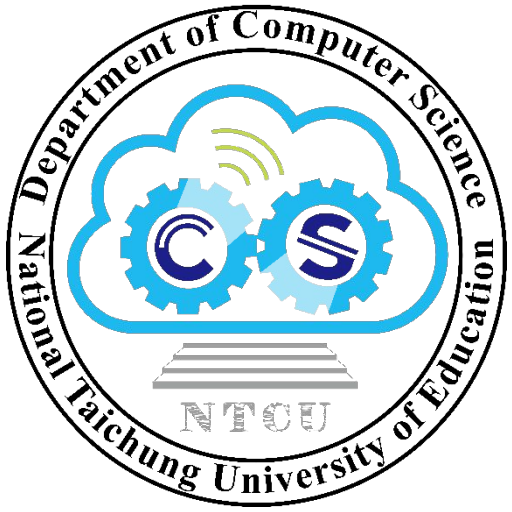


軟體工程概念與文件撰寫簡介

July 2024

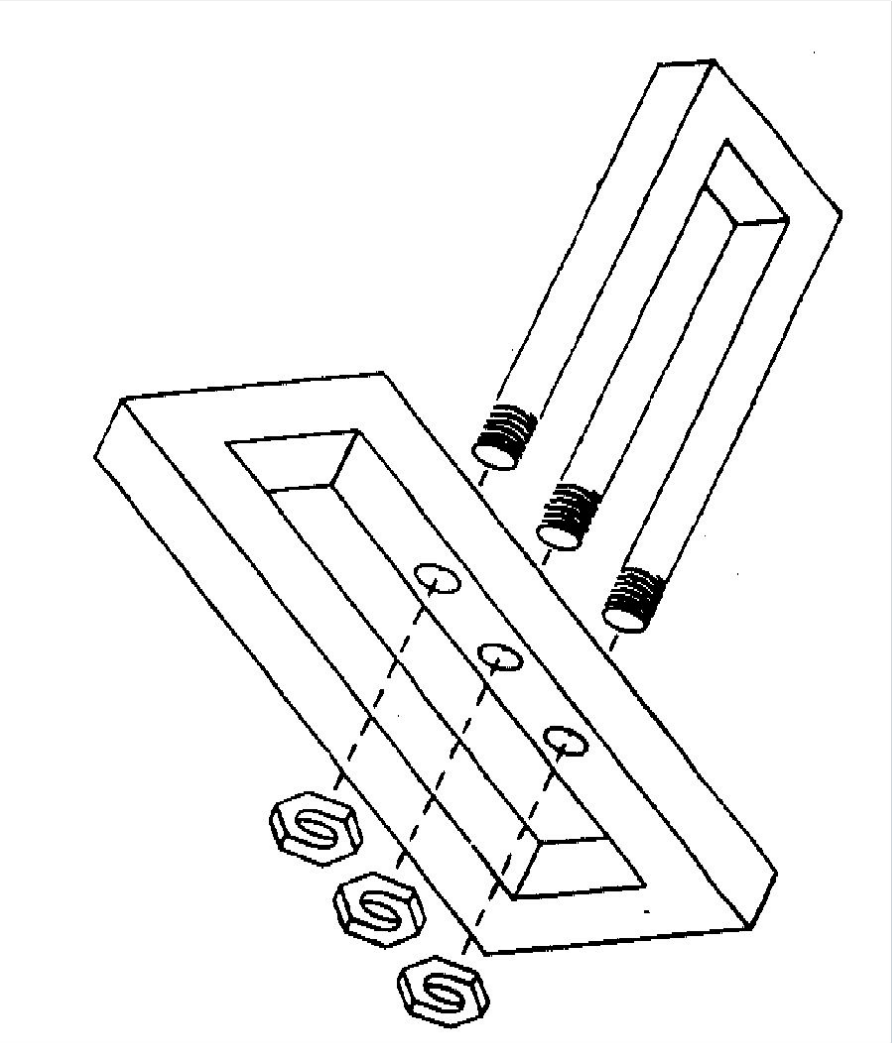
臺中教育大學資訊工程學系副教授
徐國勛
E-mail: glenn@mail.ntcu.edu.tw

