



SALADPUK

Test Driven Development (TDD)

# Learning concepts

We LEARN more when we LEARN TOGETHER

# Learning concepts

We LEARN more when we LEARN TOGETHER

The heart of learning is to focus on the concepts, which is more important than focusing on the details



# Learning concepts

We LEARN more when we LEARN TOGETHER

The heart of learning is to focus on the concepts, which is more important than focusing on the details 😊

I can EXPLAIN it to you, but I can't UNDERSTAND it for you 🤔

# Agenda

1. The World of Software Testing
2. Code Refactoring
3. Isolated Test Environments
4. High Level Testing
5. Workshop

# The World of Software Testing



# ทำไมต้องเขียนเทส ?

**เสียเวลา** - เอาเวลาไปเขียนโคดเดยไม่ดีกว่าเหรอ? **จะเปลี่ยนทำใหม่** - ทุกวันนี้ก็ไดแล้วอยู่แล้วนิ?

**งานเร่ง** - วันหยุดก็แทบไม่ได้พักแล้วยังจะเพิ่มงานอีกเหรอ? บลาๆ

— เสียงเดฟแบบกล่าวในใจ —

# โลกของซอฟต์แวร์

# ปัญหาบริษัทซอฟต์แวร์

การอยู่รอดของบริษัทขึ้นกับการจัดการของเพียง 4 อาย่าง



1. รับงานใหม่
2. จัดการงานเก่า
3. จัดการทีม
4. บริหารสภาพคล่อง

# ปัญหาบริษัทซอฟต์แวร์

บริษัท — ไม่สามารถรับงานใหม่ได้ แม้ว่าอยากรับแค่ไหนก็ตาม 😞



1. รังสรรค์งานใหม่
2. จัดการงานเก่า
3. จัดการทีม
4. บริหารสภาพคล่อง

# ปัญหาบริษัทซอฟต์แวร์

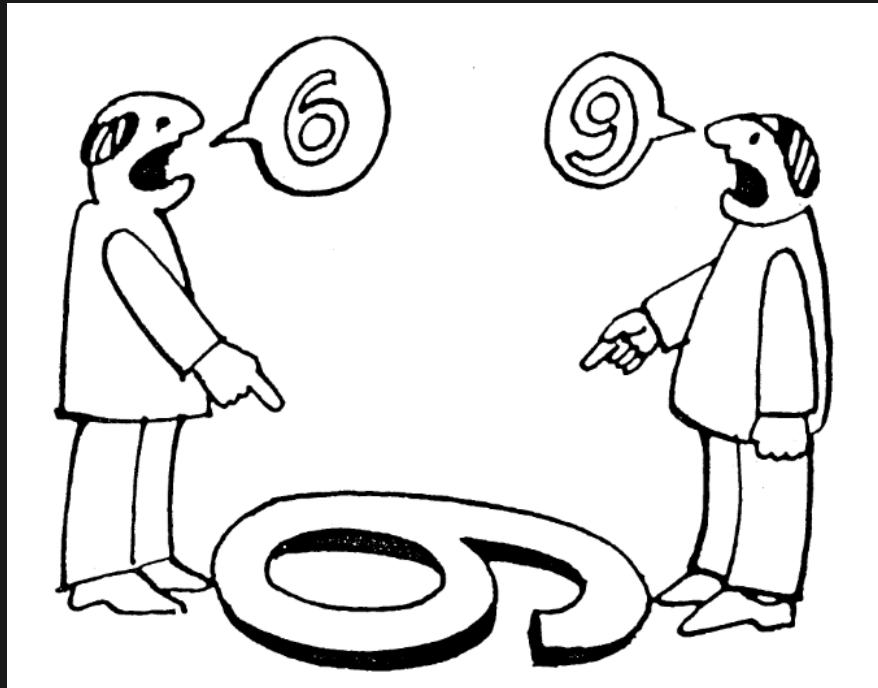
เดฟ — ไม่กล้าแก้โค้ดเดิม เพราะกลัวว่าของที่ใช้งานได้จะพัง 😞



