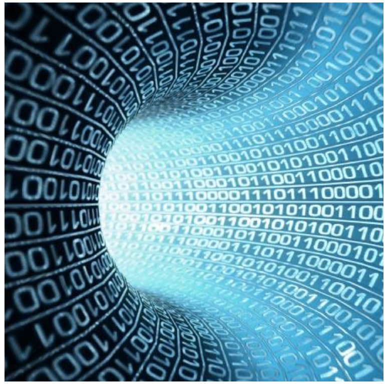
# Redefine SW Test in Telecommunications: Takeaway Messages Toward a Better Test Automation

Xiao Shiliang-Shelwin, PhD

2017.05

NOKIA 上海贝尔

# SW Test in Telecommunications



Tested SW is complex in nature

Distributed SW dev & test

UT CT MT

ET IT ST

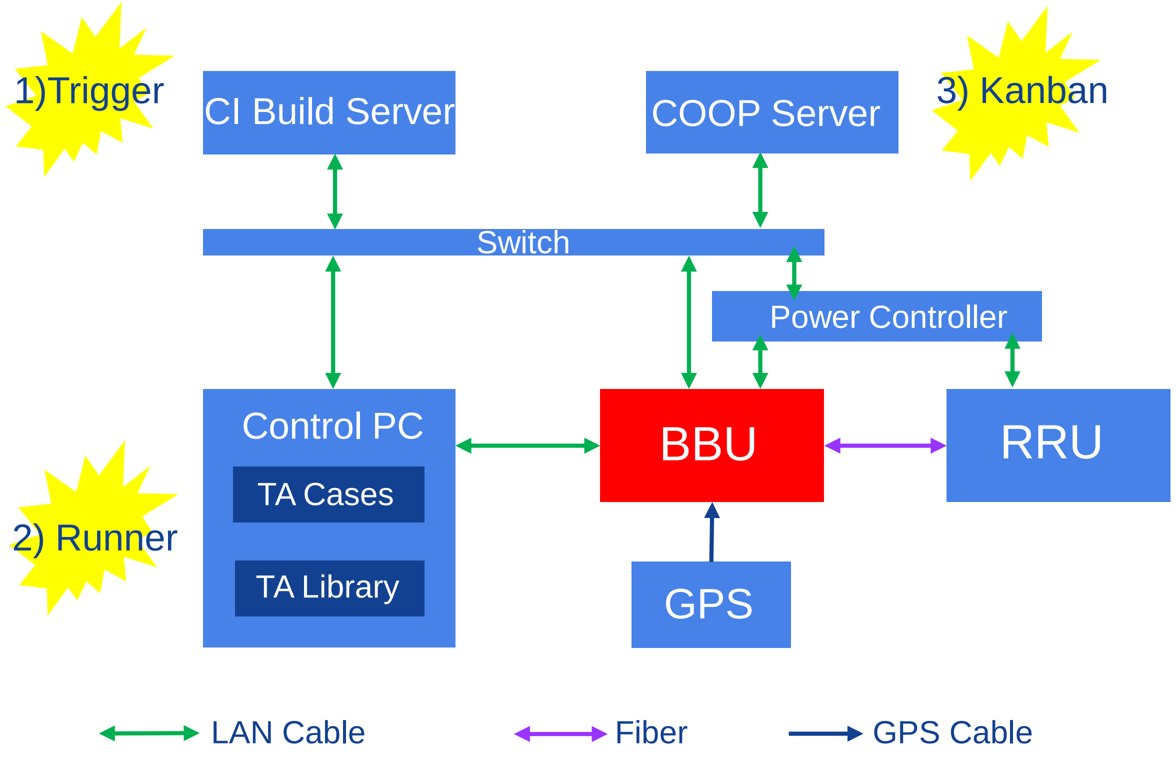
QT CIT CRT

PET KPI ……

Broad-spectrum of testing flow

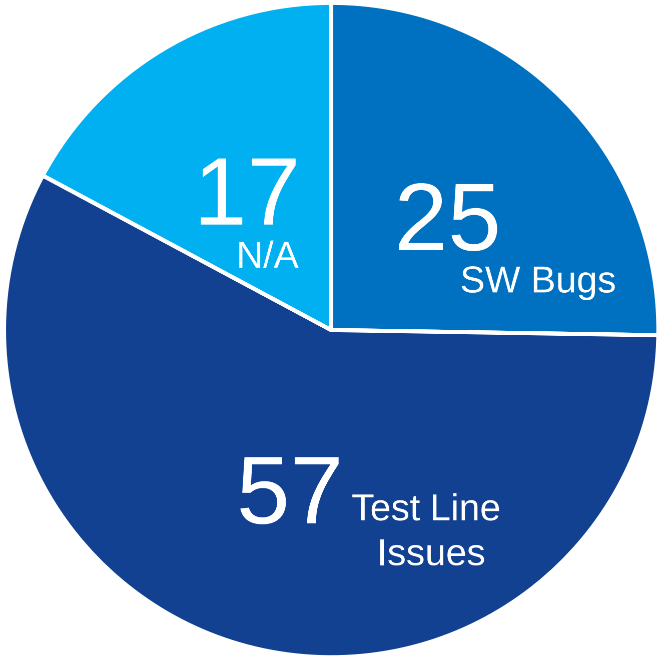
