Conclusions & Summary
關於單元測試的各種....

1. No test cases, no quality !!!

現在你們的 product 總算有一堆 unit test cases !! 而且還自動化 !!

跟 QA 的 test case 不一樣，這些 test case 都是 code !!

Code 的維護問題，有時候比 QA test cases 麻煩了一點

人多了就有江湖，code 多了就有軟體工程問題

Unit test case code management (江湖)

- 你會開始遇到 test case 放哪裡的問題 (團隊內有規則?)
- 你會遇到 test case 命名的問題 (公司內，有命名規則?)
- 你會遇到 test case 維護的問題(團隊內有紀律?)
  - 一忙起來，直接把壞掉的 test case disable 的故事
- 你會遇到 test case 品質的問題 (品質有沒有 review? 有沒有最低標準)
  - test cases 也是可以應付交差了事的，例如可能連最基本 code coverage 都沒有達成
- 你會遇到某些程式碼不曉得如何把其 unit tests 寫出來 (code quality?)
  - 先看看你是不是讓一個 function 做了太複雜的事情
- 你會遇到 test case 很玻璃(fragile)的問題 (有些要先 design 才有救!!)
  - 小小改個程式碼，你要修正十幾個 test case 壞掉，要修正...
  - 通常這表示你們的 design (有問題)

Test Cases are built to fail

當測試壞掉的時候，完全是好事情 !!

其實這代表軟體工程成熟度的躍進

有此一說:單元測試不是測試

- 摔柔道之前，要先學會被摔
- 確保被摔的人懂得保護自己不會受傷，才能開始學摔人的技巧
- 在學重構，設計模式等等新技術時，如果不先學保護自己的技巧，隨便改個一兩行就出 bug 誰敢改 code?

Unit Test 要不要包含 DB?

- 總體來看，Unit Test 不管怎麼定義，至少「不要牽扯到外部系統」這點應該是無庸置疑
- 幾年前來說。DB 是珍貴資源，同一個測試環境，RD也要開發，QA 也要測，DB 就是那麼一套還沒有版控
- 但是近幾年，container 技術的普及，schema 管控工具的進步，製造一個隔離的測試環境，遠比以前簡單多了，代價也廉價非常多

理論與實務上的限制

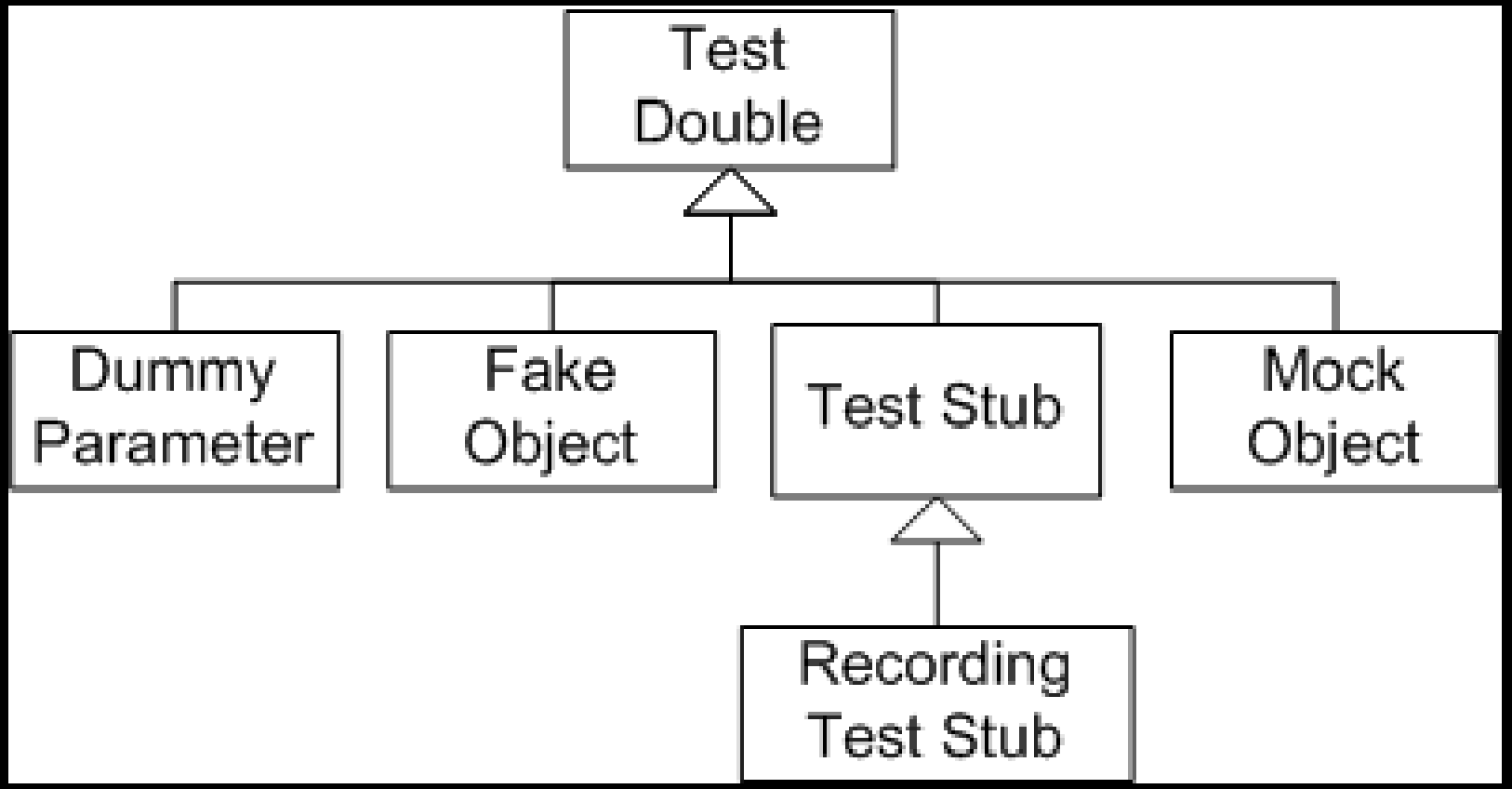
- test in isolation 是個 common practice
- 但是沒有人阻止你用來做 integration testing
  - 例如撰寫一些測試，實際上貫穿幾個工程師的程式碼。例如一個 package 內的程式碼，但是又還沒有到完整的 e2e test
  - 通常有必要，例如這個 package 的邏輯常常更動，人員來來去去，每次更動就會需要重新檢測。那沒有什麼懸念，就請你把 integration test 用 xUnit framework 建立起來。立即加入這些 test 到 DEVOPS pipeline