1. รับงานใหม่
2. จัดการงานเก่า
3. จัดการทีม
4. บริหารสภาพล่อง

# ปัญหานบริษัทซอฟต์แวร์

เดพ & Biz — เข้าใจประเด็นแตกต่างกัน ทำให้งานไปกันคนละทิศ 😠

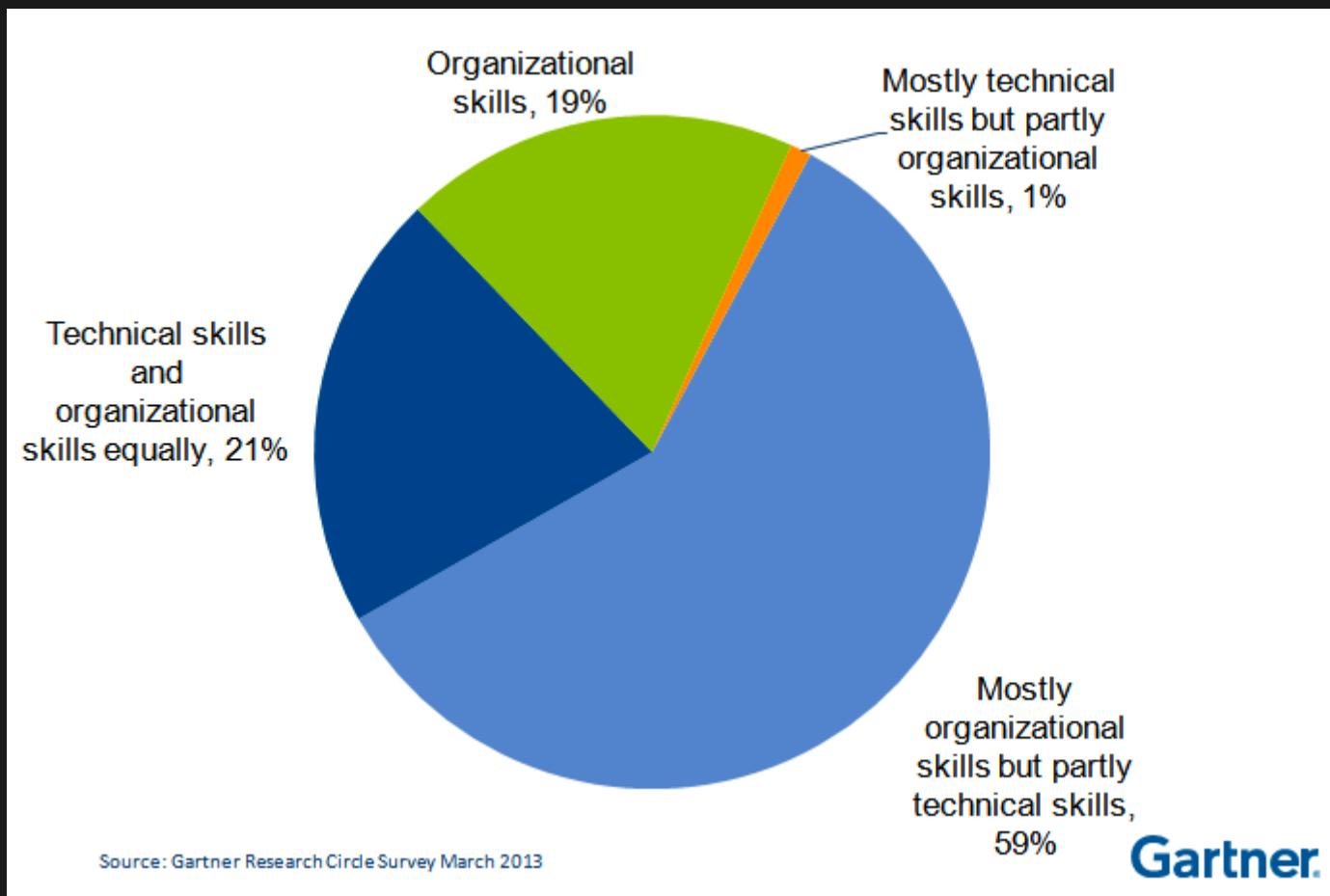


1. รับงานใหม่
2. จัดการงานเก่า
3. จัดการทีม
4. บริหารสภาพคล่อง

ปัญหาของบริษัทซอฟต์แวร์  
ไม่ใช่การทำซอฟต์แวร์ 😐

# ปัญหารि�ชัฟต์ซอฟต์แวร์

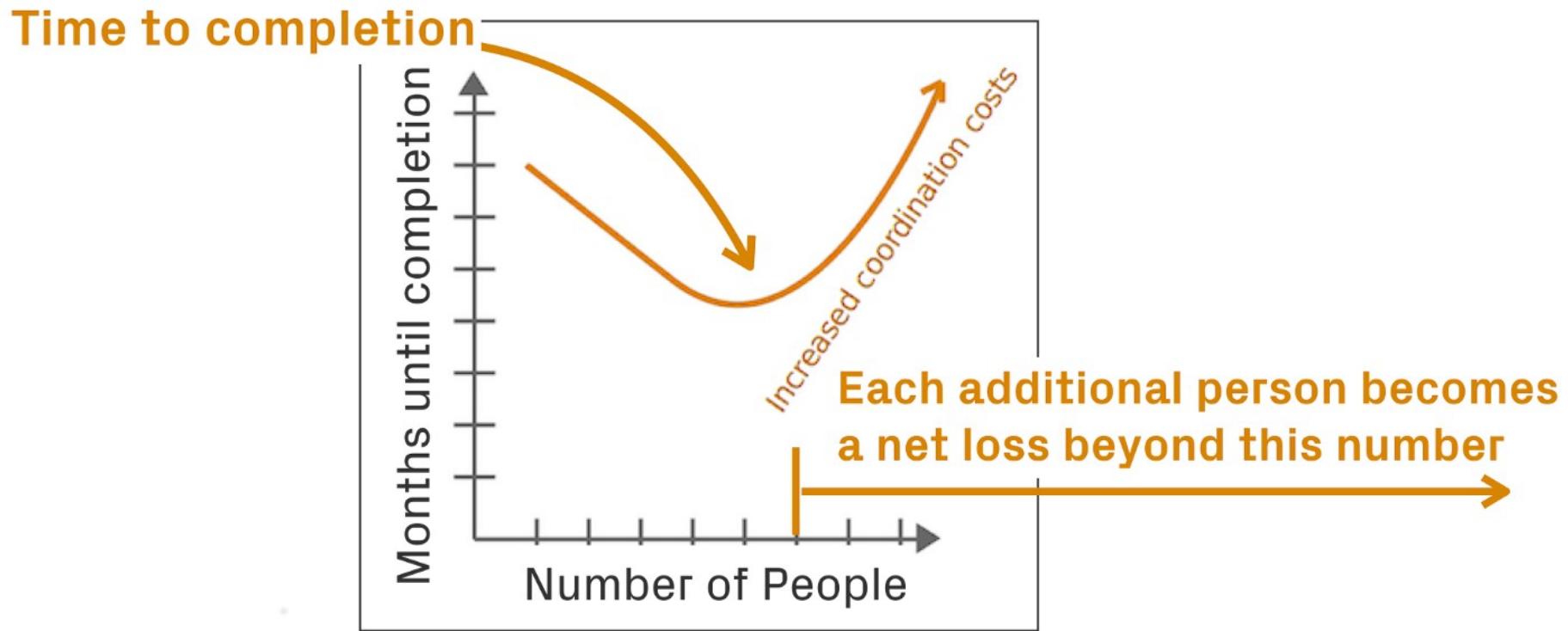
สาเหตุหลักที่ทำให้บริษัทซอฟต์แวร์ตายนี้ได้เกิดจาก Technical skills แต่กลับเป็น Non-technical skills ต่างหาก (การบริหารจัดการโปรเจค ทัศนคติ การทำงานเป็นทีม)



# ปัญหาระบบทุกซอฟต์แวร์

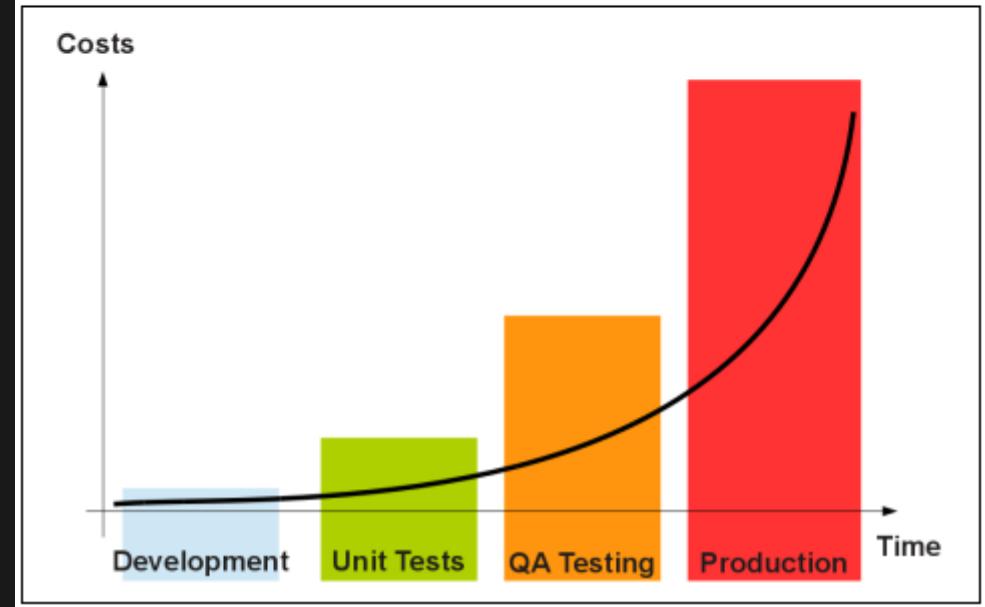
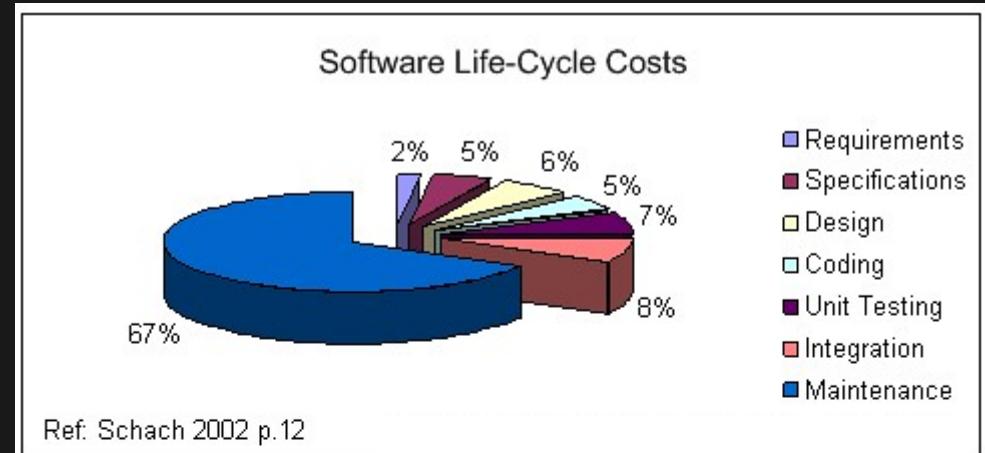
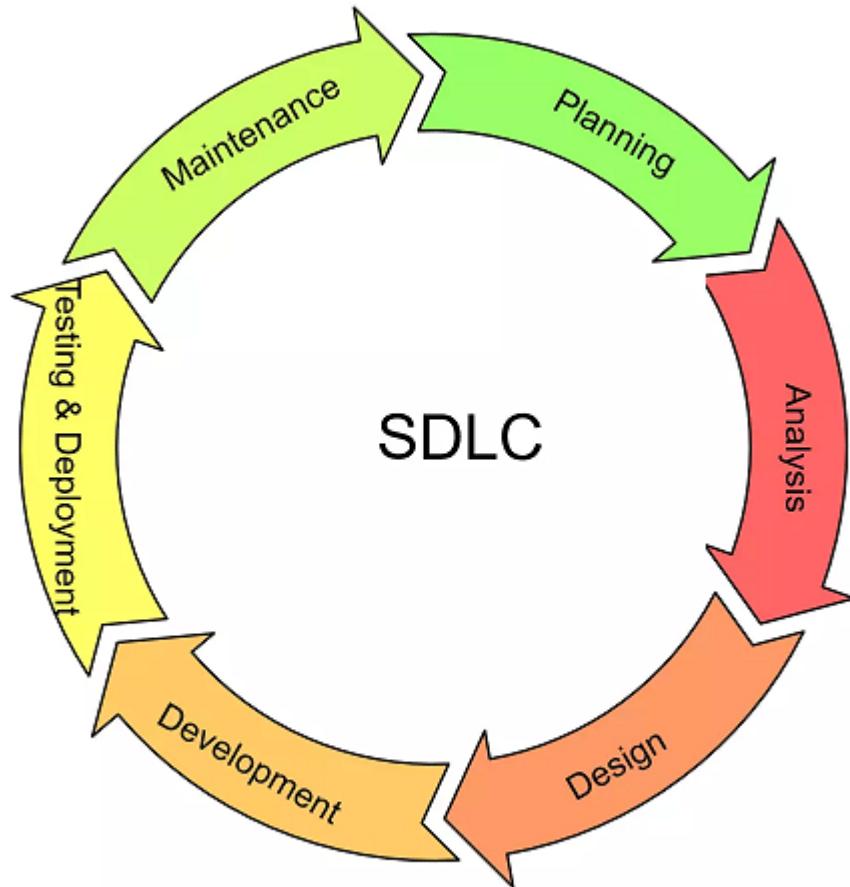
Brooks's Law — เมื่อถึงจุดหนึ่งยิ่งเพิ่มคนจะยิ่งทำให้งานช้าลงเรื่อยๆ