Quality really does matter

# Personal Practice(a): BTS CRT, from 2016/03 to 2016/06

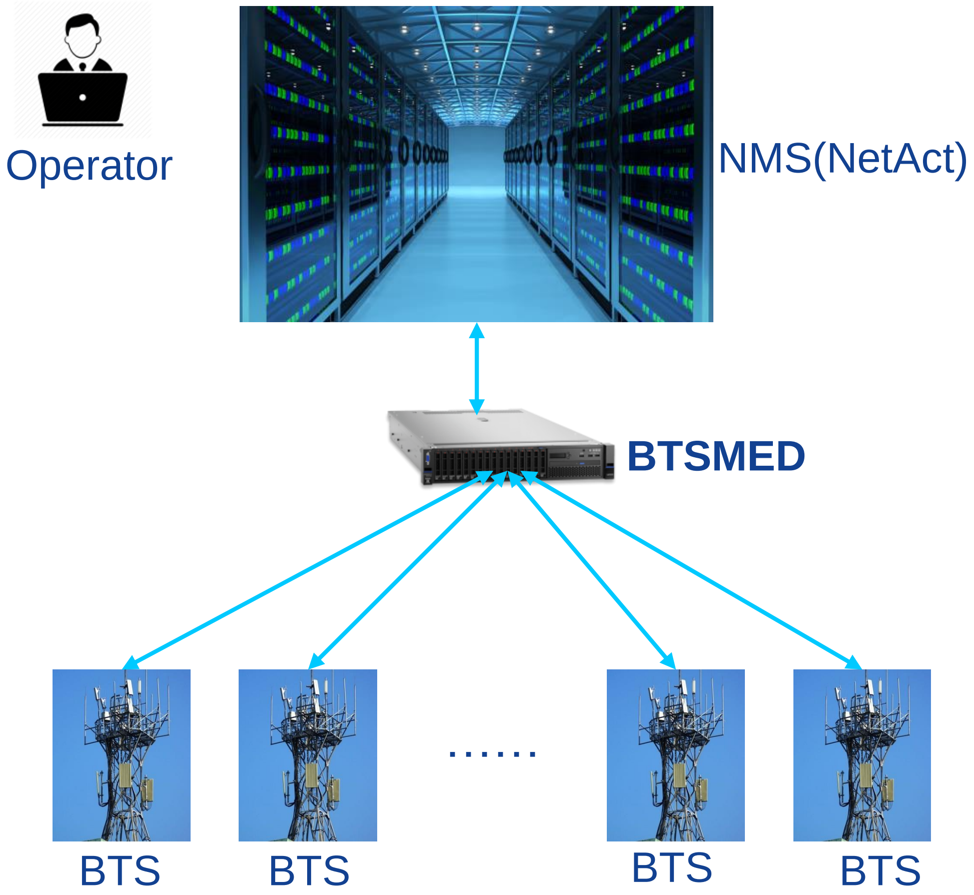


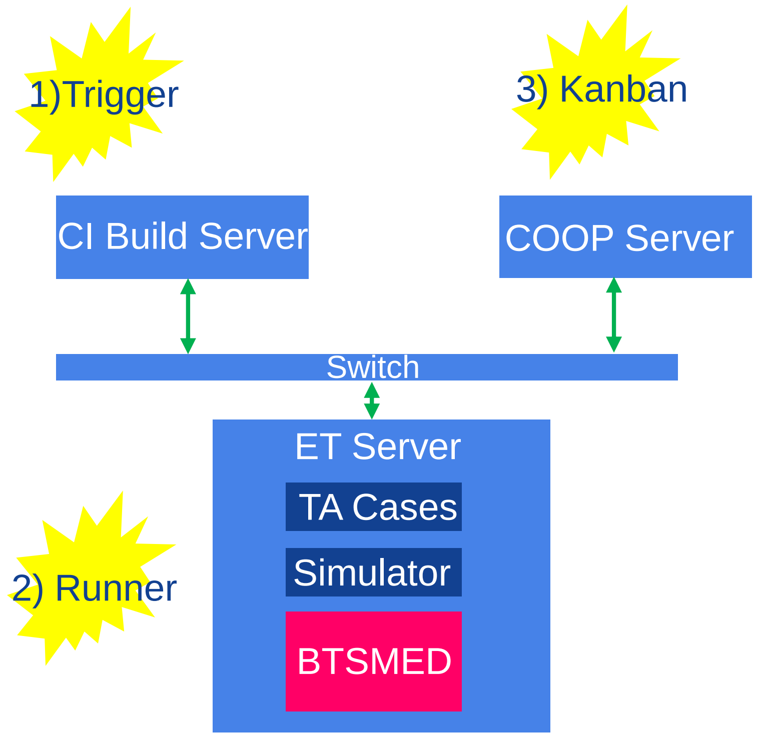
**BBU**: baseband unit **RRU**: remote radio unit **CI**: continuous integration

## During 3-months' testing:

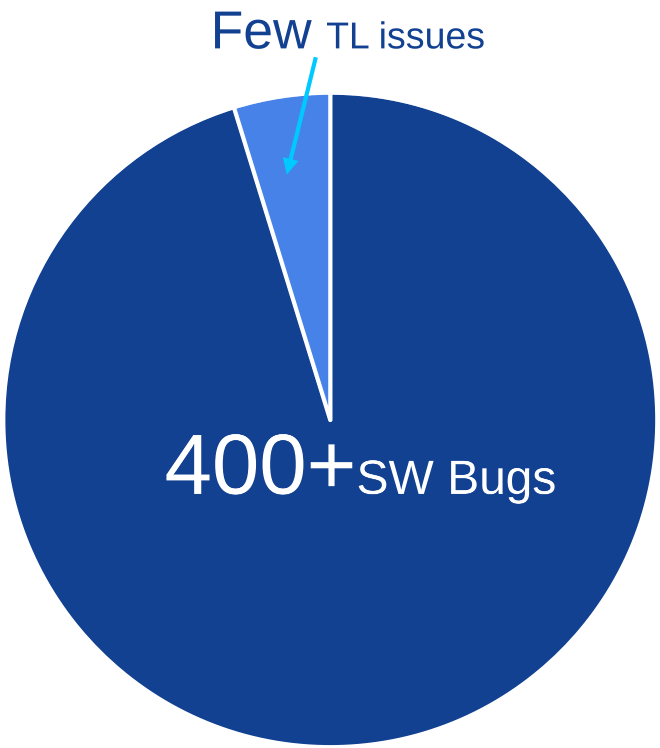


# Personal Practice(b): BTSMED ET, from 2016/10 to 2017/06





## During 8-months' testing:



## I lost in thinking

## 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**

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

**RELIABLE**

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

## Eight Takeaway Messages on A Better TA

## Achieving A Good TA

**(1) Add *TA Grooming* as part of software testing process.**

## TA Grooming

TA or not TA for each case?

New lib/keyword requirement?

