Lessons Learned From TA Practices

Xiao Shiliang-Shelwin (肖世良)

2017.05.25

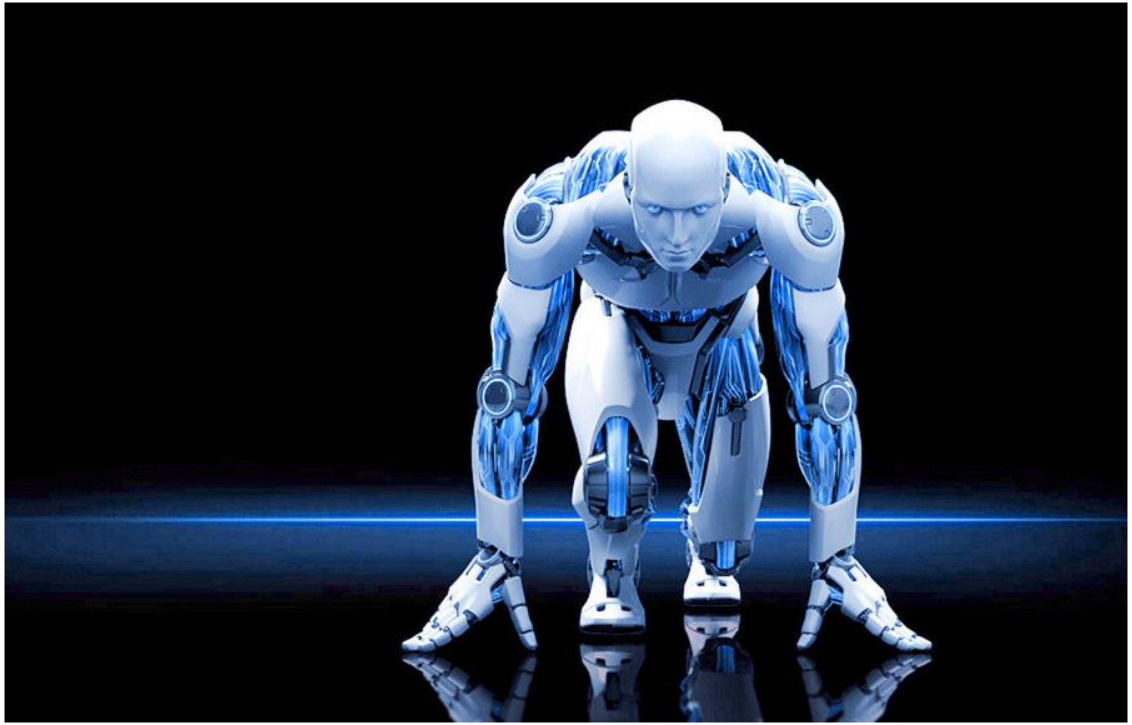
TA: Test Automation

“Use of special software to control the execution of software tests and the comparison of actual outcomes with predicted outcomes”

—— Wiki

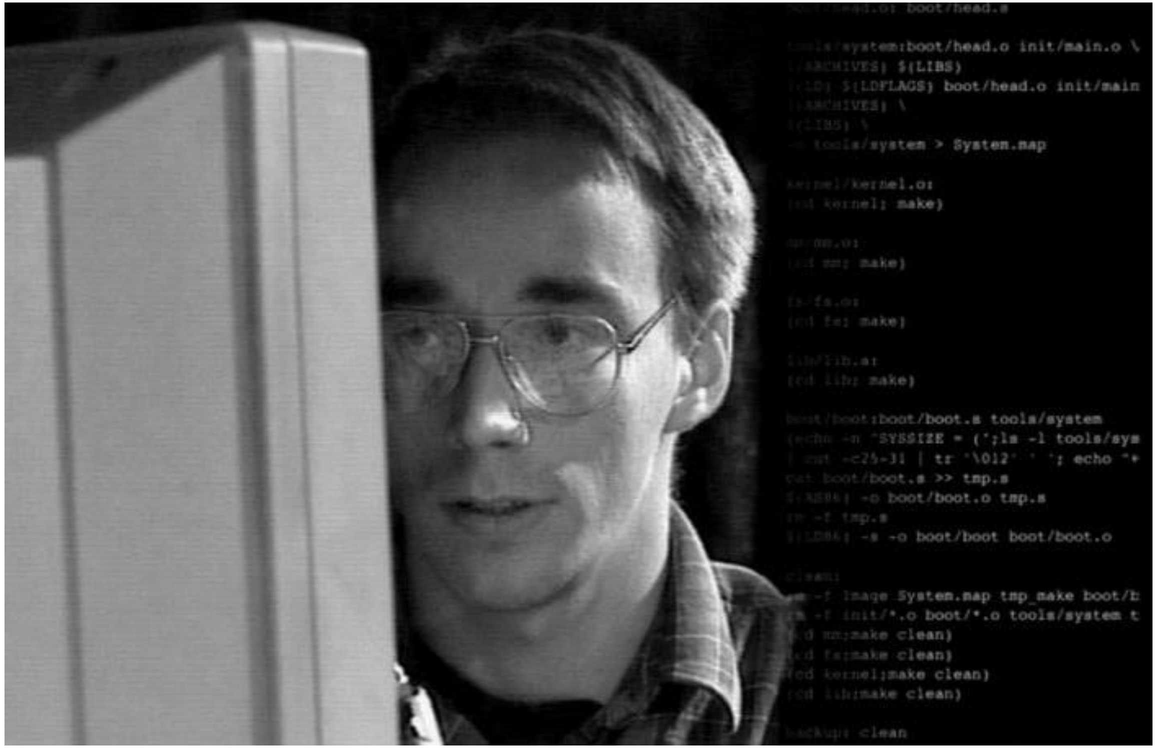
“Let computer do software testing for human”