Persons vs Time to Completion



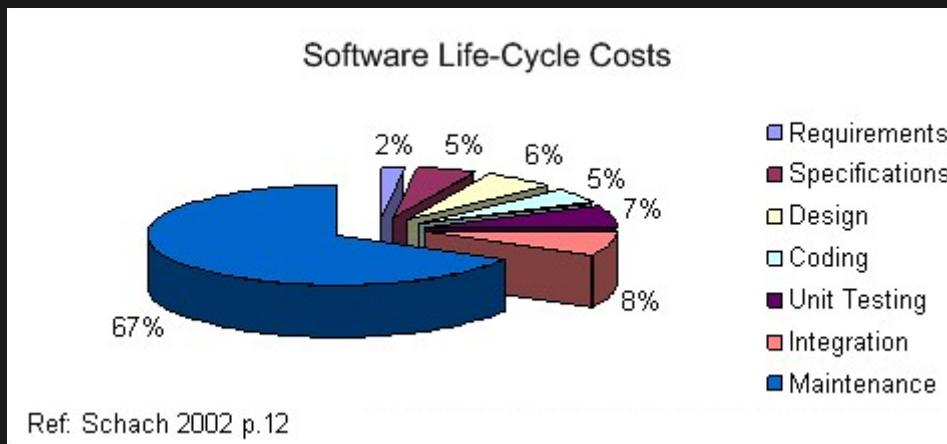
# ปัญหาบริษัทซอฟต์แวร์

Maintenance — คือค่าใช้จ่ายหลักในการทำซอฟต์แวร์โปรดเจด



# Maintenance Challenges

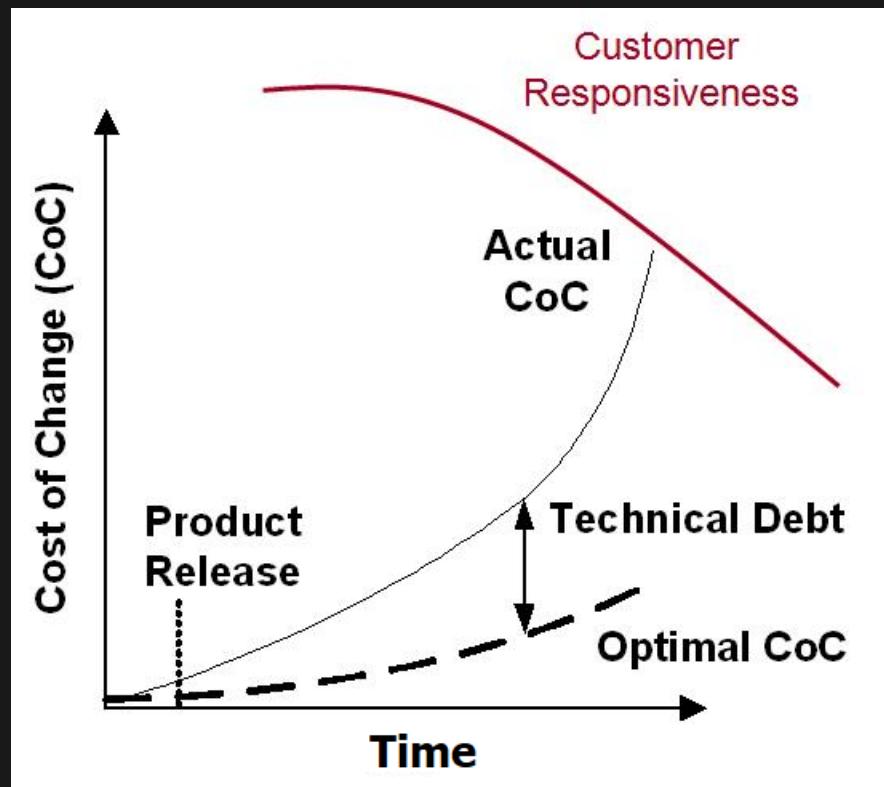
หลากหลายเจ้าของ - พีสัมที่เขียนไว้เข้าลาออกไปแล้ว, การรักษาคุณภาพของโคด - เดียวค่อยกลับมาแกะลอกกัน, Legacy code - คงมีแต่พระเจ้าที่เข้าใจโคดนี้แล้วซินะ, Domain expertise - การคำนวณทางบัญชีใช้สูตรอะไร, Verification & Validation - ทำตามลูกค้าพูดแล้วนะแต่ทำไมยังไม่ถูก, Requirement change - อ้าวไม่ได้ออกแบบให้ทำเรื่องพวกนี้ได้นะ



↑ Maintenance cost → การรับงานใหม่ + จัดการงานเก่า + ความรู้สึกของคนในทีม

# ปัญหาบริษัทซอฟต์แวร์

**Technical Debt** — เดฟไม่ชอบโปรเจคที่ทำอยู่ เพราะมีปัญหาเยอะ รอเวลาขึ้นโปรเจคใหม่ เพราะทุกอย่างใหม่และเร็ว แต่ผ่านไปซักพักก็เป็นปัญหาเหมือนโปรเจคเดิม วนเป็นวัฏจักร



The performance paradox

ยิ่งปล่อยเวลาผ่านไป

- ยิ่งทำยิ่งແຍ່လັງ
- ค่าใช้จ่ายยิ่งสูงขึ้น
- ตัวเลือกยิ่งน้อยลง
- ความน่าเชื่อถือยิ่งลดลง
- ประเมินเวลาทำงานไม่ได้

4. ຂອບຄວາມສົກລົງ



คนร้ายที่ก่อเหตุคือใคร ?



ความจริงมีเพียงหนึ่งเดียว

คนร้ายคือพวกรานี่แหละ

imgflip.com

Developers



# ເຕີມໄນ່ຮູ້ເຫຼວວ່າມີປົງຫາພວກນີ້ ?

ຮູ້ນານແລ້ວ - ແຕ່ໄນ່ຮູ້ຈະແກ້ໄງດີ, ແກ້ໄນ່ໄດ້ຫຮອກ - ແກ້ໄປແລ້ວຮະບນມີປົງຫາໃດຈະຮັບຜິດຂອບ?, ເຄຍລອງແລ້ວ - ສຸດທ້າຍກລັນມາເປັນເໜືອນເກ່າ, ໂຄດຟີສົມ - ອ່ານແລ້ວໄມ່ເຂົາໃຈ ແຕ່ຮູ້ວ່າແກ້ແລ້ວພັ້ງຕລອດເລຍປ່ອຍໄວ້ງໍ່ ບລາງ — ເສື່ອງເຕີມແອນກລ່າວໃນໃຈ —



# ເຕີມໄມ່ຮູ້ແຮວວ່າມີປົງຫາພວກນີ້ ?

ຮູ້ນານແລ້ວ - ແຕ່ໄນ່ຮູ້ຈະແກ້ໄງດີ, ແກ້ໄມ່ໄດ້ຮອກ - ແກ້ໄປແລ້ວຮະບນມີປົງຫາໃດຈະຮັບຜິດຂອບ?, ເຄຍລອງແລ້ວ - ສຸດທ້າຍກລັນມາເປັນເໜືອນເກ່າ, ໂຄດຟືສົມ - ອ້ານແລ້ວໄມ່ເຂົ້າໃຈ ແຕ່ຮູ້ວ່າແກ້ແລ້ວພັ້ງຕລອດເລຍປ່ອຍໄວ່ຈີ່ ບລາງ — ເສື່ອງເຕີມແອນກລ່າວໃນໃຈ —

ງານຊອົບຕົວແວຣ໌ເປັນງານແຊັນເມດເລຍວັດຄວາມຄຸກຕ້ອງໄດ້ຢາກ



# งานซอฟต์แวร์วัดความถูกต้องยาก ?

TOR - เขียนงานตามนั้นเดี๋ยวแล้วมันใช้งานได้จริงปะ?, คู่มือการใช้งาน - โคดที่รันกับที่เขียนในคู่มือมันยังเหมือนกันอยู่ป่าว?, ทีมเทส - เทสได้เฉพาะเท่าที่เห็นเท่านั้น แล้วของที่ไม่เห็นล่ะ?, ความถูกต้องในการเทส - เทสเข้าจุดเดียวกันยังไม่พออีกเหรอ? หึ! จะขอเทสตอนตี 3 ในวันเสาร์อาทิตย์ด้วย — สีียงเดฟแอนกล่าวในใจ —



# งานซอฟต์แวร์วัดความถูกต้องยาก ?

TOR - เขียนงานตามนั้นเดี๋ยวแล้วมันใช้งานได้จริงปะ?, คู่มือการใช้งาน - โค้ดที่รันกับที่เขียนในคู่มือมันยังเหมือนกันอยู่ป่าว?, ทีมเทส - เทสได้เฉพาะเท่าที่เห็นเท่านั้น แล้วของที่ไม่เห็นล่ะ?, ความถูกต้องในการเทส - เทสเข้าจุดเดียวกันยังไม่พออีกเหรอ? หึ! จะขอเทสตอนตี 3 ในวันเสาร์อาทิตย์ด้วย — สีียงเดฟแอนกล่าวในใจ —

ก็พอได้อยู่นะแต่ไม่สามารถนำมาใช้ในทางปฏิบัติได้จริง

# ❤️ ปัญหาริช็อฟต์แวร์

รับงานใหม่ไม่ได้ — จัดการงานเก่าไม่ได้ — จัดการทีมไม่ได้ — จัดการสภาพคล่องยาก

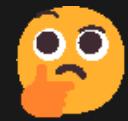
#ไม่กล้าแก้โค้ดเดิม #เข้าใจไม่ตรงกัน #การบริหารจัดการโปรเจค #ทัศนคติ #การทำงานเป็นทีม #การเพิ่มคนเข้าทีม #MaintenanceCost #TechnicalDebt #ยิงทำยิ่งแย่ลง #ค่าใช้จ่ายสูงขึ้นเรื่อยๆ #ค่าน่าเชื่อถือตก #TechnicalSkills #NonTechnicalSkills #รู้ว่ามีปัญหาแต่แก้ไม่ได้ #วัดความถูกต้องได้ยาก #เดพ

คือคนร้ายเสียเงิน 



แล้วจะแก้ยังไง ?