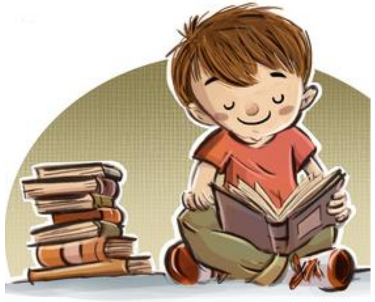
Action plan

***Start TA Grooming As Early As Possible***

## Achieving A Good TA

**(2) 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

**(3) *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”*

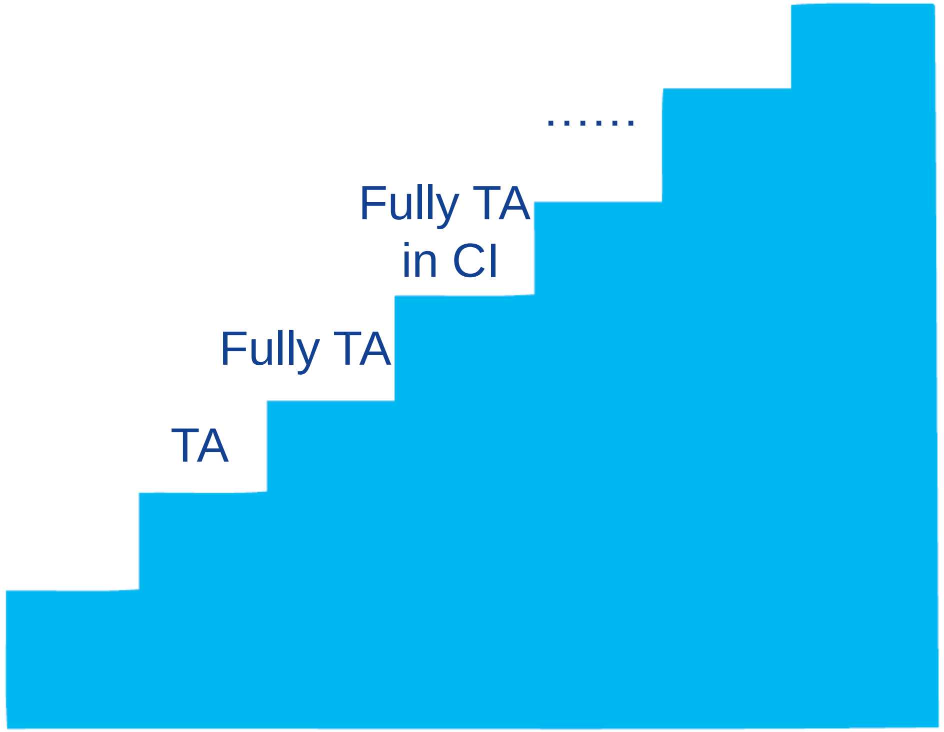
http://gitlab.china.nsn-net.net/ta/nbs

## A Good Example: NBS

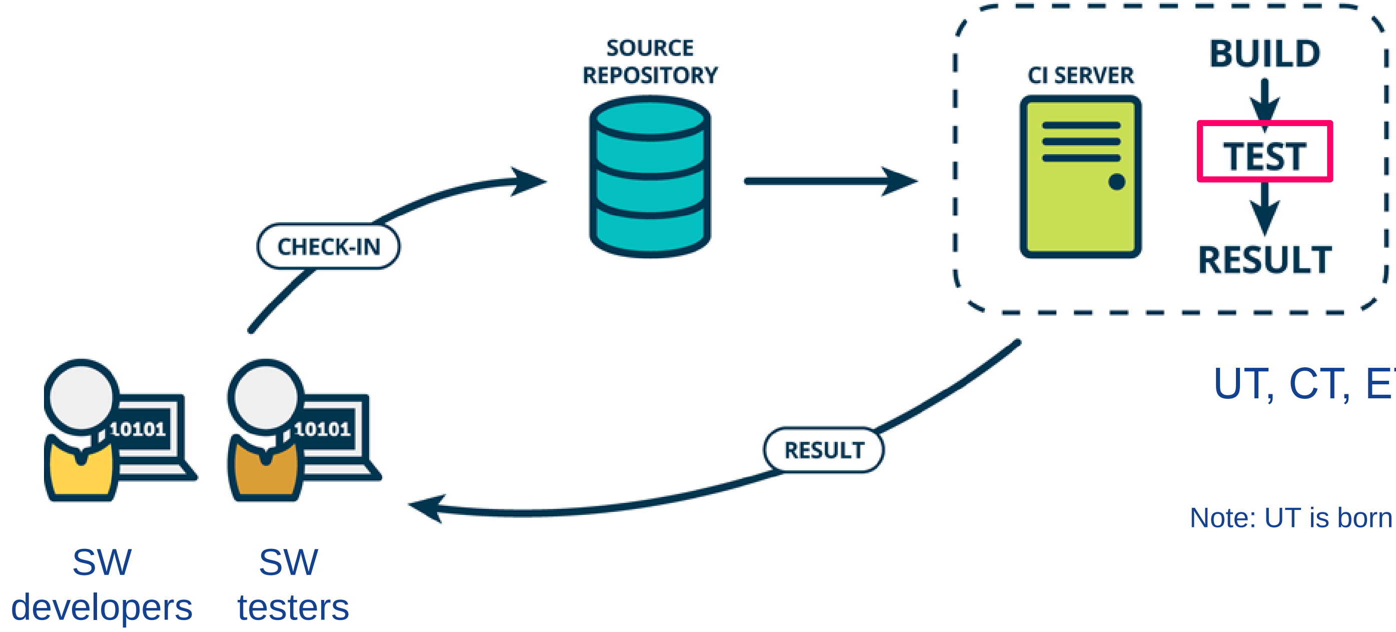
| | | | | |
|---|---|---|---|---|
| **1** | **212** | **~20** | **2146** | **257** |
| mocked BTSMED | unit test cases | seconds | commits | versions |