—— Anonymous



Essence of Test Automation

Test Automation is
Software Development



```
boot/head.o: boot/head.s

tools/system:boot/head.o init/main.o \
$(ARCHIVES) $(LIBS)
$(LD) $(LDFLAGS) boot/head.o init/main
$(ARCHIVES) \
$(LIBS) \
-o tools/system > System.map

kernel/kernel.o:
(cd kernel; make)

mm/mm.o:
(cd mm; make)

fs/fs.o:
(cd fs; make)

lib/lib.a:
(cd lib; make)

boot/boot:boot/boot.s tools/system
(echo -n "SYSSIZE = (";ls -l tools/sys
| cut -c25-31 | tr '\012' ' '; echo "*
cat boot/boot.s >> tmp.s
$(AS86) -o boot/boot.o tmp.s
rm -f tmp.s
$(LD86) -s -o boot/boot boot/boot.o

clean:
rm -f Image System.map tmp_make boot/b
rm -f init/*.o boot/*.o tools/system t
(cd mm;make clean)
(cd fs;make clean)
(cd kernel;make clean)
(cd lib;make clean)

backup: clean
```

Content

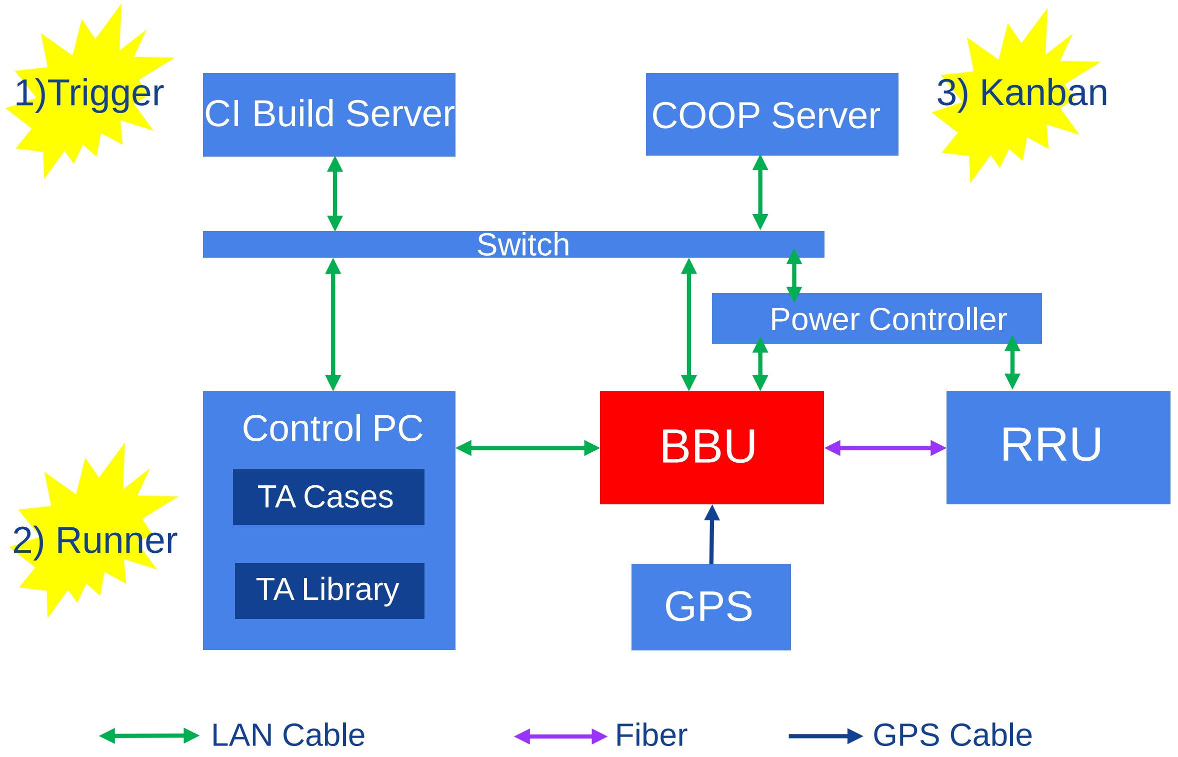
- Practice: TDLTE BTS CRT
- Practice: BTSMED ET

- Lesson: What a good TA is
- Lesson: How to achieve a good TA

TA Practice

- **Project**: TDLTE BTS CRT (Continuous Regression Testing)
- **Time**: 2016.03 —— 2016.06
- **Participant**: me