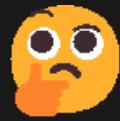
# Why Why Analysis



# สิ่งประกันความปลอดภัยที่ใช้ได้จริง?



โค้ดที่รับประกันตัวมันเองได้ ?



# ทำแล้วได้ประโยชน์อะไร ?

รับงานใหม่ — จัดการงานเก่า — จัดการทีม — จัดการสภาพคล่อง

เช็คตรวจสอบความถูกต้องได้ก่อน ลดค่าใช้จ่ายในระยะยาว มีเวลาไปพัฒนาส่วนอื่น มีตัว

ช่วยตรวจสอบปัญหา แก้โค้ดได้สบายใจ เพิ่มความมั่นใจ เพิ่มคุณภาพโค้ด

เปลี่ยนแปลงโครงสร้างได้ มีตัวชี้วัดที่ชัดเจน เรียกเช็คได้ตลอดเวลา ไม่มีวันหยุด

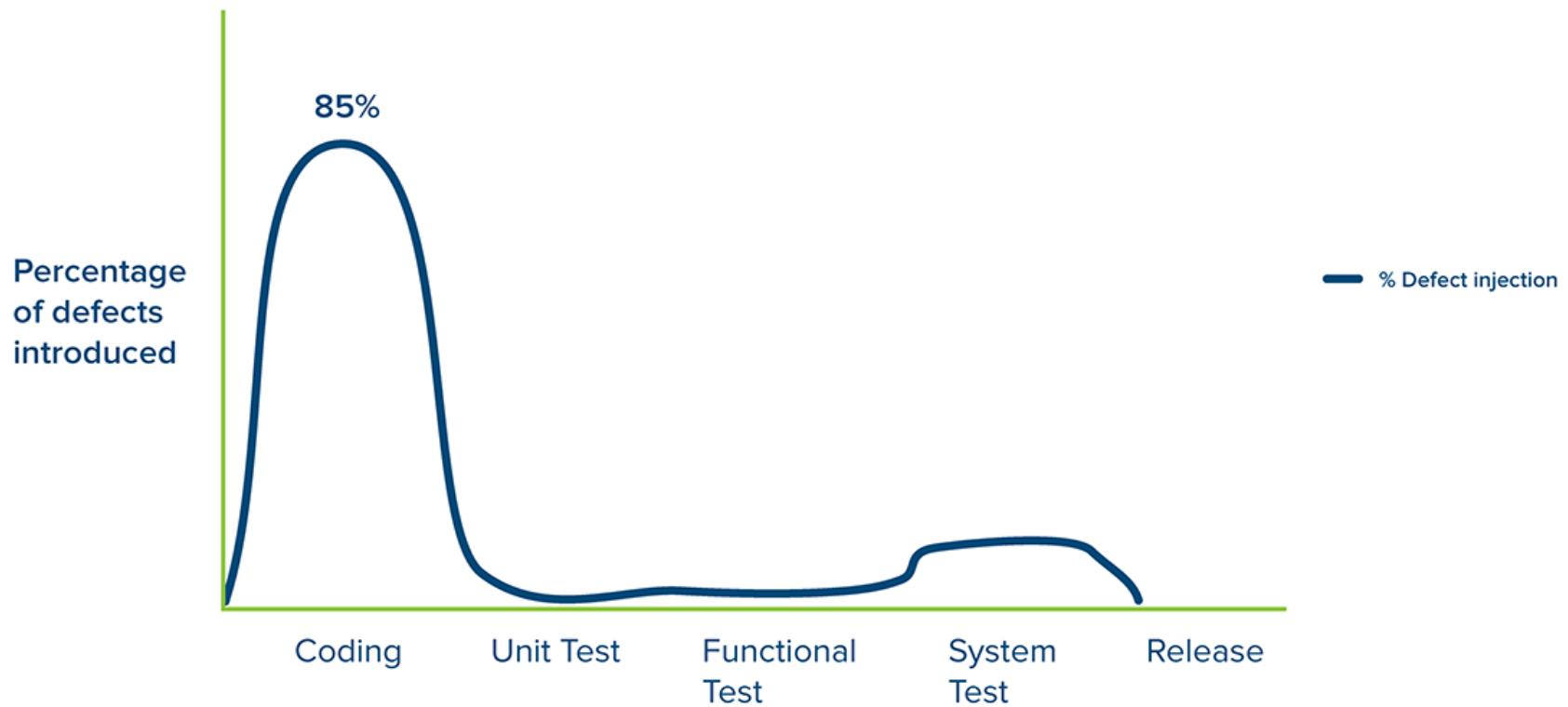


ทำไมต้องเขียนเหล斯 ?



# ทำไมต้องเขียนเทส ?

When do bugs enter the code?

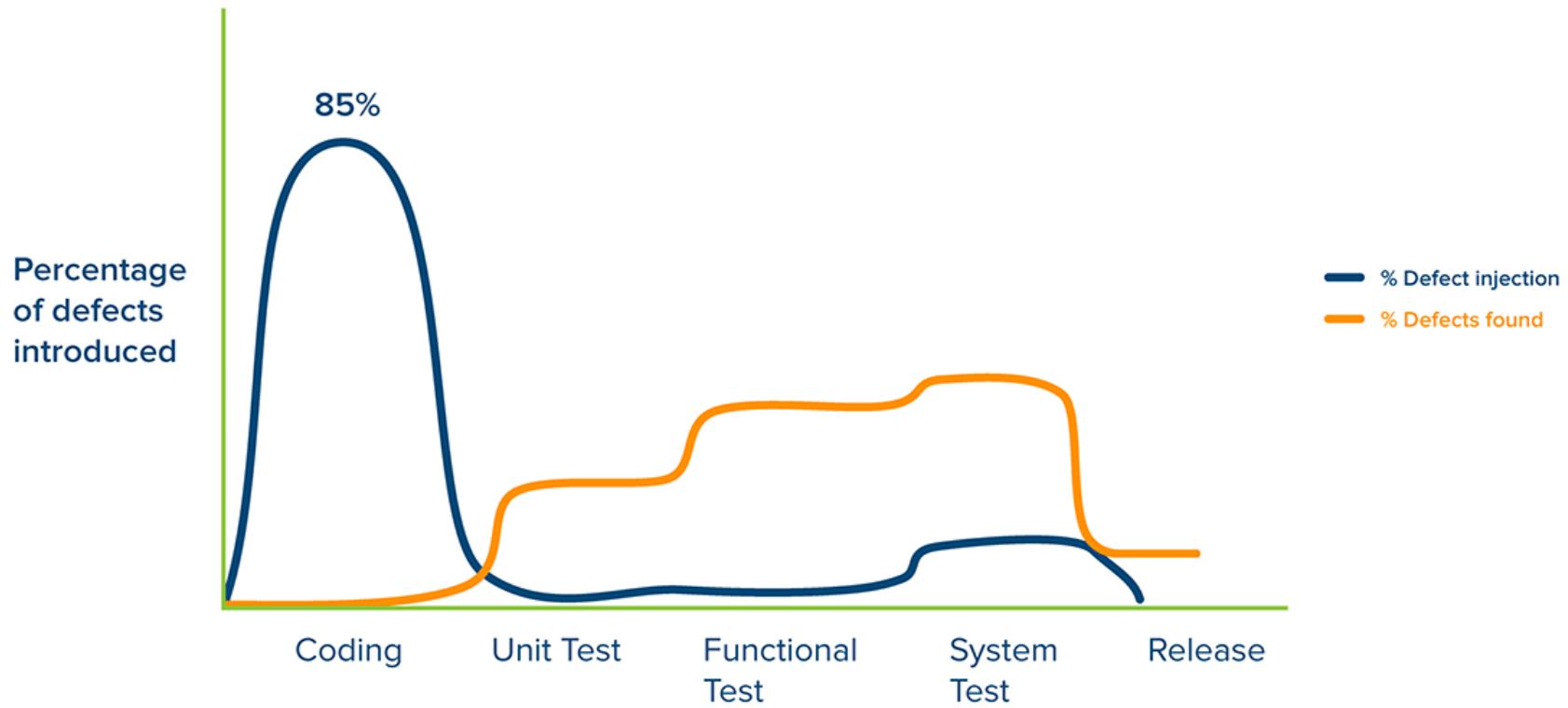


Jones, Capers. *Applied Software Measurement: Global Analysis of Productivity and Quality*.