沒有自動化測試的敏捷，就是搞死 QA 而以

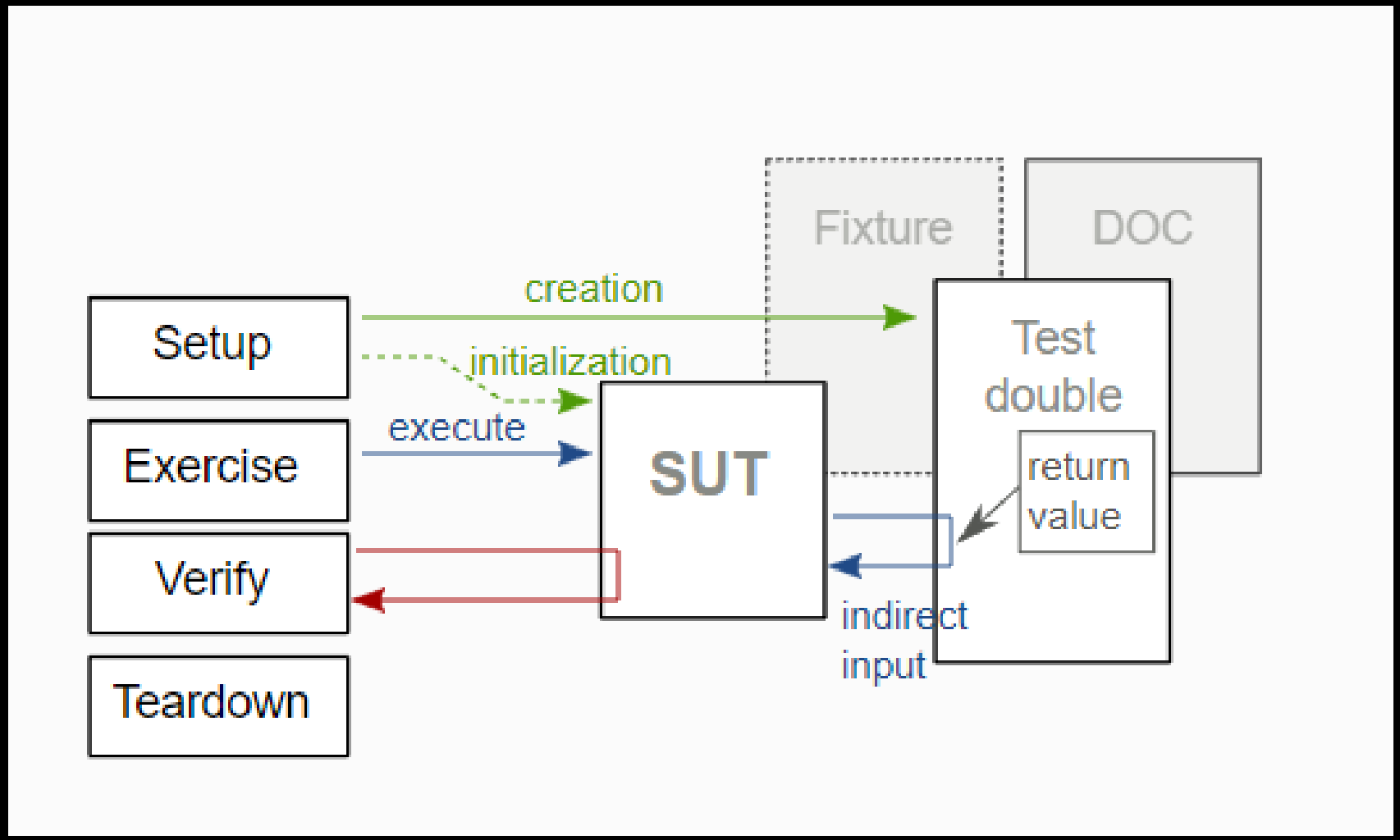
- 假設你第一個 sprint 要推出10個功能吧
- QA 要測10個不為過
- 假設 RD 輸出很穩定，下個 sprint 也出10個功能
- **請問 QA 要測多少功能?**

test isolation
名詞總整理

Mocks and Other Test Doubles



FYI，目前的單元測試使用的名詞尚未底定，所以請各位自行區分其定義


Fixture
DOC
creation
Setup
initialization
execute
Exercise
SUT
Test
double
return
value
Verify
indirect
input
Teardown