BTS CRT

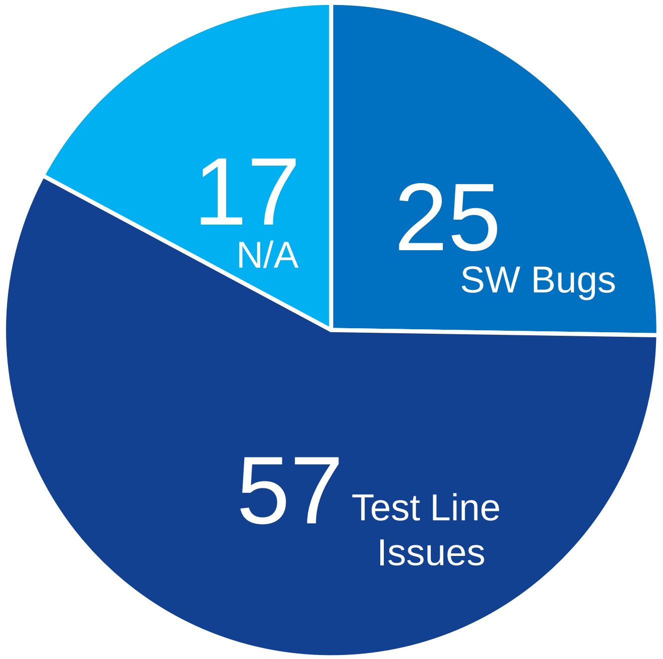


BBU: baseband unit **RRU**: remote radio unit **CI**: continuous integration

Test Summary

6	50+	2~4	5000+
test lines	times/day	builds/day	times in 3-months

Issue Summary

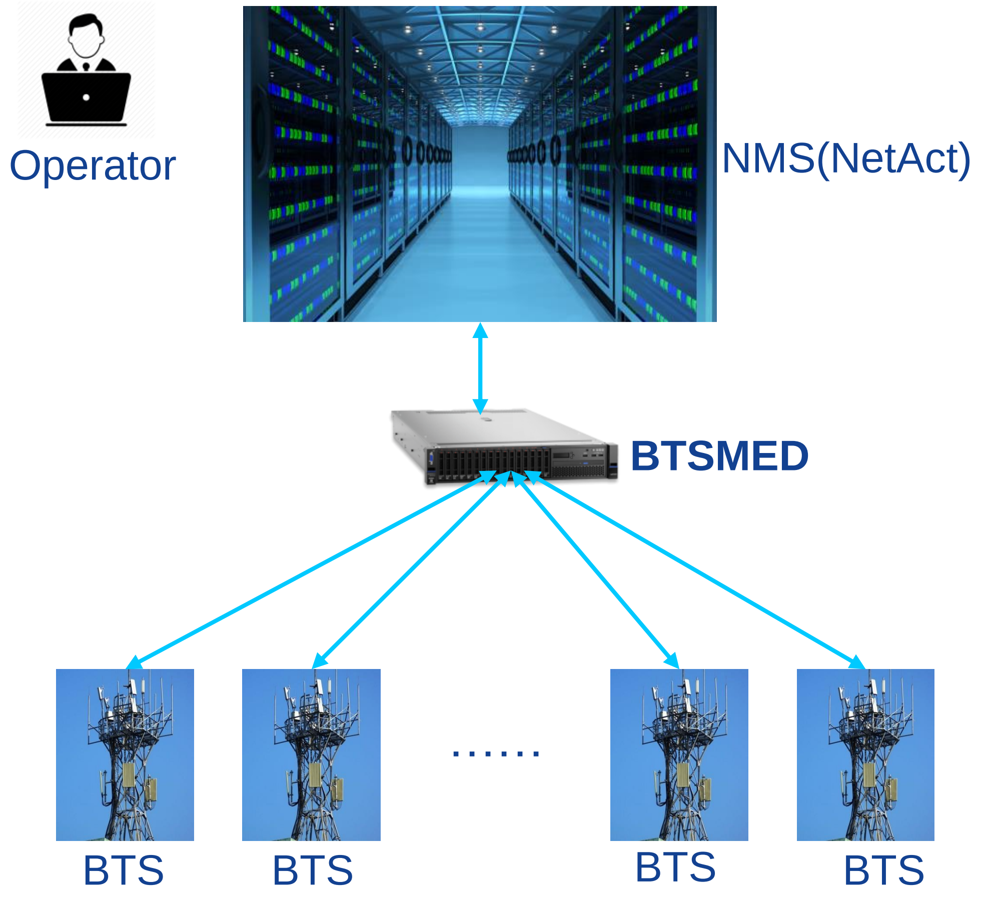


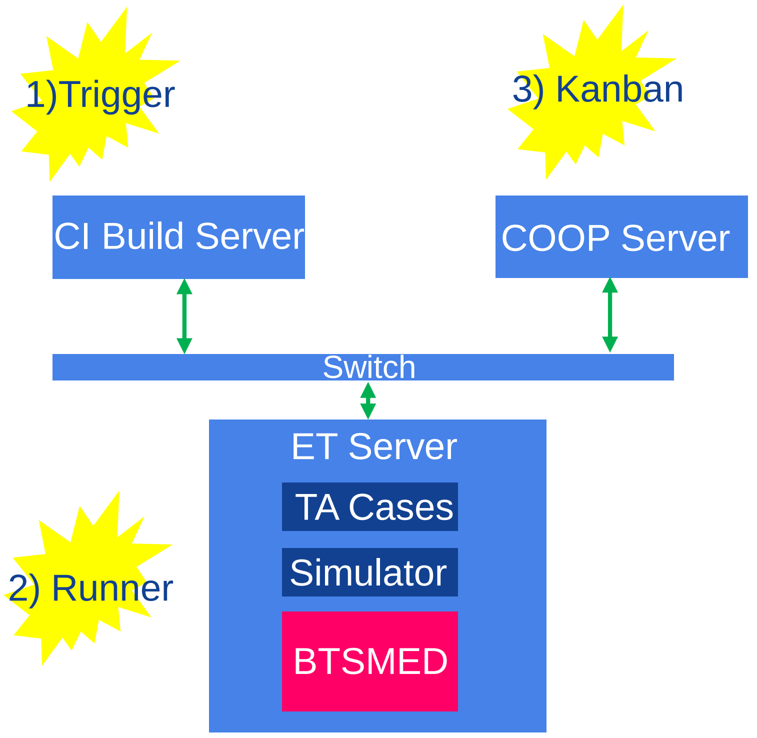
I lost in thinking ……

TA Practice