## Achieving A Good TA

**(4) Add *fully* automated testing into continuous integration (CI) system.**



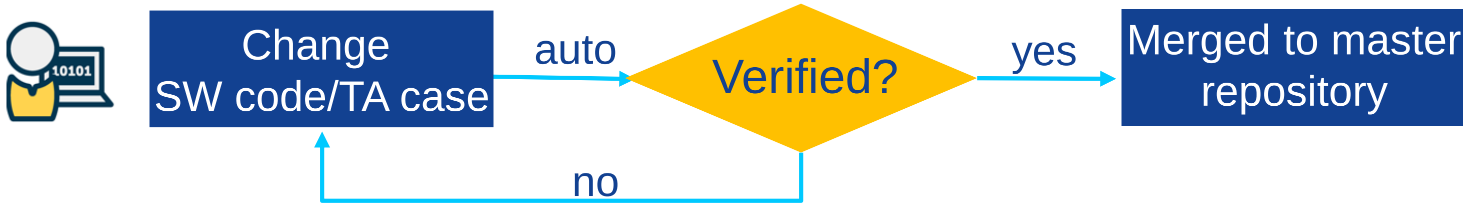
# Add Fully TA into CI



UT, CT, ET, IT, ST, …

Note: UT is born with automation

## 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 times/day = **250** x 1 + 15 x 6

$\Downarrow$

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

$\Downarrow$

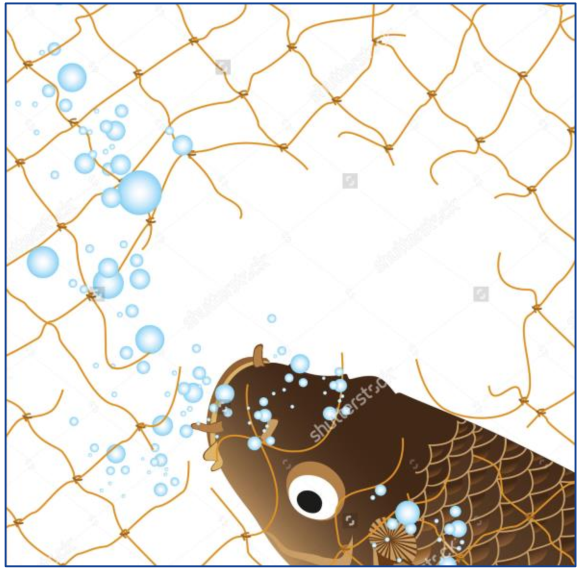
Find SW bugs efficiently, Find SW bugs ***early***

