every binary
- Build **two** libs
  - one for **final target**
  - one **for test**
- Link once

```
set(SOURCES prot-tcp.c proto-tcp-pool.c proto-udp.c
            net.c test-net.c)

add_library(net SHARED ${SOURCES})
add_library(net-for-utests SHARED ${SOURCES})

add_library(net dl)
add_library(net-for-utests dl)
#...
target_link_libraries(net ${ZMQ_LIBRARIES} ${WRAPPER_LIB})
target_link_libraries(net-for-utests ${ZMQ_LIBRARIES})
```

```
#build test
add_executable(test-net-tcp test-net-tcp.c)
target_link_libraries(test-net-tcp net-for-utests)
add_test(test-net-tcp test-net-tcp)
```

A word on unit testing static

- Also unit test your static functions
- If you **built two libs** that's easy.
  - Final one with static
  - Test one (not installed) without
- Use a MALT_STATIC macro instead of static keyword
  - Enabled for final software
  - Disabled for unit test intermediate lib

```
#ifdef DISABLE_STATIC
   #define MALT_STATIC
#else
   #define MALT_STATIC static
#endif

static malt_my_local_func(void);
MALT_STATIC malt_my_local_func(void);
```

One word on mutable testing

Quality of my test ?

- This is **more a theoretical topic** (a least for me)
- I never really used except tests made last **Friday in the train**…
- **Idea**:
  1. Generate **code mutation**
  2. **Run** unit **tests**
  3. Did we **detect** the change **?**

Example of run


```
diff --git a/src/sctk_allocator.c b/src/sctk_allocator.c
index 7321e6d..d80ff09 100644
--- a/src/sctk_allocator.c
+++ b/src/sctk_allocator.c
@@ -1728,7 +1728,7 @@ SCTK_PUBLIC void * sctk_alloc_chain_realloc(struct sctk_alloc_chain * chain, voi
                        //need to reallocate a new segment and copy the old data
                        res = sctk_alloc_chain_alloc(chain,size);
                        //copy the data
-                       if (res != NULL)
+                       if (res == NULL)
                        {
                                copy_size = (old_size < size) ? old_size : size;
                                memcpy(res,ptr,copy_size);
================ SUCCESS ================
```

Example of results on 300 mutations

Project	Remark	Code lines	Coverage	Detection
MPC_Allocator	C	14700	70%	??
MPC_Allocator	No assert			
hpc_alloc_rust	Rust	8441	91%	??
hpc_alloc_rust	No assert			??
lhcb-daqpipe	C++	26000	70% (42%)	??

Example of results on 300 mutations

Project	Remark	Code lines	Coverage	Detection
MPC_Allocator	C	14700	70%	**25%**
MPC_Allocator	No assert			
hpc_alloc_rust	Rust	8441	91%	**44%**
hpc_alloc_rust	No assert			**42%**
lhcb-daqpipe	C++	26000	70% (42%)	**40%**

Conclusion

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**
- Hope you better unit test philosophy
- **Even more true** to target **performance**
  - Always need to change design to follow architectures
  - Make tons of performance mistake we need to fix
- I can make another talk on more technical details and time metrics

BACKUP

On remark… building unit tests

- Do **not repeat yourself**
- Avoid putting again sources files in every binary
- Build **two** libs
  - one for **final target**
  - one **for test**
- Link once

```
set(SOURCES prot-tcp.c proto-tcp-pool.c proto-udp.c
            net.c test-net.c)

add_library(net SHARED ${SOURCES})
add_library(net-for-utests SHARED ${SOURCES})

add_library(net dl)
add_library(net-for-utests dl)
#...
target_link_libraries(net ${ZMQ_LIBRARIES} ${WRAPPER_LIB})
target_link_libraries(net-for-utests ${ZMQ_LIBRARIES})

target_include_directories(test-net-tcp PRIVATE ${C2_INCLUDE_DIRS})
target_include_directories(test-net-tcp PRIVATE ${D3_INCLUDE_DIRS})
target_link_libraries(test-net-tcp PRIVATE ${ZMQ_LIBRARIES})
target_link_libraries(test-net-tcp PRIVATE ${B1_LIBRARIES})

#build test
add_executable(test-net-tcp test-net-tcp.c)
target_link_libraries(test-net-tcp net-for-utests)
add_test(test-net-tcp test-net-tcp)
```