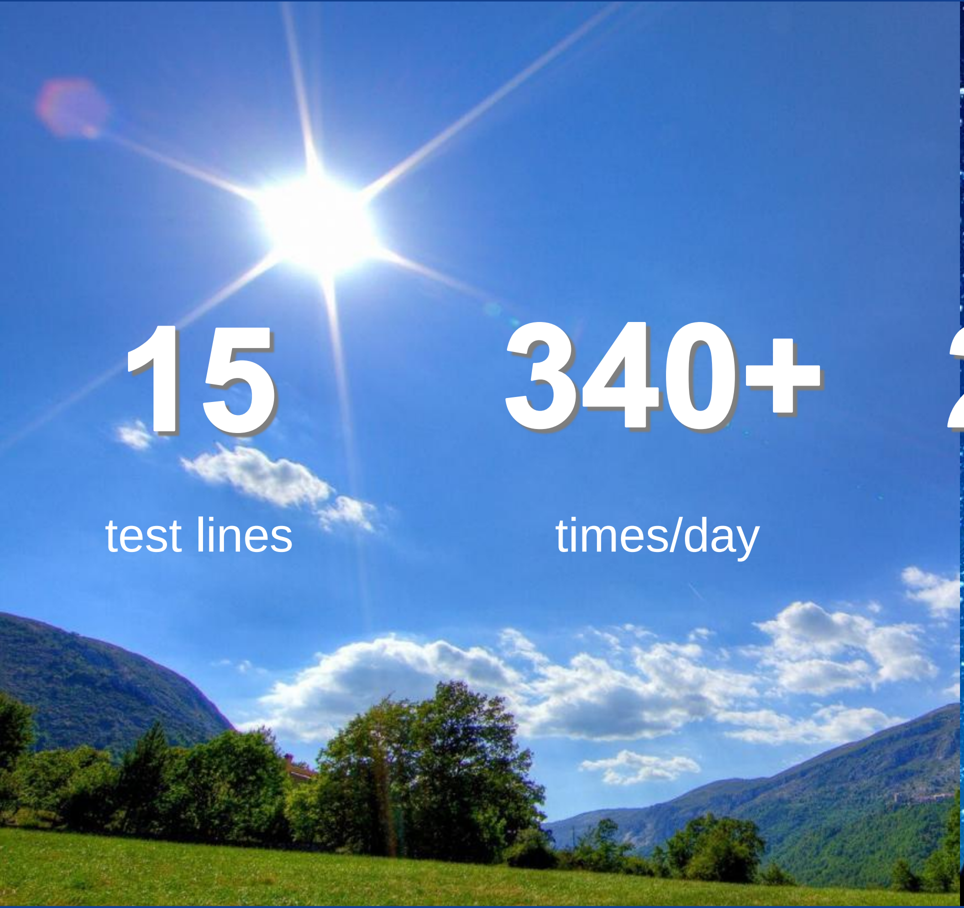
- **Project**: SRAN BTSMED ET (Entity Testing)
- **Time**: 2016.08 —— now
- **Participants**: Dev HZ3 FV1 team, led by Ye Jason.

BTSMED ET


Operator
NMS(NetAct)
BTSMED
BTS
BTS
......
BTS
BTS


1)Trigger
3) Kanban
CI Build Server
COOP Server
Switch
ET Server
TA Cases
Simulator
BTSMED
2) Runner