## Achieving A Good TA

**(5) Continuously improve TA by RCA/EDA.**

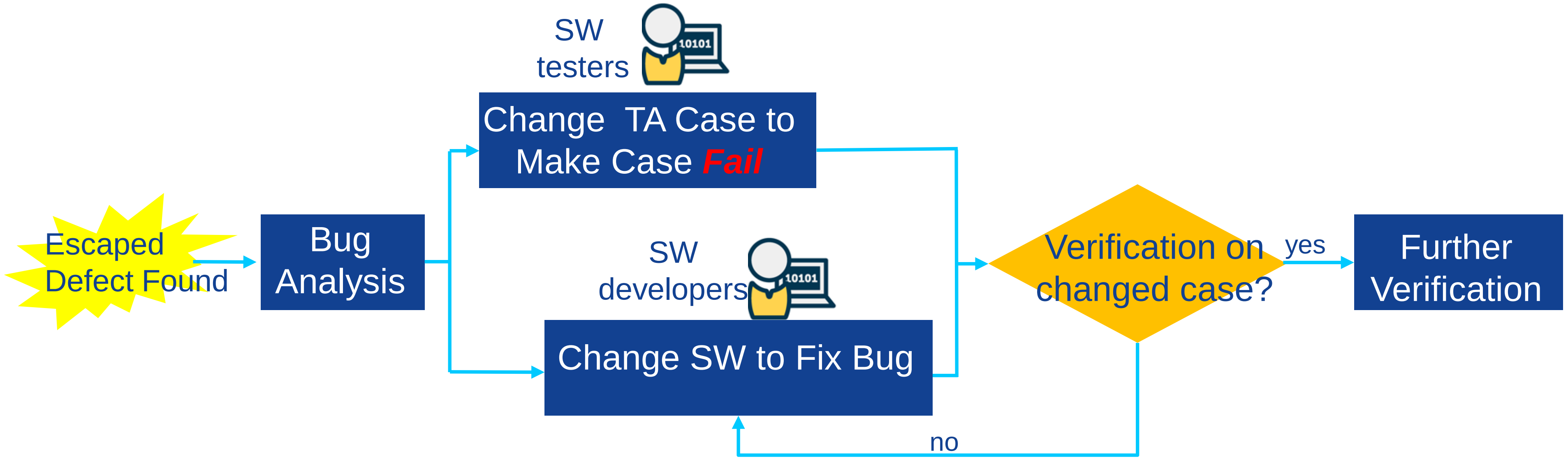
## RCA/EDA of Test Automation

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



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

# Do EDA by *Reproducing the Escaped Bug*



***(1) Quality of TA case is improved***
***(2) Verification of SW Bug Fixing is accelerated***

## Achieving A Good TA

**(6)TA Left-shift: strengthen automation of *early* test stages.