# ทำไมต้องเขียนเทส ?

When are those bugs found?

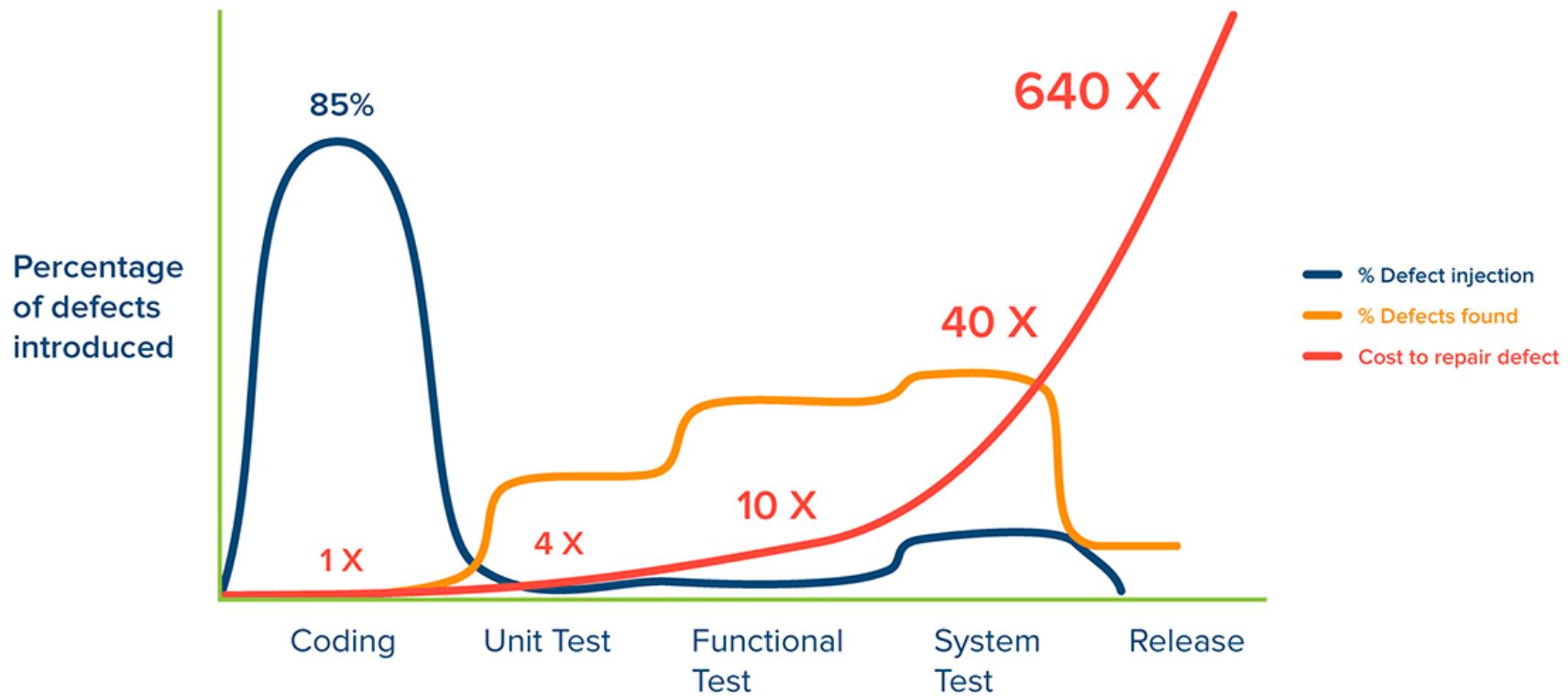


Jones, Capers. *Applied Software Measurement: Global Analysis of Productivity and Quality*.



# ทำไมต้องเขียนเทส ?

What does it cost to fix bugs?

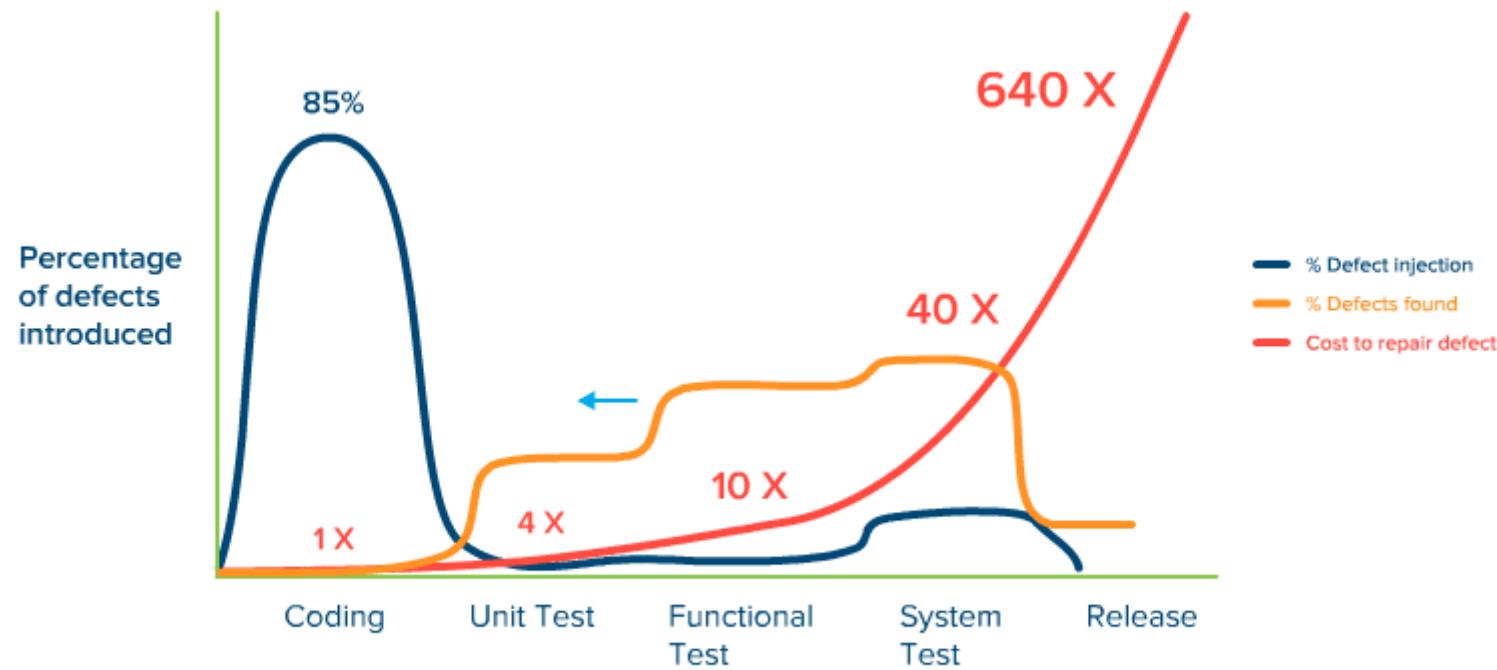


Jones, Capers. *Applied Software Measurement: Global Analysis of Productivity and Quality*.



# ทำไมต้องเขียน-test ?

Test early, test often (the shift left approach)