Test Summary



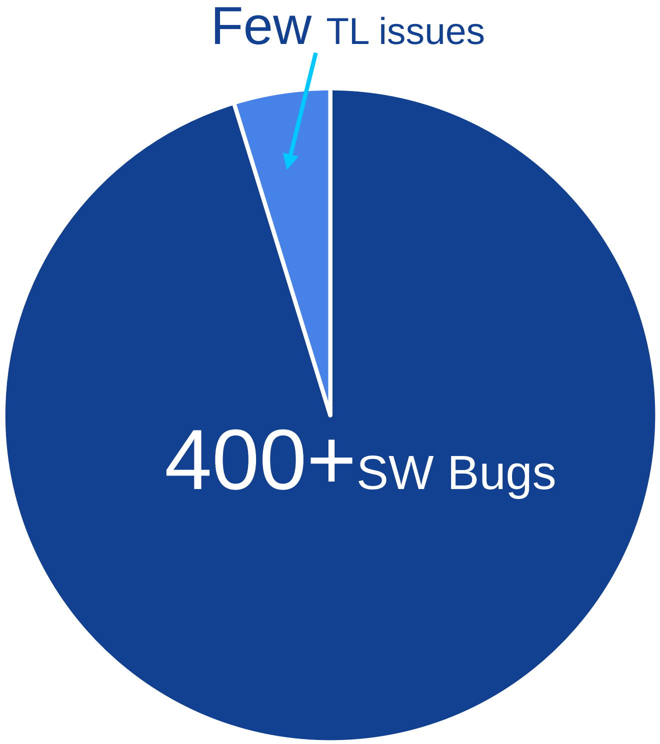
15 test lines

340+ times/day

250 builds/day

5000+ times in 1-months

Issue Summary



I lost in thinking, again ……

Lessons Learned

What a good TA is

A not-so-good TA

Cannot find SW bugs efficiently

Find many TA issues

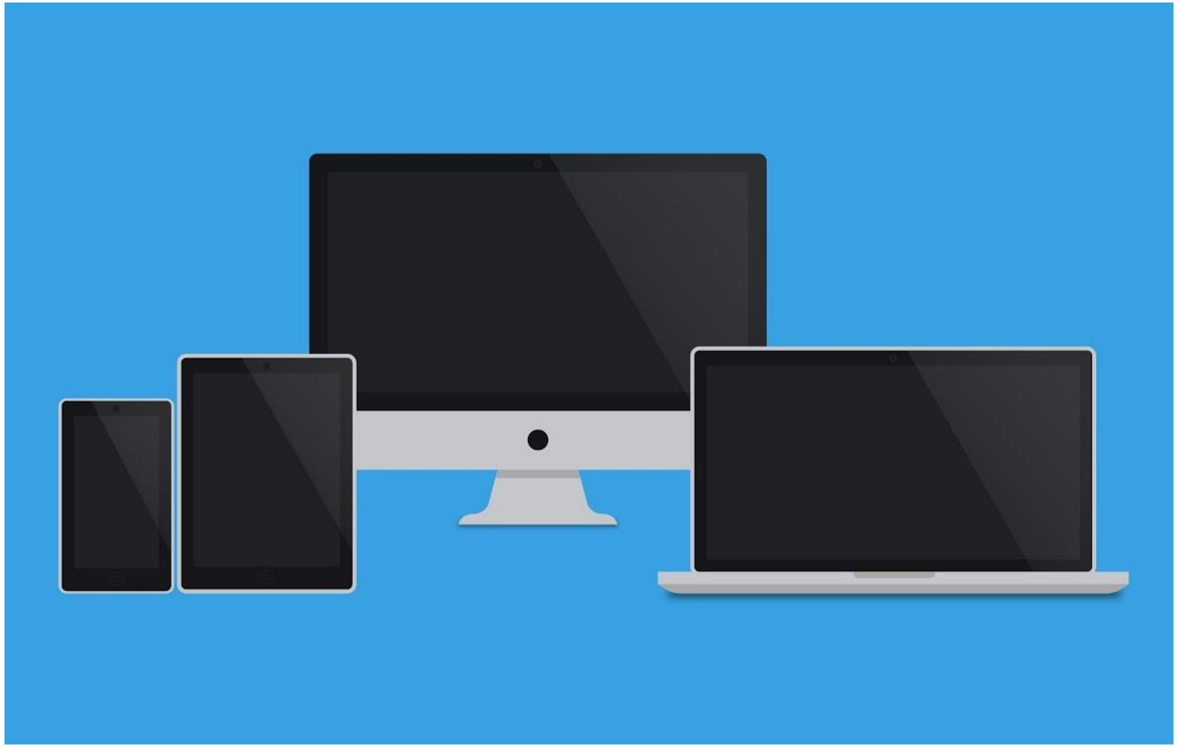
Test ENVs are unstable

Test lib/cases are hard to maintain

A Good TA

Simple & Reliable

(equivalent to Efficient & Productive)



TA Simplicity

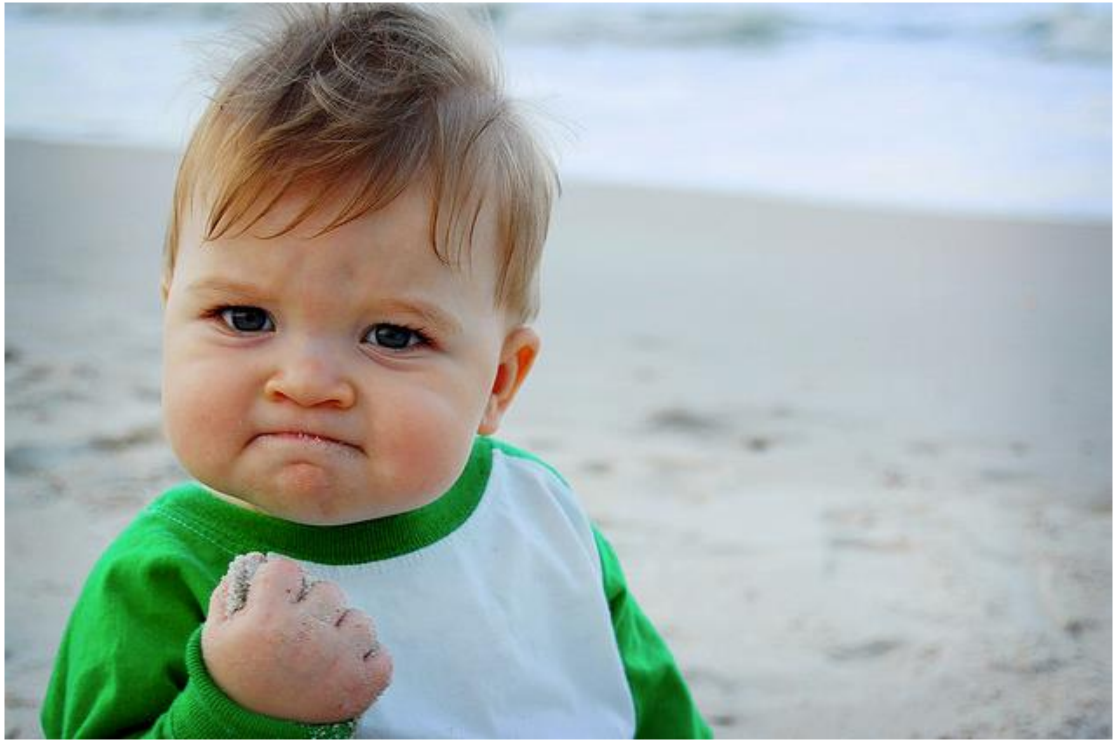
How much efforts are needed to develop & maintain TA libraries and TA cases?

(equivalent to Efficiency)

TA Reliability

How much confidence do we have about that case failure is caused by SW bugs, not by TA itself?

(equivalent to Productivity)



Lessons Learned

How to achieve a good TA

My Eight Proposals……

Achieving A Good TA

[P1] Add *TA Grooming* as part of software testing process.

TA Grooming

TA or not TA for each case?

New lib/keyword requirement?

Action plan

Start TA Grooming As Early As Possible

Achieving A Good TA

[P2] Add *TA Case Review* as part of software testing process.

TA Case Review

Cases be as readable as requirement docs

All cases follow common paradigms

Large-screen meeting review

Everybody involved

Achieving A Good TA

[P3] *Test* of Test Automation is needed, especially for TA library/tools.

A Good Example: NBS

NBS: Newbie Simulator

“Simulating NetAct & SOAM BTS for BTSMED Entity Testing & Performance Testing”