**

## Necessity of TA Left-shift

**The Google Testing Law :**

***"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 of TA Left-shift

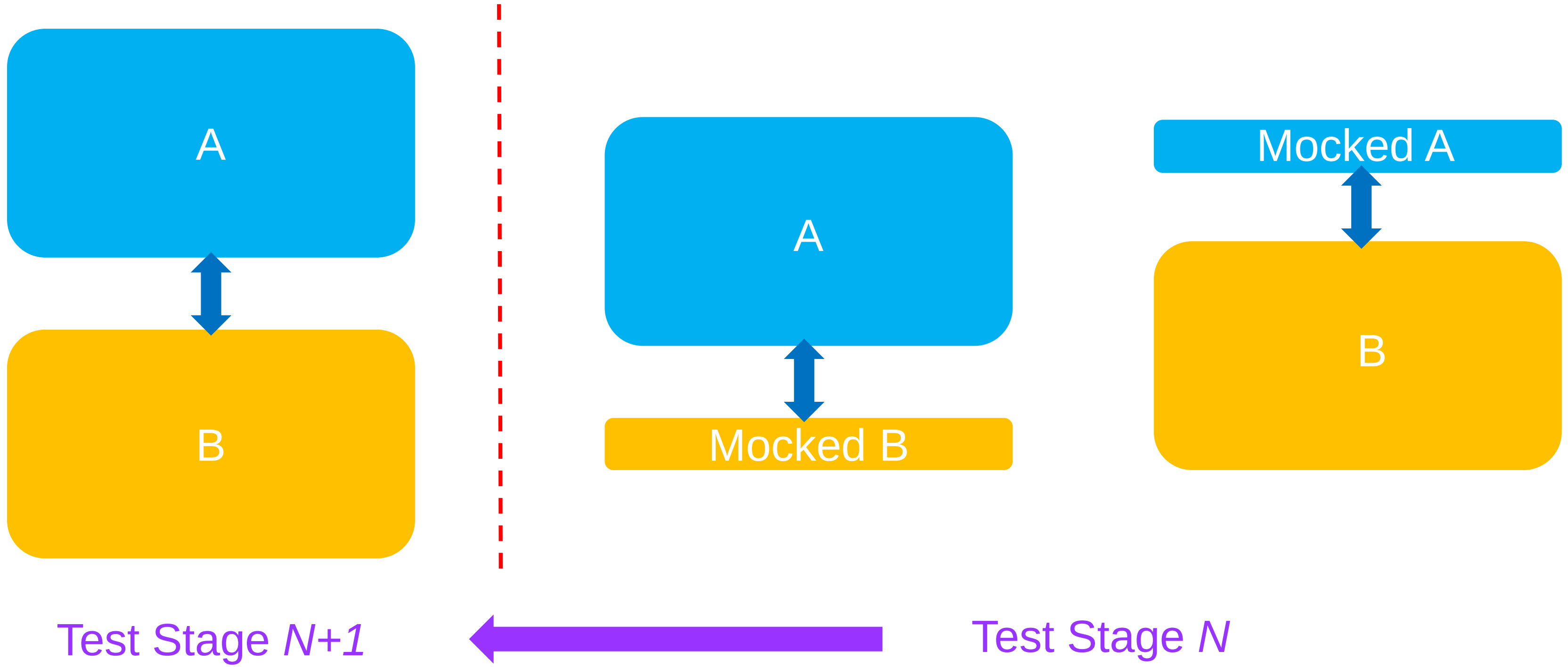
**The Testing Coverage Law：**

***"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

**(7) Use Mock technique as much as possible.**

## What is Mock?



## Benefits of Mock

Focus on the tested object

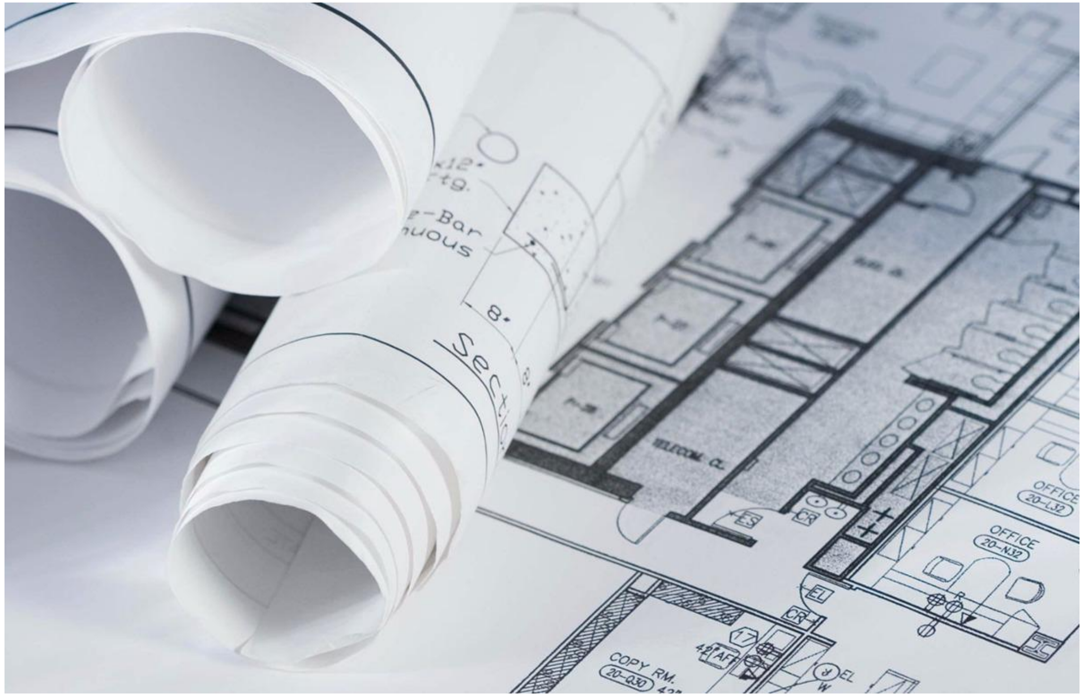
Starts test early

Test ENVs easily copied

