

# Tester: One Step Forward

Jacky Guo

# About this presentation

- Overview
  - Testing vs. Quality Engineering
  - Start preventing bugs

# Objectives

- After this presentation, you'll be able to:
  - Identify a phased approach for preventing bugs
  - Identify two actions for tomorrow to prevent bugs

# Test vs. Quality Engineering (SQE)

*"Test: to try something out... in order to find out... how well it works..."*

*"Engineering: the application of science in the design planning, construction and maintenance of... manufactured things"*

# Test vs. Quality Engineering (SQE)

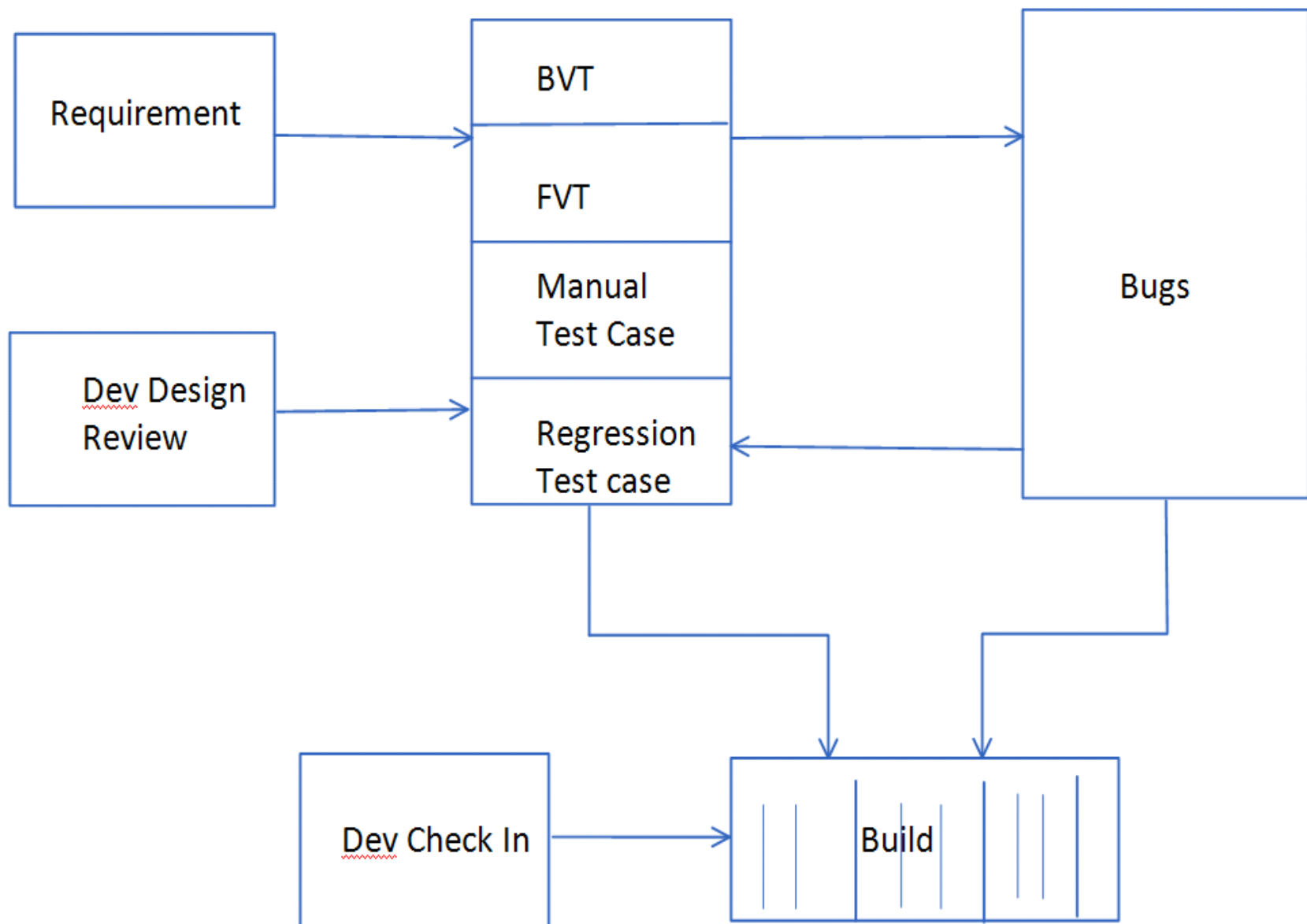
(continued)