http://gitlab.china.nsn-net.net/ta/nbs

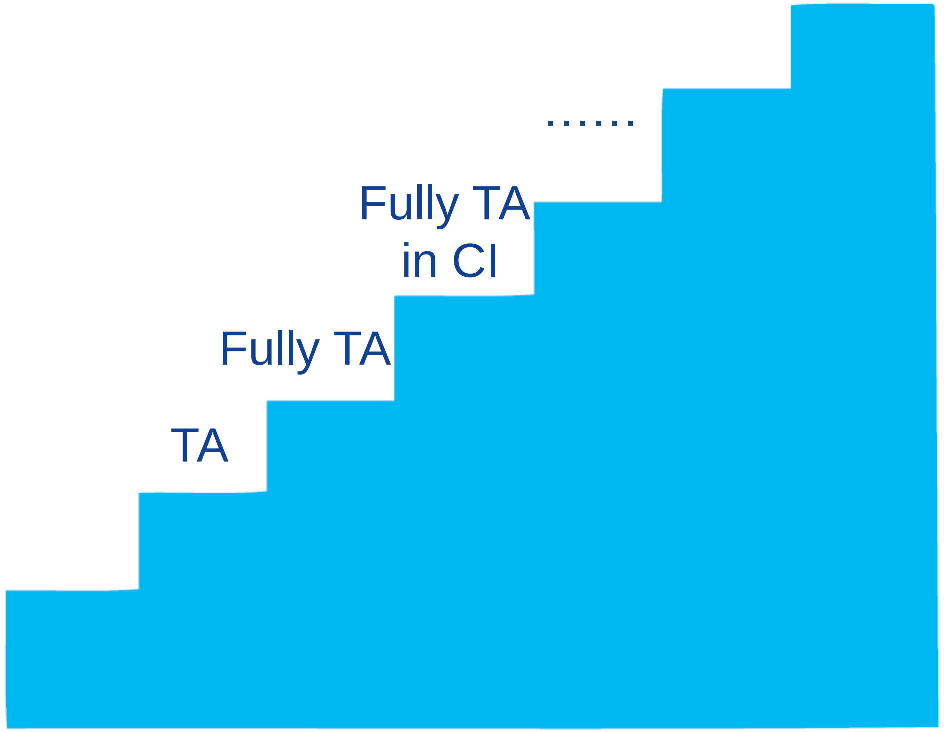
A Good Example: NBS

1	212	~20	2146	257
mocked BTSMED	unit test cases	seconds	commits	versions

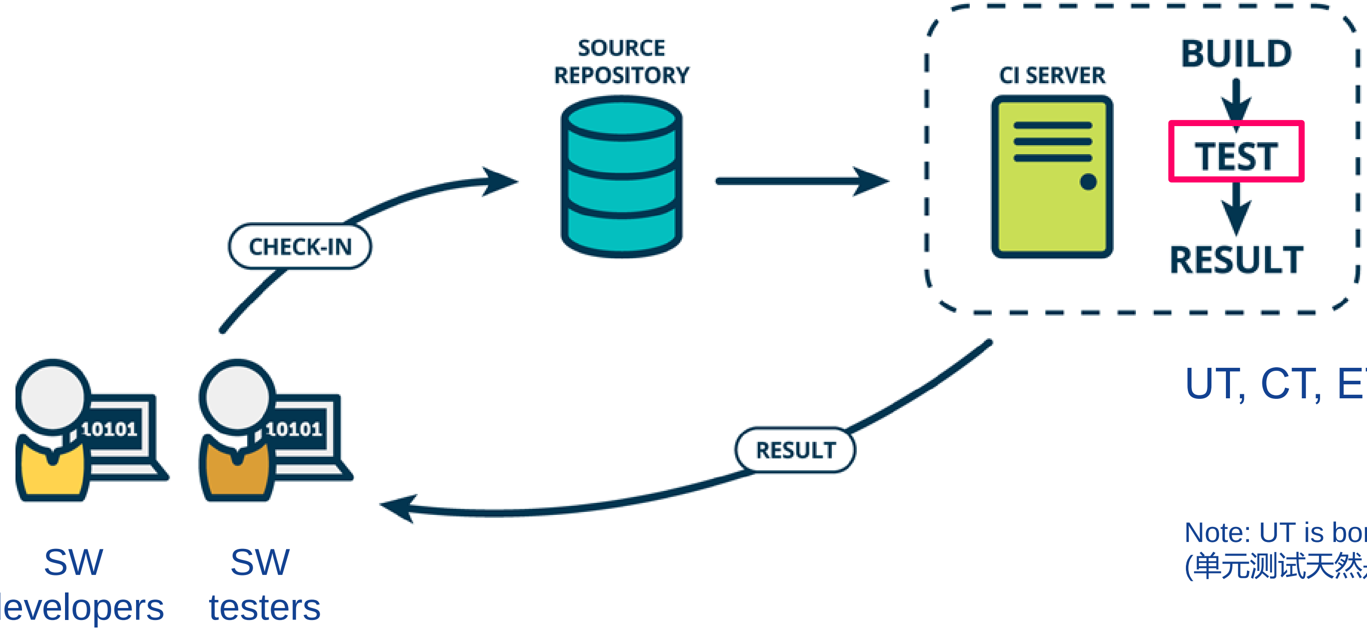
Our response time to issue pre-check/bug fixing is in hours, NOT in days

Achieving A Good TA

[P4] Add *fully* automated testing into continuous integration (CI) system.



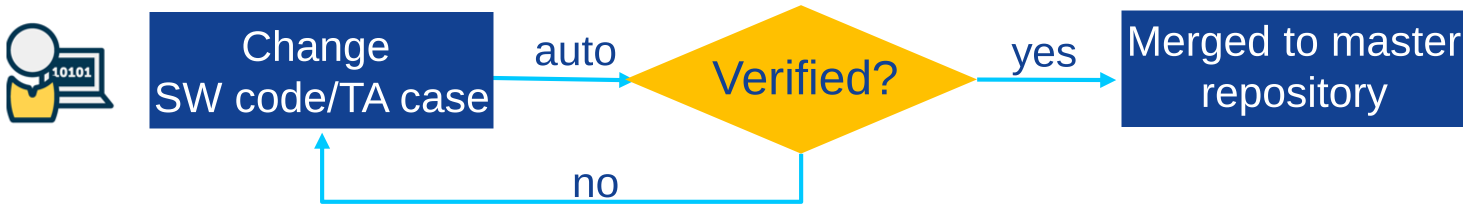
Add Fully TA into CI



UT, CT, ET, IT, ST, …

Note: UT is born with automation
(单元测试天然是自动化的)

A more aggressive example: BTSMED ET



ET is part of verification

NOBODY can bring change to master repository with a failed ET !