軟體工程概念

為什麼需要軟體工程？

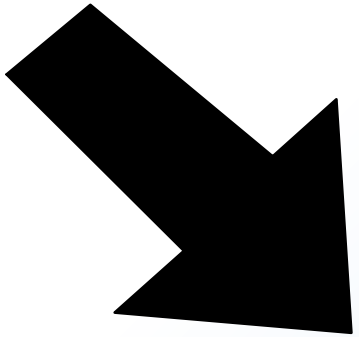
只要會Coding不就夠了嗎？

Can You Develop This?



Limitations of Non-engineered Software

需求



There is a huge gap between requirement & software

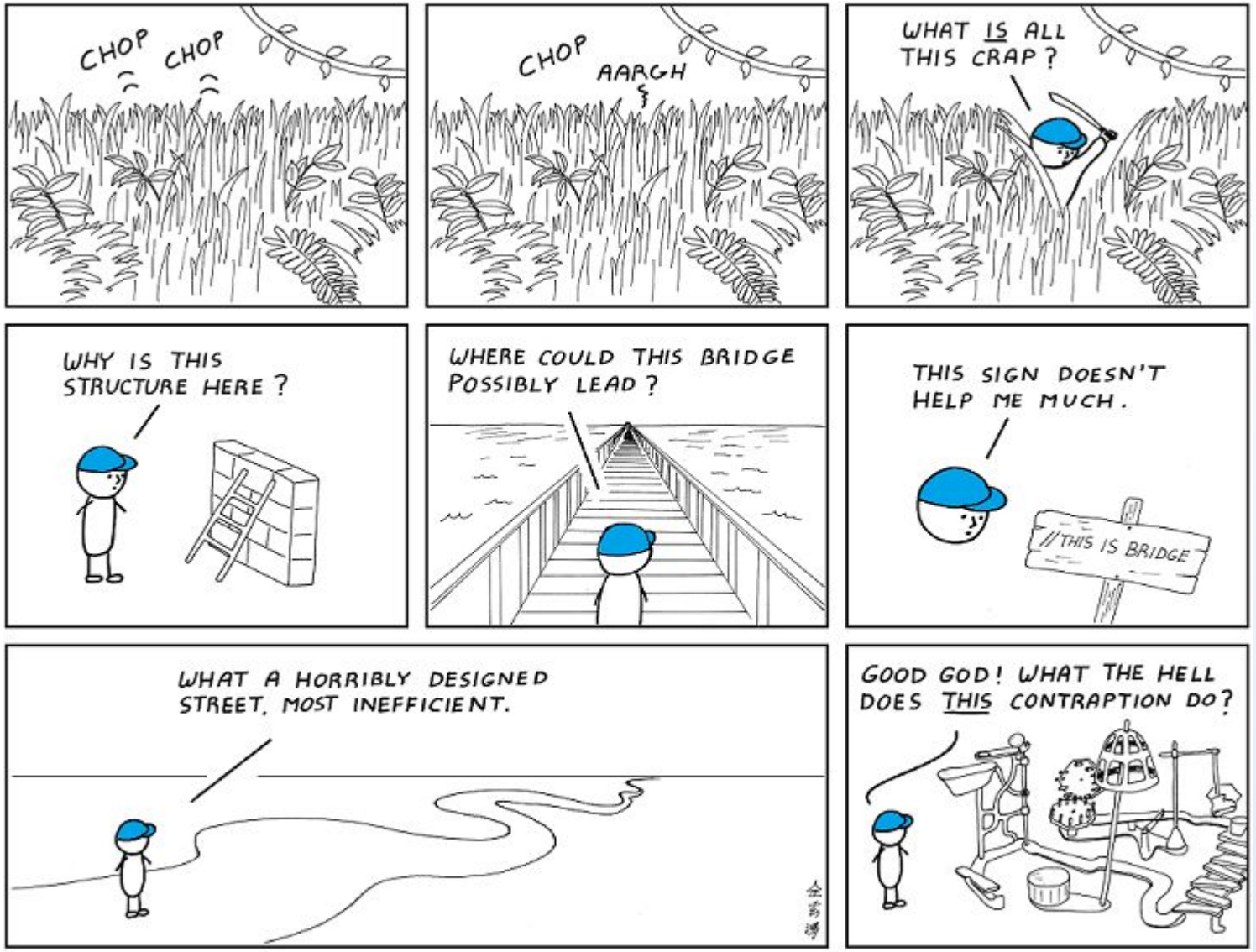
軟體

而且，現今軟體已不再是由一個人單打獨鬥寫出來的...

所以你會需要看懂別人的code

在AI時代，你可能還要需要看得懂ChatGPT產生出來的code

How Do You Deal with Other People's Code



I hate reading other people's code.

https://abstrusegoose.com/432

除此之外，過往的失敗經驗也告訴我們

- 美國銀行於1982年進入信託商業領域，並著手規劃發展 信託軟體系統。
  - 規劃時程：花了18個月進行該系統之規劃及分析。
  - 計畫原訂預算：2千萬美元；
  - 原訂開發時程：9 個月，預計1984/12/31完成；
  - 開發期間：直至1987年三月都未完成本系統，並已經投入6千萬美元。