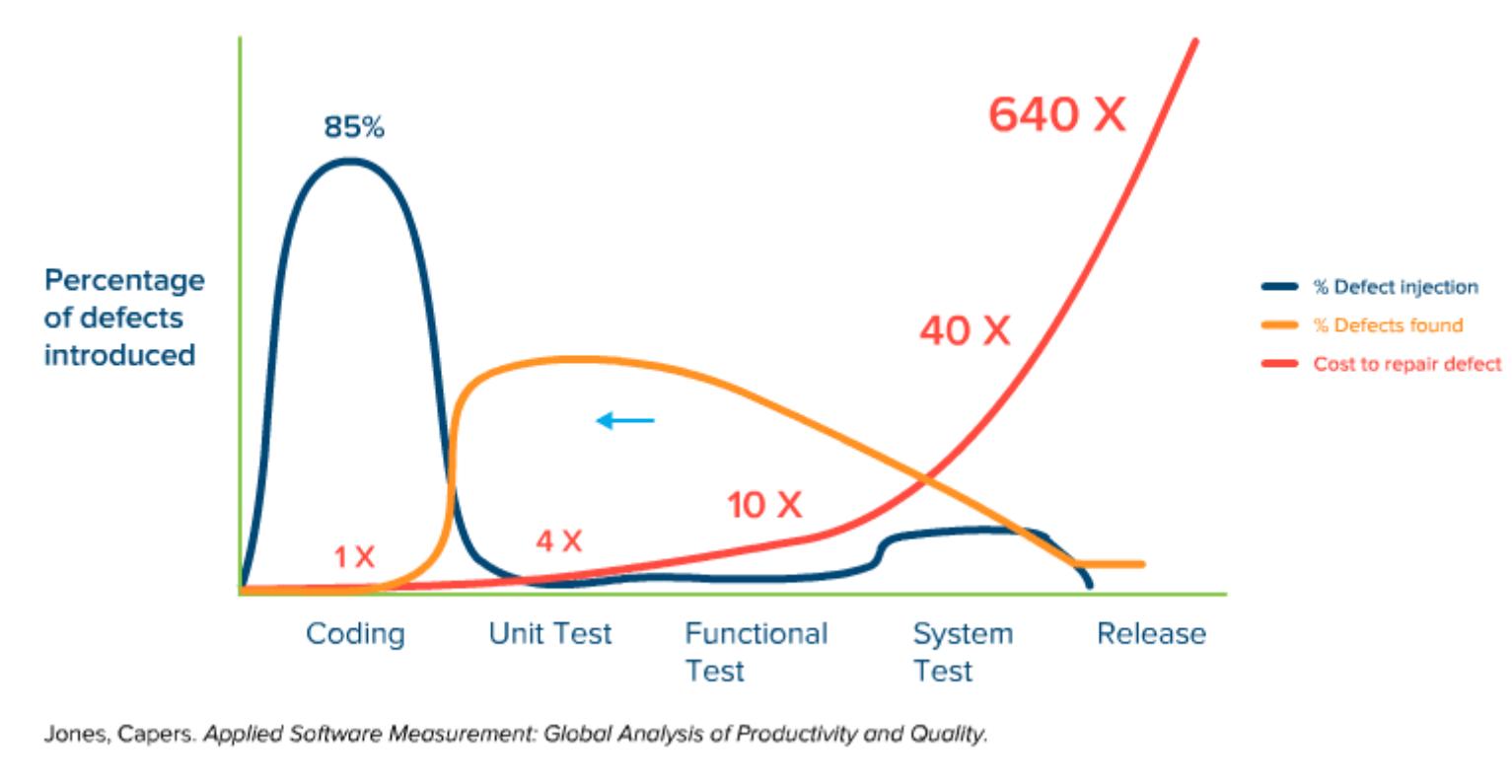


Jones, Capers. *Applied Software Measurement: Global Analysis of Productivity and Quality*.



# ทำไมต้องเขียน-test ?

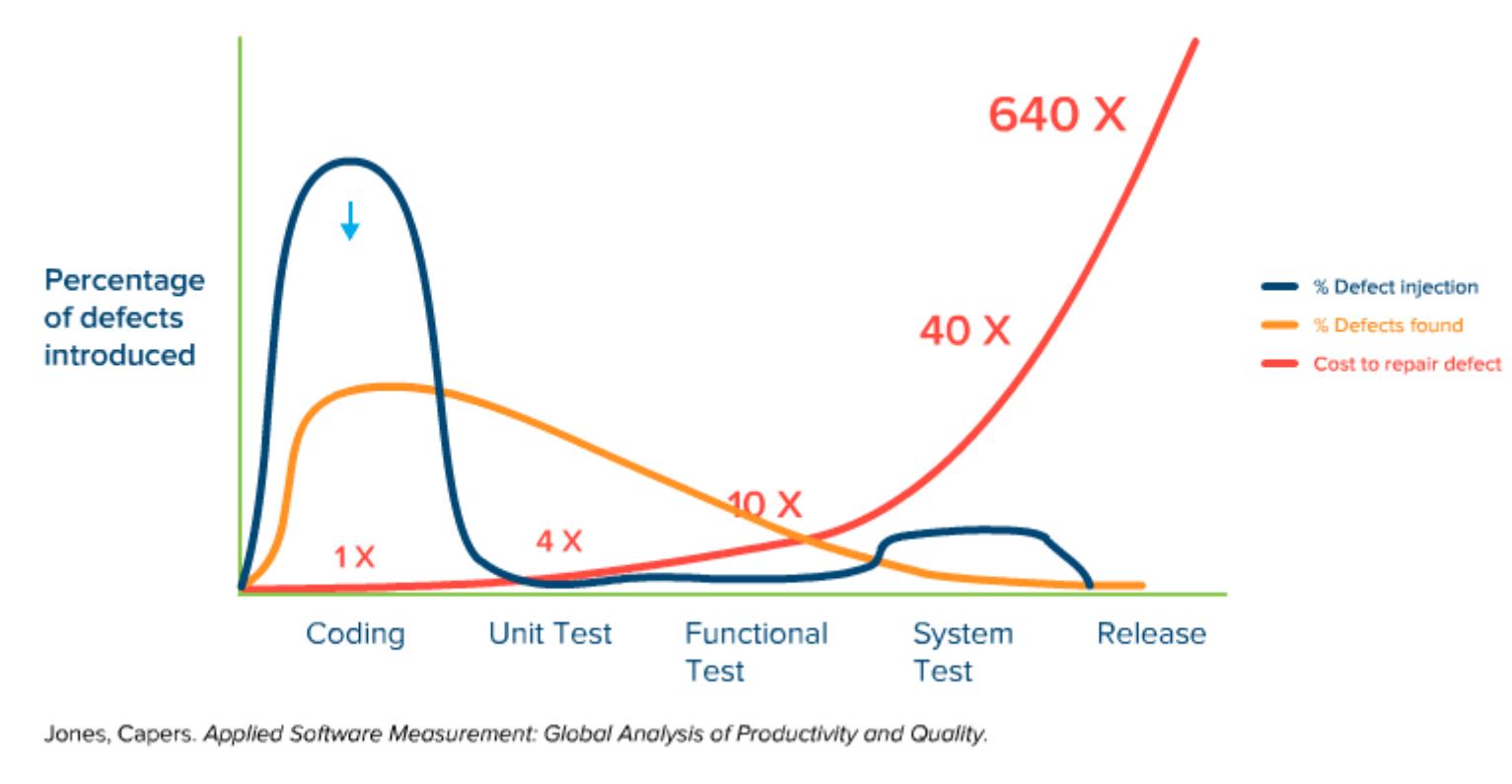
Test early, test often (the shift left approach)





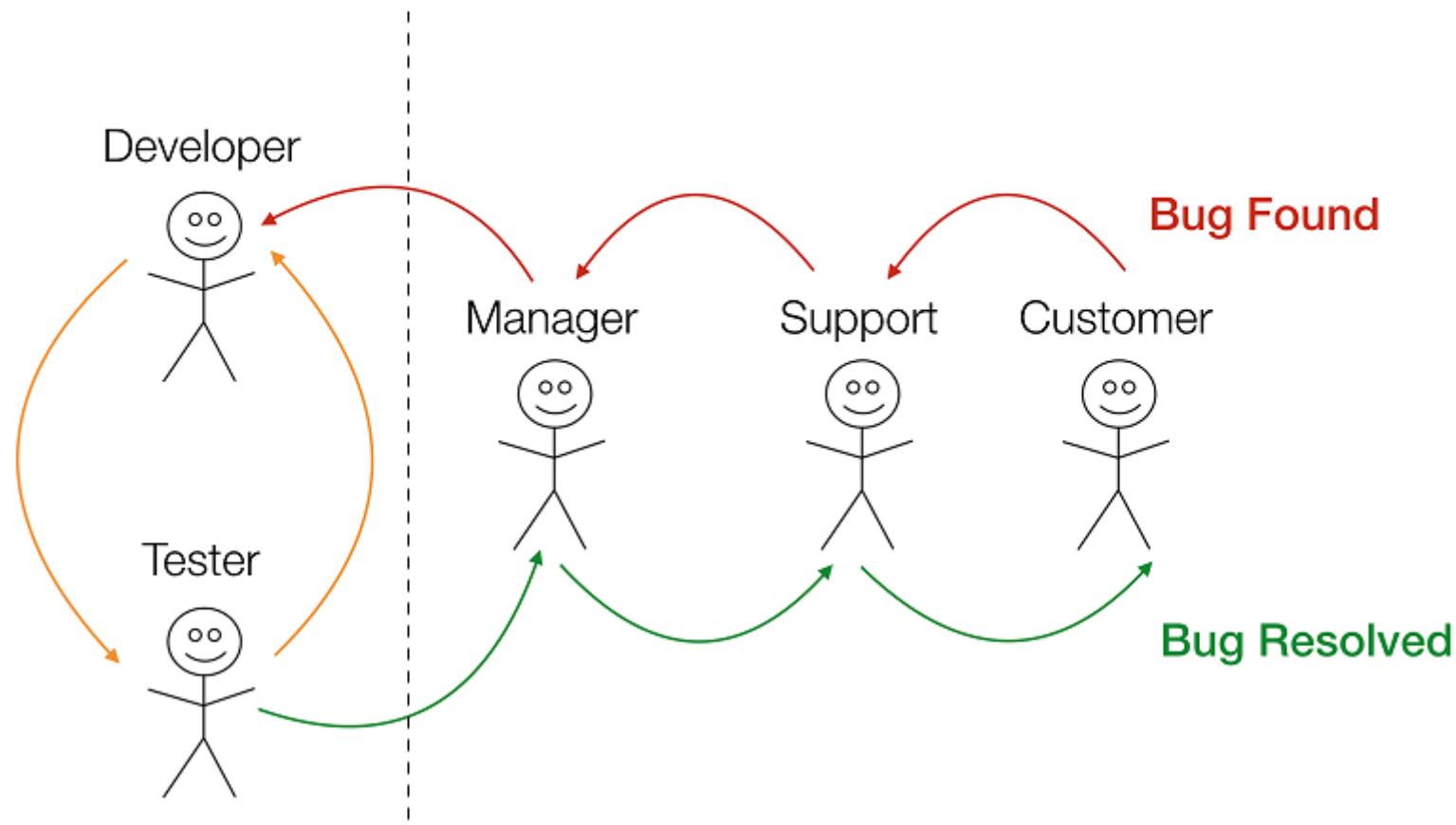
# ทำไมต้องเขียน-test ?