But it is NEVER an easy work ……

Nevertheless, it is worth doing since “QUALITY MATTERS”

CBRT: change-based regression testing in BTSMED ET

$$340 = \mathbf{250} \times 1 + 15 \times 6$$

$\downarrow$

Whenever there is a SW change, there is an ET verification

$\downarrow$

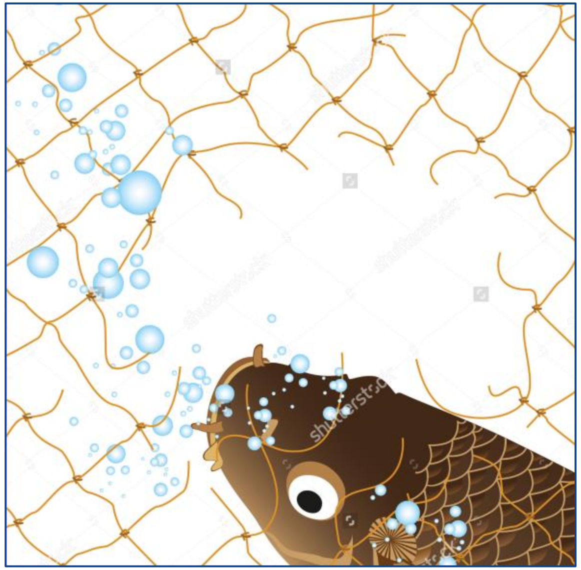
Find SW bugs efficiently, Find SW bugs ***early***

Achieving A Good TA

[P5] Continuously improve TA by RCA/EDA.

RCA/EDA of Test Automation

1. For each issue proven to be TA bug, do RCA (root cause analysis)



2. For each SW bug found by post test stages, do EDA (escaped defect analysis)

First of all, reproducing bug by changing test code

Achieving A Good TA

[P6] TA Left-shift: strengthen automation of *early* test stages.

Necessity(必要性) of TA Left-shift

The Google Testing Law (谷歌测试定律)：

"As SW test proceeds(UT->CT ->IT->ST or small->medium-> large test), the cost of fixing a discovered SW bug increases at an exponential scale".

Feasibility(可行性) of TA Left-shift

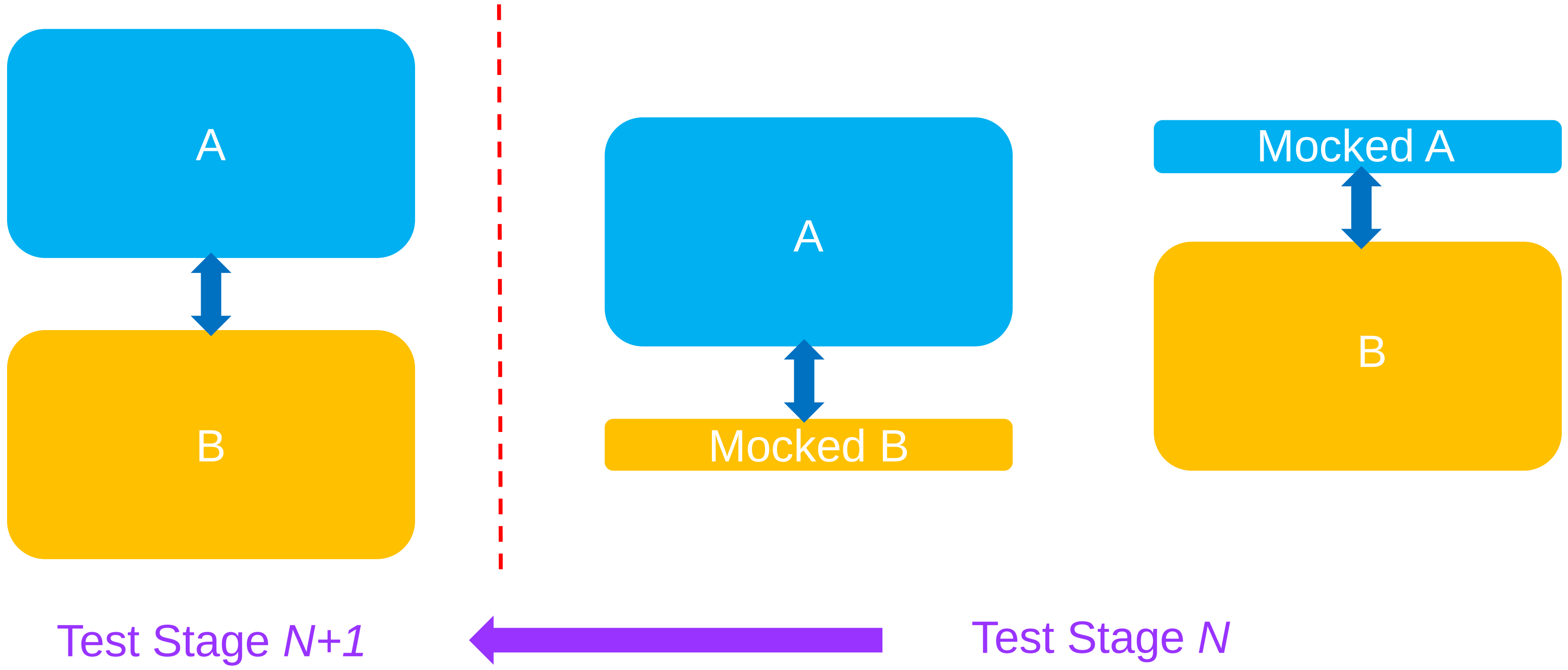
The Testing Coverage Law (测试覆盖定律)：

"For multi-stages SW testing, any SW bug discovered at the current test stage, could have been discovered at the former stage by increasing or modifying one test case"

Achieving A Good TA

[P7] Use Mock technique as much as possible.

What is Mock ?