  - 後果：失去了6億美元的信託生意商機，最後因為此系統並不穩定而不得不放棄此系統，並將340億美元的信託帳戶轉移出去。

- 從1995年的調查報告中(by Standish Group)
  - 以美國境內8000個軟體專案為調查樣本
    - 有84%的軟體計劃無法於既定的時間及經費當中完成
    - 超過 30%的計畫執行到中途被取消。
    - 專案的預算平均超出189 %。
- 其他案例：
  - 1996年，亞利安五號原型爆炸。
  - 1998年，波音Delta III火箭爆炸。

除此之外，過往的失敗經驗也告訴我們

- 電腦錯亂存戶A三千多萬，一銀只討回二十萬 (2003)
  - 男子九十二年七月到第一商銀繳交增資卡循環信用貸款利息二千六百元，因櫃台疏失在電腦上多打了一個「1」，造成電腦程式錯亂。
  - 他在兩個月內於自己帳戶提領378次、共三千三百六十三萬餘元，被告等八人雖遭判刑，但銀行只討回廿餘萬元。

https://www.ptt.cc/bbs/Bank_Service/M.1183195444.A.9C7.html

- 廣東銀行出錯多發工資，三百人爭相提款 (2008)
  - 銀行第一次轉賬時，電腦提示交易失敗，之後銀行又轉帳一次，但其實第一次轉賬是成功的。
  - 由於銀行出錯，該廠超過八成員工多發了雙倍甚至四倍工資，共有三百多名員工多發工資。