## **Test**

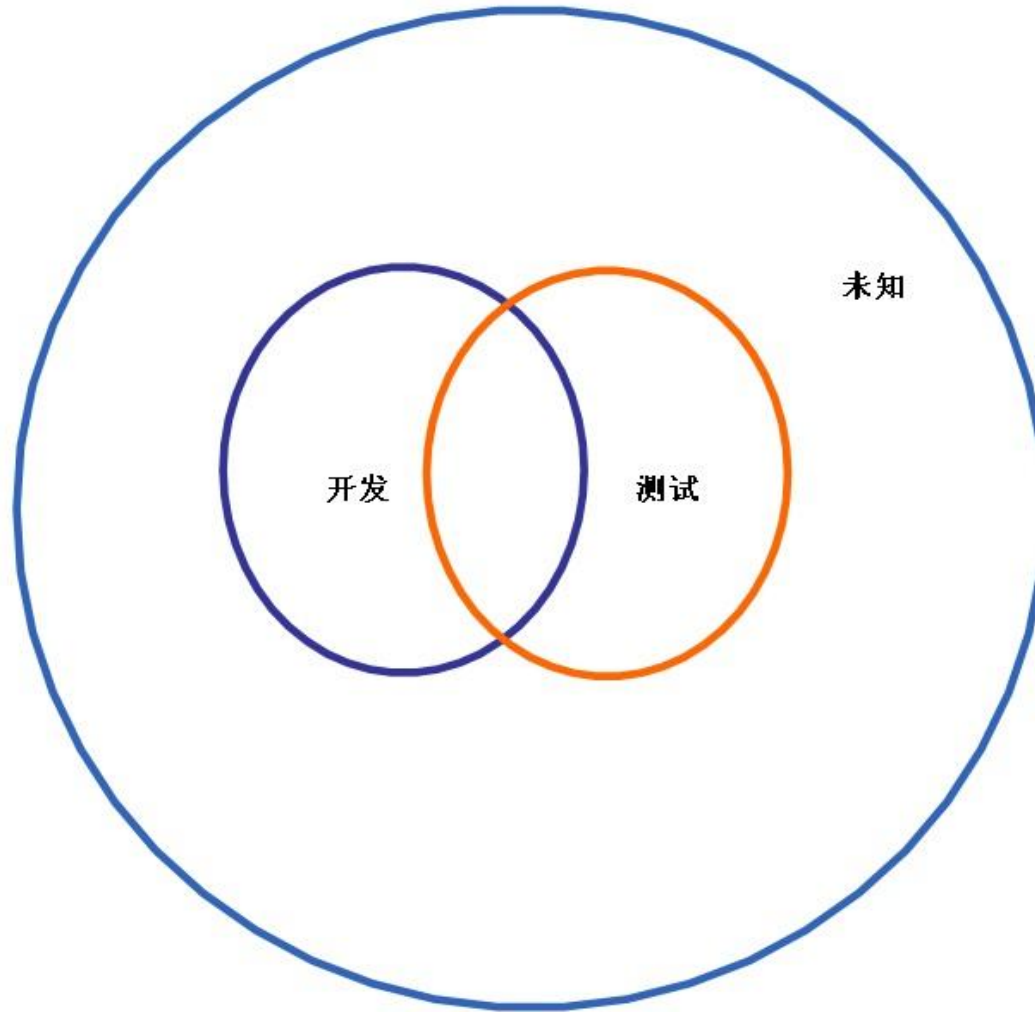
- Late in the timeline
- Reactive
- Find defects
- Singular in affect

## **Quality Engineering**

- First in the timeline
- Proactive and reactive
- Understand defects
- Multiplicative in affect



# Knowledge



# Phase 1: how do I start?

- Written requirements
  - Clarify user & user scenario
  - Have your design principle
- Simple
  - One Step instead of two or three
  - Avoid repeated things
  - Batch command
- Bug the documents
  - "Does everyone see the same picture?"
  - "Will" versus "might, should, desirable"
- Testability/ Sustainability
- SMART





米世界

通知中心

米世界

握手



王金平

米聊号 578922

你的名片太棒了



爱生活，爱米聊  
[Mi][v5]