Benefits of Mock

Focus on the tested object

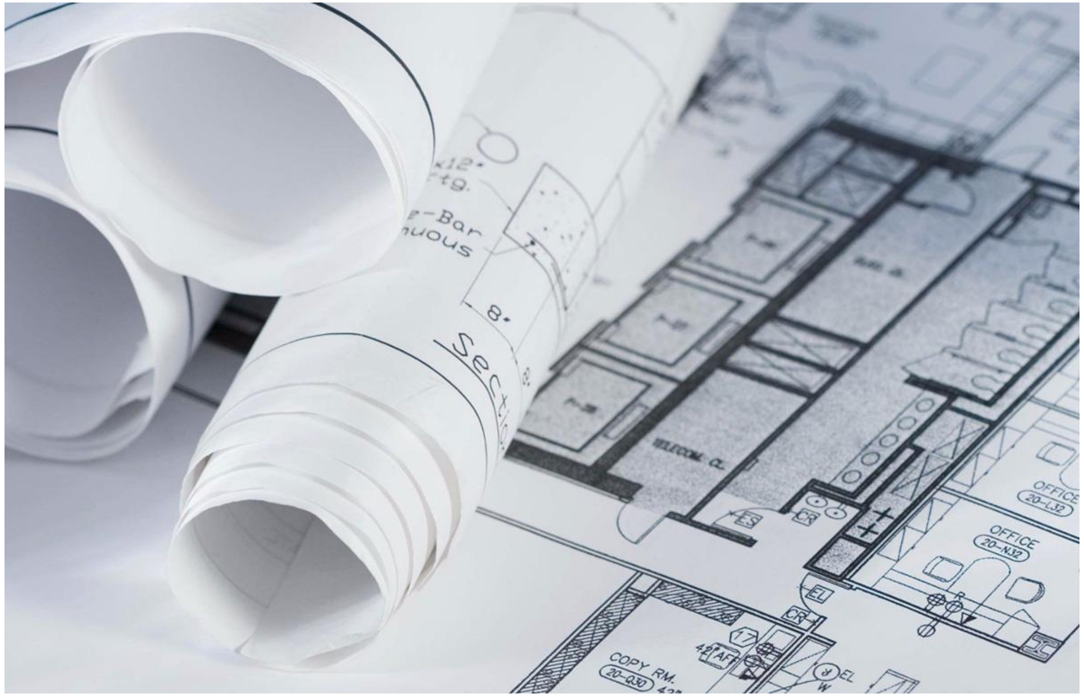
Starts test early

Test ENVs easily copied

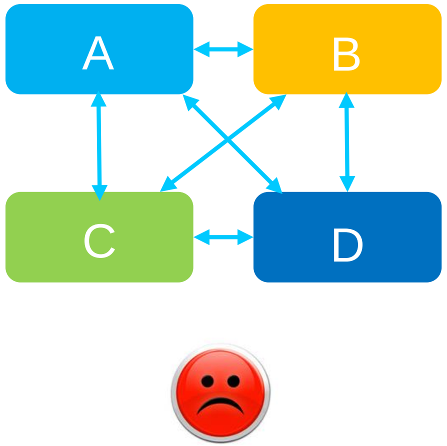
Cheap since only mocking interface

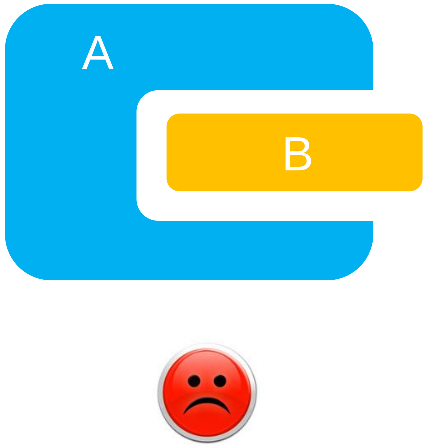
Achieving A Good TA

[P8] Take *testability* into account when designing software.

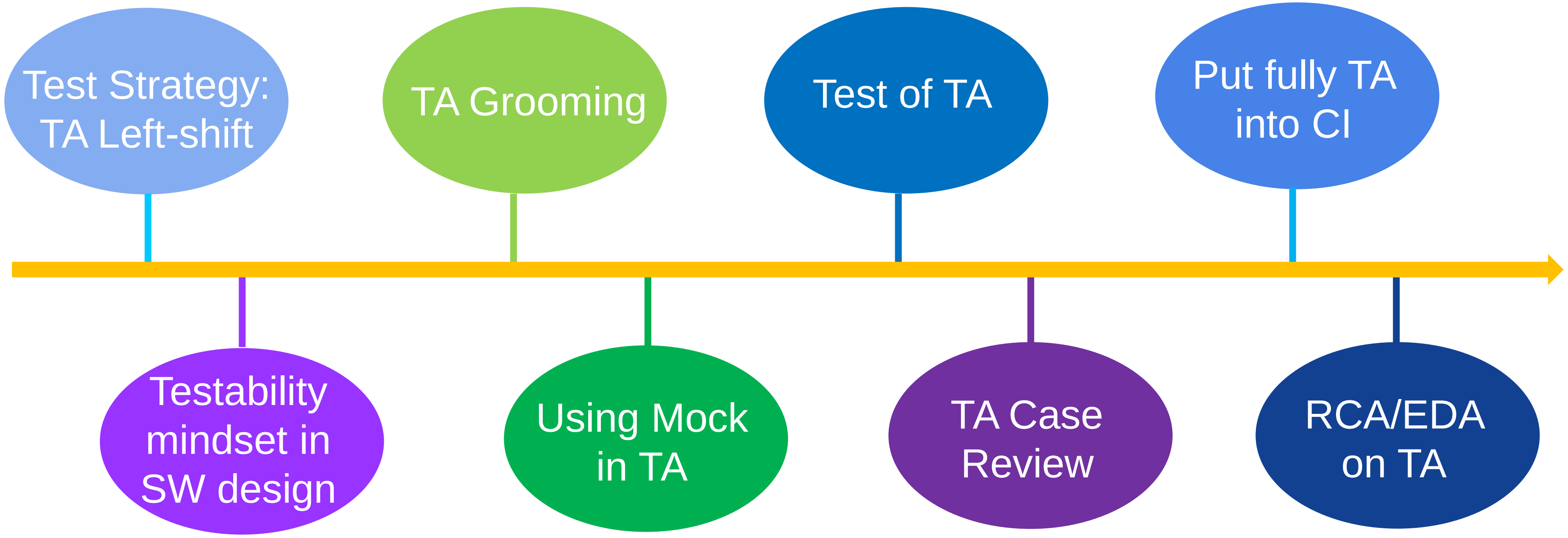


Some Examples of Bad Testability





Summary of my proposals towards a good TA



To Repeat……

Test Automation is
Software Development

Learn More

My Blog

GTAC

Q & A