Test early, test often (the shift left approach)



# ทำไมต้องเขียนเหล ?

Bugs found after this line  
are way more expensive to fix.







ทำไมต้องเขียนเหล斯 ?

 เราก็ต้องเขียนเหส

TDD

Test Driven  
Development



# ควรเขียนเทสเมื่อไหร่ ?

รูปแบบการทดสอบ

(TFD) Test-First Development

ระยะสั้น ระยะยาว



(TLD) Test-Last Development





# ควรใช้รูปแบบไหน ?



# การทดสอบมีกี่แบบ ?

Penetration Testing   Boundary Testing   End-to-End Testing   A/B Testing   Performance Testing  
System Testing   Regression Testing   Functional Testing   Non-Functional Testing   Smoke Testing  
Load Testing   Exploratory Testing   Automation Testing   Manual Testing   Stress Testing   Sanity  
Testing   Unit Testing   Integration Testing   User Acceptance Testing



# ลองจัดกลุ่มดูซิ

## Functional Testing

Unit Testing Integration Testing System Testing  
User Acceptance Testing End-to-End Testing Smoke  
Testing Sanity Testing Monkey Testing ...

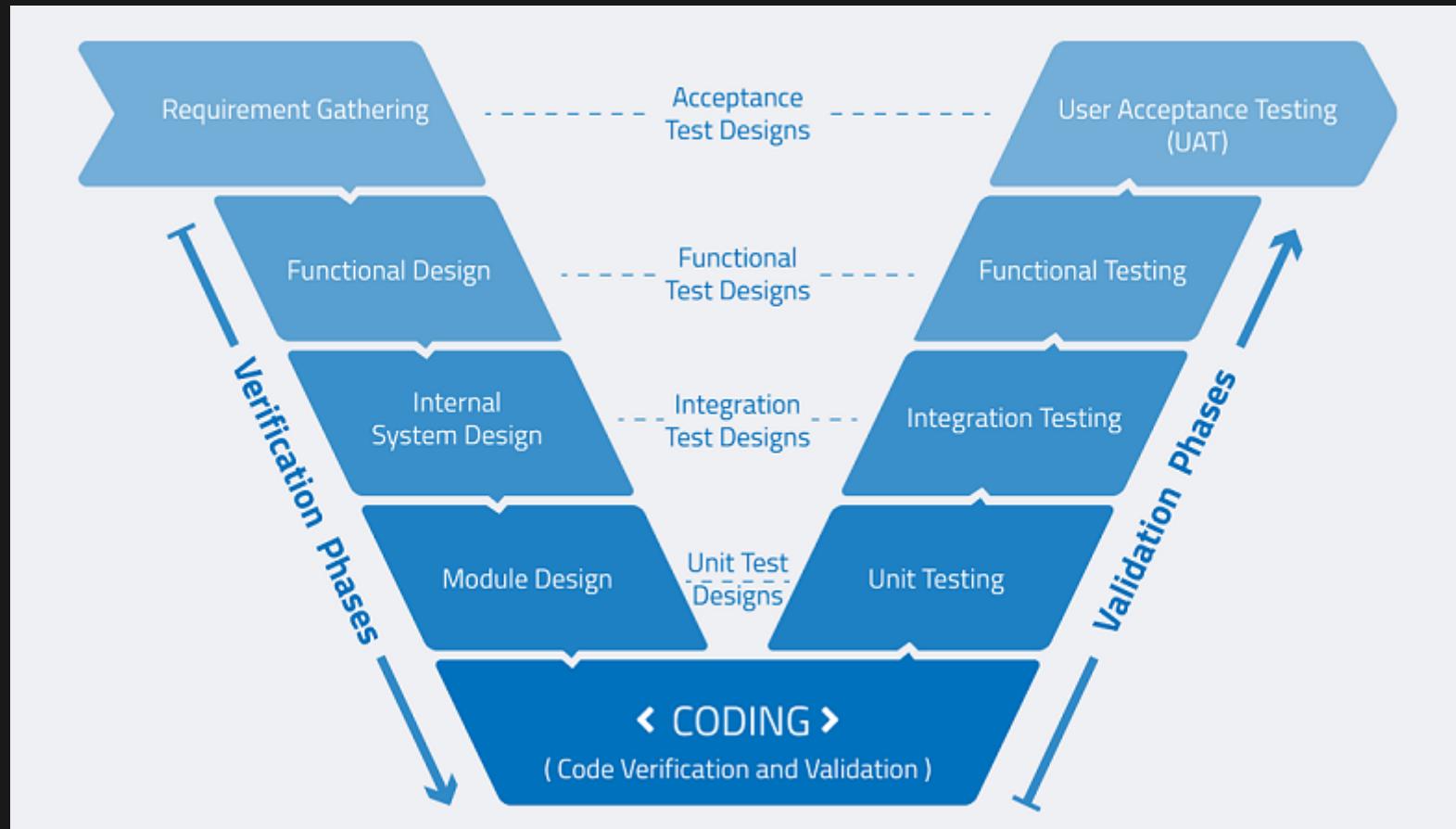
## Non-Functional Testing

Security Testing Penetration Testing  
Performance Testing Stress Testing Load  
Testing Exploratory Testing Usability Testing  
...



# มันต่างกันตรงไหน ?

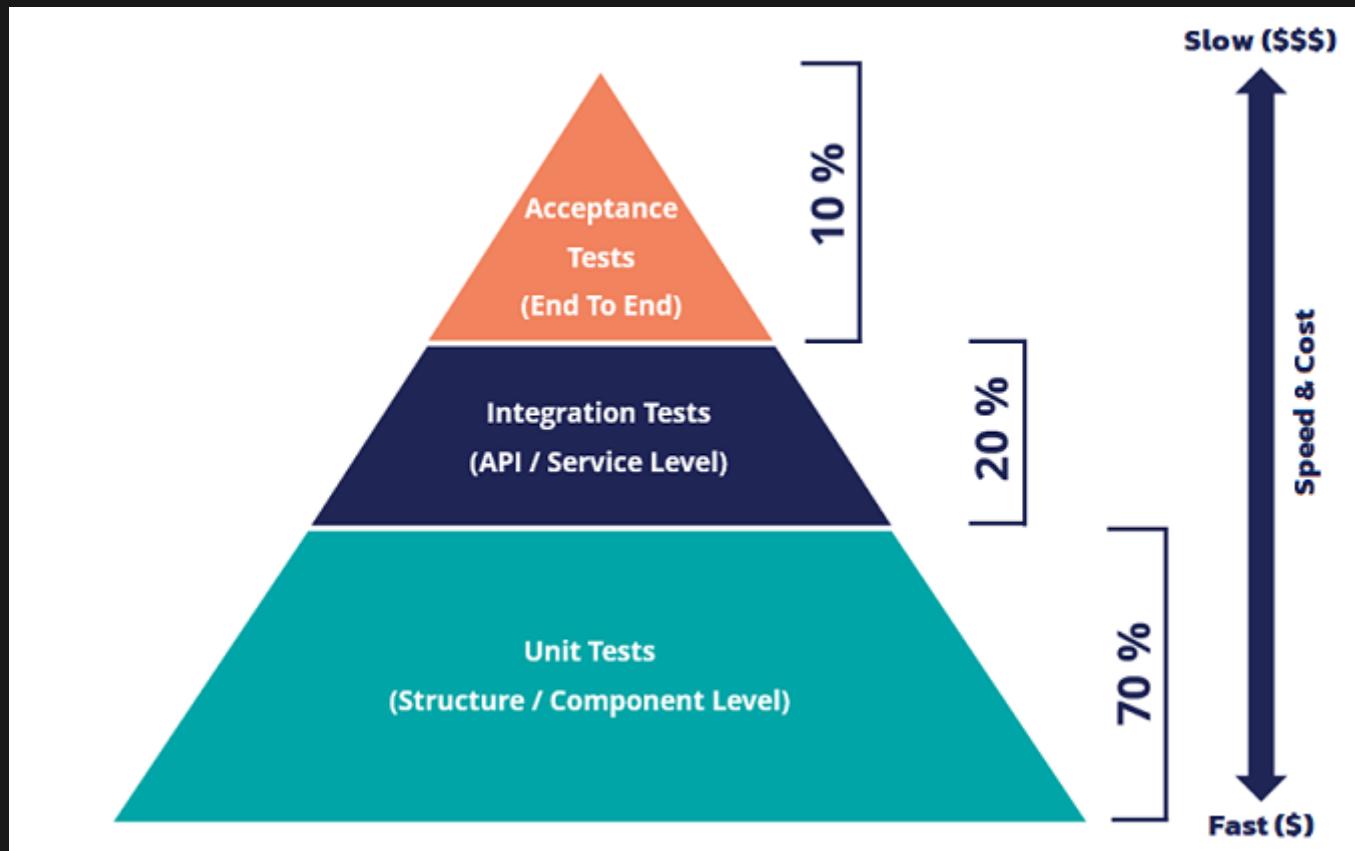
แต่ละตัวมีวัตถุประสงค์ต่างกัน ไม่มีอันไหนสำคัญกว่ากัน ขึ้นอยู่กับเราทำลังเน้น focus เรื่องไหน





# ควรลงนำ้นหนักยังไง ?

The Practical Test Pyramid





# Testing Frameworks

Unit Testing — Integration Testing — Acceptance Testing

xUnit NUnit MSTest SUnit JUnit RUnit CppUnit EUnit PerlUnit PHPUnit ...