http://www.epochtimes.com/b5/8/2/28/n2026834.htm

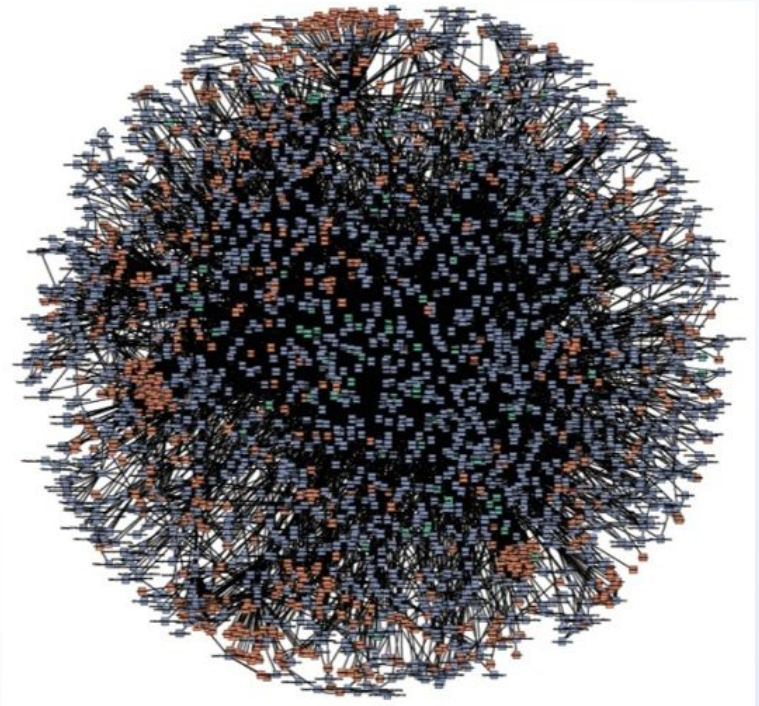
- 南山人壽(2019年)，花費超過一百億的「境界成就計劃」升級計畫失敗，還被罰千萬

https://www.cmmedia.com.tw/home/articles/18772

Why?

軟體的本質問題:複雜性(Complexity)

- 軟體系統之複雜程度,往往隨著程式的大小及軟體元件個數以非線性的方式,甚至是等比級數的方式遞增。例如:系統的狀態、程式碼大小、系統功能、程式靜態結構、系統動態行為等各層面,會越漸複雜。

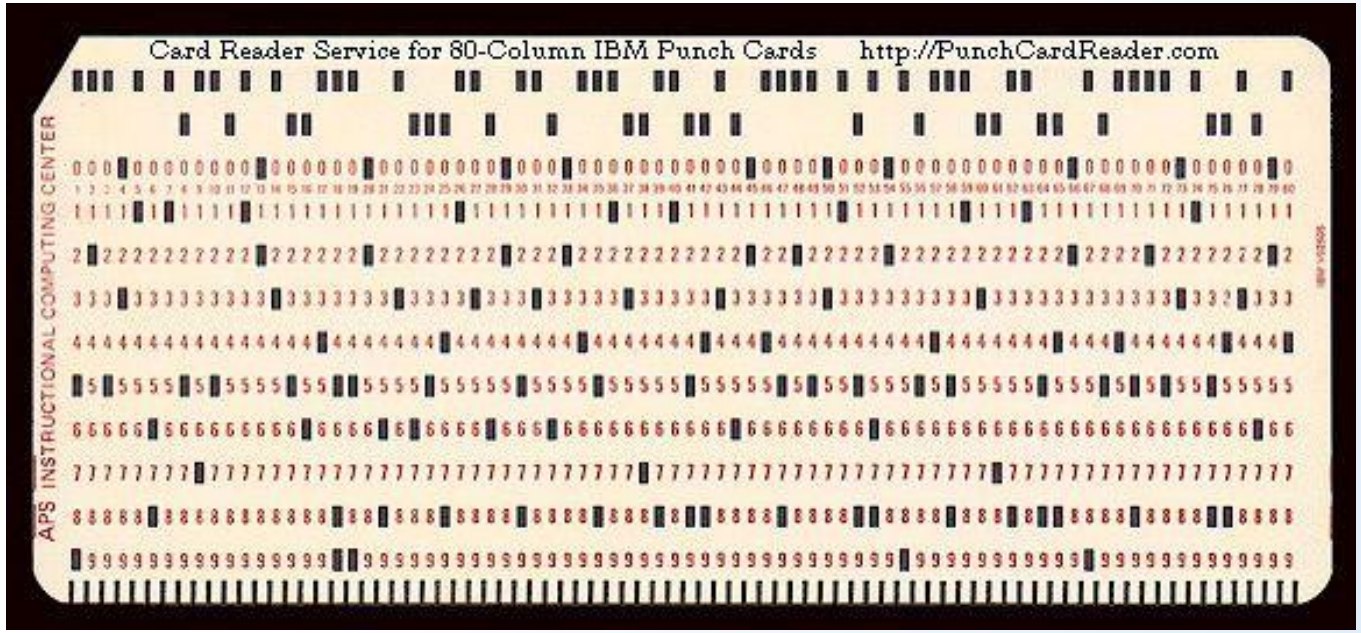


軟體的本質問題：易變性(Changeability)