附近热点



附近的人



握手



我的广播



求缘猫



对话



广播



好友



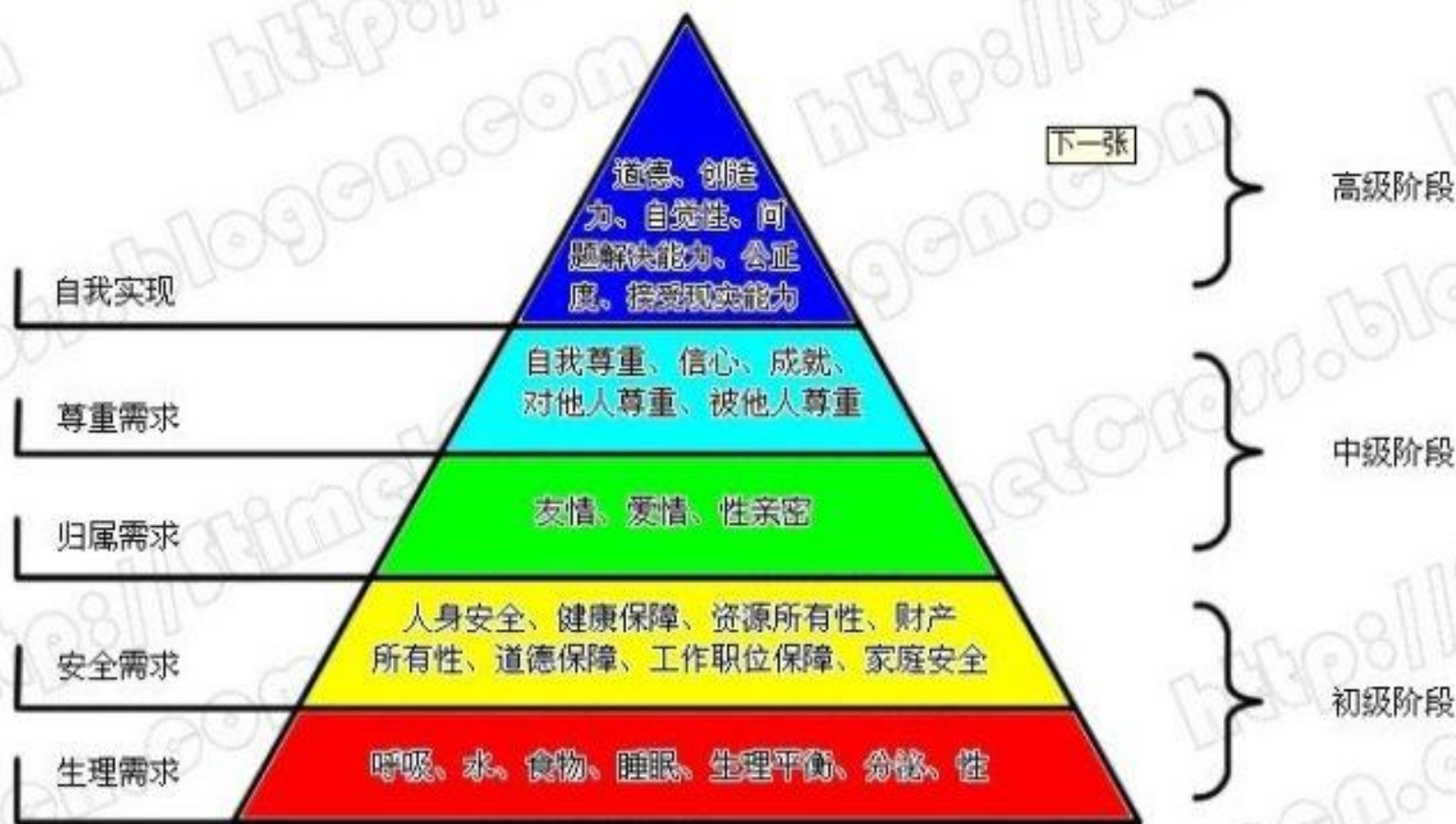
米世界



挥动你的手机和附近的同伴握手

或者

点击此处



[blog.sina.com.cn/jackguoxz](http://blog.sina.com.cn/jackguoxz)

# Case Studies

- Weibo
- 360
- Douban
- Alipay
- Dianping
- Online Game



淘宝会员

支付宝会员

— 为了您的账号安全，请输入验证码。

账户名

@hotmail.com

密 码

\*\*\*

验证码

M-5N

看不清  
换一张

☒ 安全控件登录

登 录

[忘记密码?](#)

 使用手机号码登录 | [免费注册](#)

已经购买过的访客，[点此登录](#)







# Phase 1: how do I start? (continued)

"Look" at Dev Design

- Understand dev design
  - ✓ Work flow
  - ✓ Data flow
  - ✓ Data Structure
- Cover different user scenario
- Testability
- Risk



# Phase 1: how do I start? (continued)

"Look" at code

- API & Parameters
- Code inspections or reviews
  - Reviews are light-weight
  - Target code subsets
  - Inspect only the most critical/difficult sections
- Debugging skill

我的Windows 7多次蓝屏，每次出现蓝屏的操作都不一样，比如这次是打开一个程序，下次是打开另外一个程序；最近实在受不了了，只好下载了windbg来分析一下dump文件（在C:\Windows\Minidump下面）；

- 到这里下载windbg<http://msdn.microsoft.com/zh-cn/windows/hardware/gg463009/> 或者这里<http://msdn.microsoft.com/zh-cn/windows/hardware/gg463016>

- 打开dump文件，然后运行 !analyze -v

ADDITIONAL\_DEBUG\_TEXT:

Use '!findthebuild' command to search for the target build information.