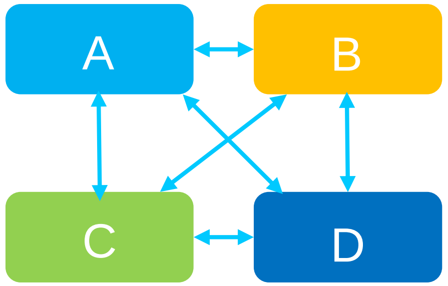
Cheap since only mocking interface

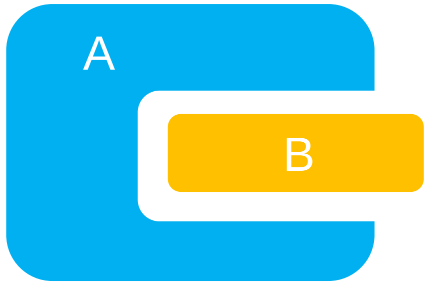
## Achieving A Good TA

**(8) Take *testability* into account when designing software.**

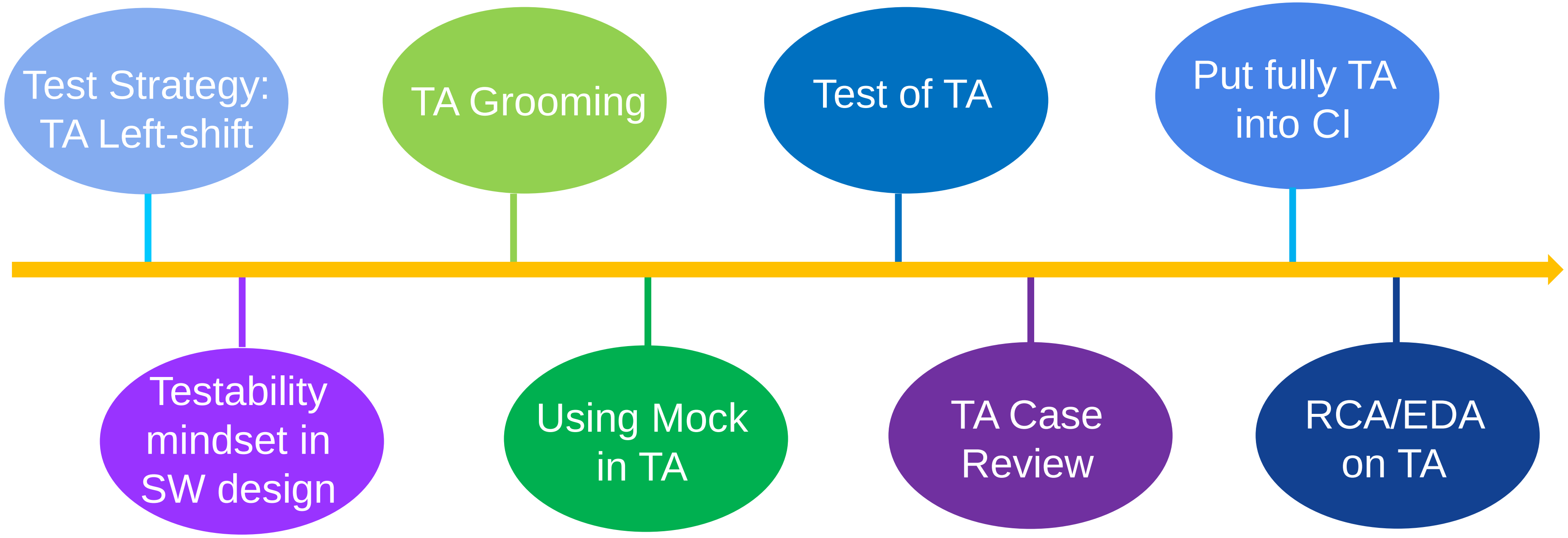


## Some Examples of Bad Testability





# Summary of the messages towards a good TA



## The improved test process is much like developing a software because:

*Test Automation is*
*Software Development*

## Q & A