- 為了滿足客戶的需求，一套成功的軟體系統，從開發到完成、從產品交付到營運維護，隨時都有可能要做變更。

軟體的本質問題:隱藏性(Invisibility)

- 軟體本身是看不到、摸不著的，導致需求容易存有誤解、疏忽的地方不容易被發現，而大大的妨礙了彼此溝通的進行。



軟體的本質問題:一致性(Conformity)

- 在大型的協作環境下發展軟體系統, 介面跟介面間、模組跟模組間、系統跟系統間的介接, 便都存有一致性的問題需要解決, 因此需要透過各種方式來轉換或是介接不一致的地方。

我們需要一些方法和工具來協助我們！

什麼是軟體工程?

Software Engineering

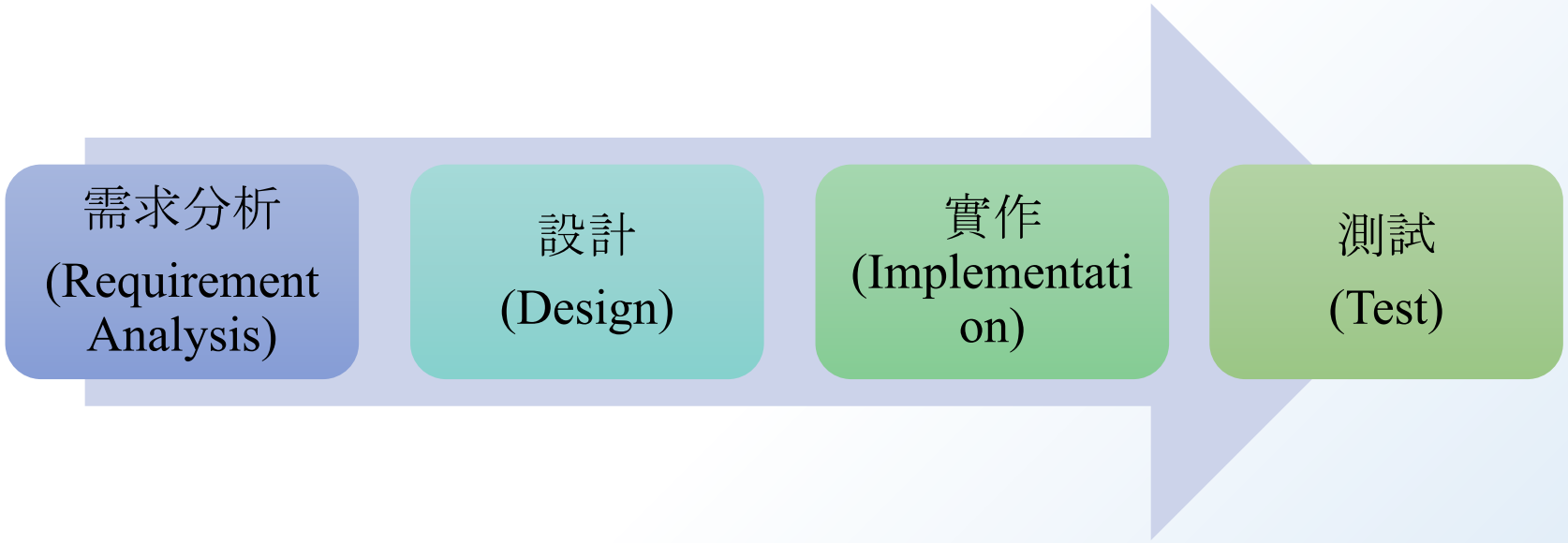
Real World

Software World

Software engineering is a discipline that integrates methods, tools, and procedures for development of computer software.