If the build information is available, run '!findthebuild -s ; .reload' to set symbol path and load symbols.

MODULE\_NAME: Hooksys

FAULTING\_MODULE: 8403e000 nt

DEBUG\_FLR\_IMAGE\_TIMESTAMP: 4edda48b

BUGCHECK\_STR: 0x7f\_8

CUSTOMER\_CRASH\_COUNT: 1

DEFAULT\_BUCKET\_ID: VISTA\_DRIVER\_FAULT

CURRENT\_IRQL: 0

LAST\_CONTROL\_TRANSFER: from 840ce18e to 840cd1f0

- 然后运行!mvm Hooksys

start end module name

9b428000 9b450a80 Hooksys T (no symbols)

Loaded symbol imagefile: Hooksys.sys

Imagepath: \??\C:\Windows\system32\drivers\Hooksys.sys

ImageName: Hooksys.sys

Timestamp: Tue Dec 06 13:13:47 2011 (4EDDA48B)

Checksum: 0002ED85

ImageSize: 00028A80

Translations: 0000.04b0 0000.04e4 0409.04b0 0409.04e4

然后google了一下，发现hooksys.sys是瑞星实时监控所要使用的文件，与运行的文件密切相关，解决办法就是把瑞星杀毒软件卸载或者删除掉。

# Phase 1: done

## Congratulations

- You have just grabbed low-hanging fruit
- You have prevented countless bugs
- You have risen above at least 50% of the Industry

Now what?

# Phase 2: change the process

Learn about your bugs

- Find the cause
  - Unexpected customer requirement?
  - Vague spec?
  - Rushed design?
  - Domain knowledge?
  - Code error?
    - Be gentle, accurate and honest
- Make a change
- Bug “Social Network” & Data mining

# Call to action

- Start small, get a win
  - What are people most upset about?
  - What is the low-hanging fruit?
  - What does the RCA data say?
- Be prepared to set the example

# Call to action (continued)

What can you do tomorrow (pick 1)?

- Document one requirement
- Document one design
- Code-review one critical section
- RCA 10 bugs
- Implement 10 unit tests

Q&A

# summary

## Test case Management

- TC from Bugs by customer/external users
- TC from review dev design
- TC from user scenario
- TC from Bugs review in case of regression
- Regression Test strategy("diff" VS "full")
- BVT runs before Dev Check in



# summary

## Bug Management

- Map to TC
- Bug quality
- Root cause(process may needs change)
- Bug fix review to evaluate the risk
- Control the bar to fix bug(potential risk)

## Goals

1. Does the proposed design meet the *spec requirements* and features?
2. How can the proposed design be done in the *available time* in the milestone?

## Steps

### 1. Validate Design.

Are the algorithms and data structures appropriate for the problem?

### 2. Validate Tasks.

Are the tasks complete for each feature?

### 3. Validate Schedule.

### 4. Validate Dependencies. Are the dependencies complete and understood?

### 5. Validate Check-ins. Are the scheduled check-ins testable and frequent enough for test to be efficient?

### 6. Mark Open Issues. At each validation step, tasks that have open issues that keep the task from being "codable" will have the Open Issues checkbox marked.

# Code Review Checklist

## General

---

- ☐ Code has been run through static analysis tools to remove top tier defects prior to review.
- ☐ All resources are efficiently created and properly disposed of.
- ☐ Look for logic errors
- ☐ Look for design patterns \ refactoring for optimization
- ☐ Look for extensibility, scalability...
- ☒ Code should be buddy built, and provide a sanity machine for test if possible

## Security

---

- ☐ Code is free of buffer-overruns.
- ☒ Code satisfies the requirements of the [Secure Windows Initiative](#) (SWI) team.
- ☐ All input parameters must be validated and checked for null, as well as in Constructor

## Internationalization

---

- ☒ All user-accessible strings are stored in a [localizable](#) format, such as resource files.
- ☒ All controls that display strings on the UI are sized to accommodate the largest translations.
- ☐ All code satisfies team internationalization guidelines, including multi-language input and multi-cultural data formats.
- ☐ All strings would be read by end users should be localized

## Coding Standards

---

- ☒ The code adheres to team coding standards, and exceptions are documented and justified.
- ☒ Catch and handle exceptions correctly
- ☒ Complete and clear comments in code, especially the algorithm you are using
- ☒ Avoid hard-coded elements in your code, and identify the constants correctly

## Reliability

---

- ☒ All arguments on publicly accessible APIs are validated (includes protected for extensible classes).