## UI Testing

Playwright Selenium Watir Visual Studio Coded UI Testing Test Studio Silk Test ...



พูด yeอะป้าดหัว



## Challenge 01

# FizzBuzz

1 — 2 — Fizz — 4 — Buzz

Fizz — 7 — 8 — Fizz — Buzz

11 — Fizz — 13 — 14 — FizzBuzz

# TDD

“A **software development process** that relies on the repetition of a **very short development cycle**: **requirements are turned into very specific test cases**, then the software is improved to pass the new tests, only.” — Wikipedia

# หลักการเขียนเทส



การแก้ไขเทสจะไม่มีผลกระทบกับโค้ดที่ทำงานจริง

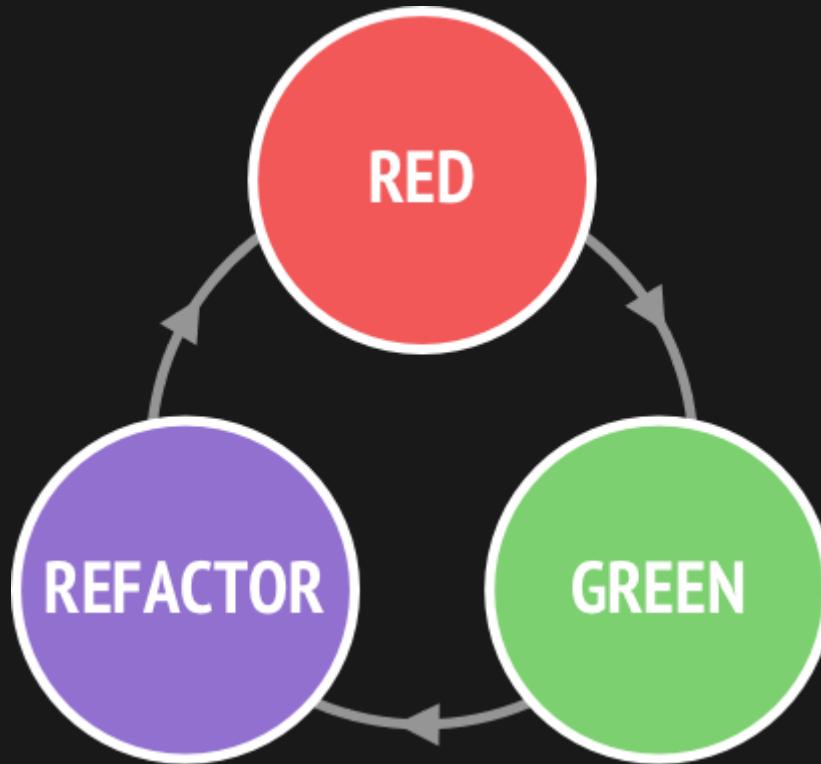
# AAA Pattern

- Arrange — เตรียมตัวก่อนทดสอบ
- Act — เรียกใช้ของที่ต้องการทดสอบ
- Assert — ตรวจสอบผลลัพธ์

ช่วยแยกขั้นตอนต่างๆ ไม่ให้ปนกัน + อ่าน/แก้ไข ง่าย

# Red → Green → Refactor

เพิ่มเคสใหม่ — เขียนให้พอผ่าน — ปรับแก้โค้ดให้ดีขึ้น



เทสเคสเป็นตัวขับเคลื่อนงาน



# Technique



2Dev 1PC

เข้าใจโจทย์ตรงกัน

ลดข้อผิดพลาด

แบ่งโหลดในอนาคต

แชร์ความรู้



## ພູດເຍຂອປາດໜ້ວ (Part 2)

# FizzBuzz

1 — 2 — Fizz — 4 — Buzz

Fizz — 7 — 8 — Fizz — Buzz

11 — Fizz — 13 — 14 — FizzBuzz

Pair Programming

การยืนยันความถูกต้อง

# Assertion

1. ตรวจสอบ ผลลัพท์ ที่ได้รับกลับมาจากการทำงาน
2. ตรวจสอบ พฤติกรรม จากการเรียกใช้ functions
  - Object states - fields, properties
  - Communication - method calls between objects



# แนวคิดในการเขียนรหัส

ลำดับ	กรอบความคิด
1. Normal cases	กรณีทั่วไปที่ผู้ใช้ส่วนใหญ่จะได้เจอบ่อยๆ
2. Alternative cases	กรณีที่นานๆ ครั้งจะเกิดขึ้น เคสขอบต่างๆ
3. Exceptional cases	กรณีข้อผิดพลาดที่อาจเกิดขึ้นกับระบบ

ไม่จำเป็นต้องคิดจนครบทุกรายละเอียด เนื่องจากมีเวลาค่อยกลับมาเพิ่มเติมได้

# ลดปัญหา ก่อนเขียนโค้ด



ตัดก่อนตาย เดือนก่อนวายวอด



# Challenge 02

# Login with Backoff

Username : Admin1, Admin2, Admin3, ... , Admin9

---

Password : P@ssw0rd

1 ~ 3 (none)

4 ~ 6 (30 sec)

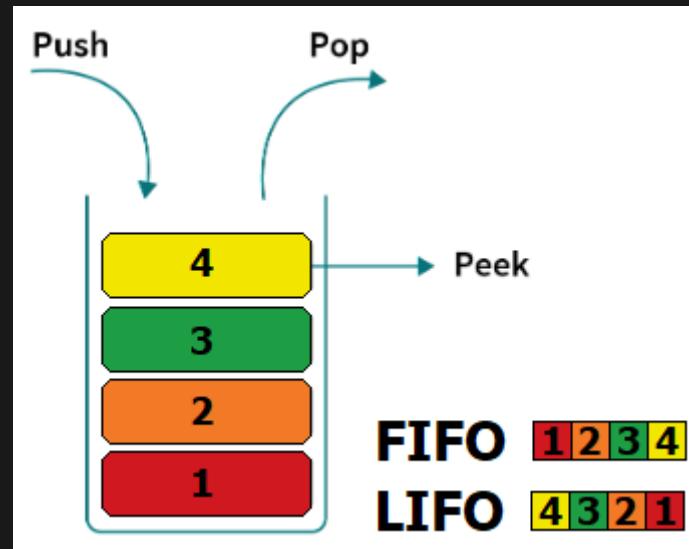
7 ~ 9 (3 min)

10+ (30 min)



# Challenge 03

## STACK



Push

Pop

Peek

—

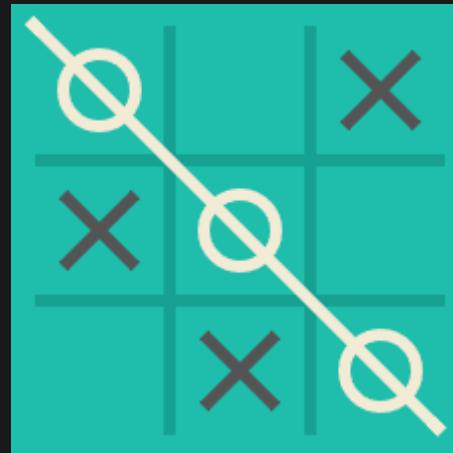
FIFO

LIFO



# Challenge 04

# Tic - Tac - Toe



Console app — Web app



# Summary

1. ปัญหาหลักของบริษัททำซอฟต์แวร์ → การจัดการ

- รับงานใหม่ แก้งานเก่า จัดการทีม บริหารสภาพคล่อง
- Maintenance คือค่าใช้จ่ายหลัก

2. เกิดครึ่งของปัญหาในทีมเดฟเป็นเรื่อง Technical Debt

3. ความสำคัญในการเขียนเทส (Shift left approach)

4. TFD & TLD

5. Functional Testing & Non-Functional Testing

6. Unit testing & Pair Programming

7. Red - Green - Refactor