- Methods: introduce a way to build software
- Tool: automated, semi-auto support for methods
- Procedure: defines the sequence in which methods will be applied, the controls that help ensure quality and coordinate change

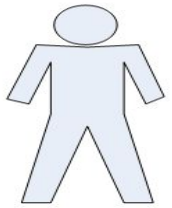
典型軟體開發流程


需求分析
(Requirement Analysis)
設計
(Design)
實作
(Implementation)
測試
(Test)

DevOps開發流程


系統驗證
研發設計
程式碼品保
需求管理
反饋機制
git
JUnit
Powered by
JACOCO
Java Code Coverage
checkstyle
gradle
FindBugs
REDMINE
flexible project management
Dev
Ops
code
plan
release
deploy
Operate
monitor
test
build
Jenkins
Pmd
DON'T SHOOT THE MESSENGER
sonarqube
FORTIFY
Reports

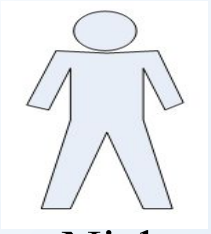
需求是什麼?值得我花時間去撰寫或管理嗎?



福委會

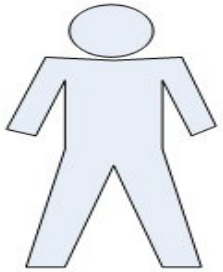
請資訊中心開發一個簡易的投票系統, 功能是「只要讓使用者點選最想去的地點即可, 選項有三個 (1)墾丁;(2) 台東;(3) 澎湖」

Nick花了一個晚上, 便利用PHP完成了系統的建置。三天後系統便上線了。



Nick

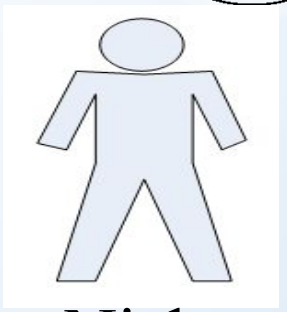
故事就此展開 …



Maggie

Nick, 很多同事反應三個地方太少了, 請再幫我加一個花蓮

不行啦, 我資料庫都寫死了, 就是三個欄位, 這樣我整個程式都要重寫, 而且我現在正在趕業務部的專案估算系統, 我哪有時間幫你改？



Nick
(資訊中心)

不過只是多打『花蓮』兩個字而已？

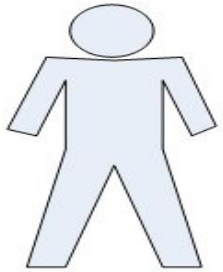
Maggie無法理解為什麼多加一個選項有這麼困難
Nick太難溝通了。

好吧！那你想清楚喔, 就是四個…還是我幫你預留五個好了, 不要再變了！

Nick答應熬夜修改程式

Nick調到人資部門
投票系統交給新進的 Kimi負責

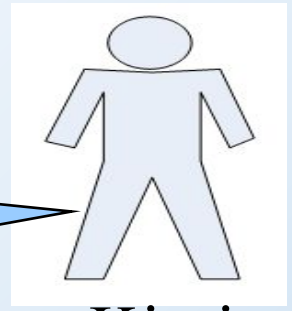
其他單位多次詢問是否可修改



其他單位

可否修改系統功能？

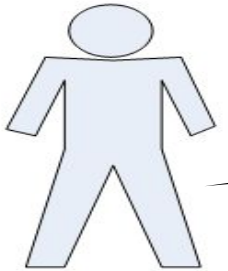
好好！我修改看看！



Kimi
(資訊中心)

由於沒有任何說明文件, Kimi甚至不知道這個系統有那些功能, 能不能修改, 只好答應先答應他們的要求再慢慢地修改。
系統於是越做越大, 功能愈來愈強大。
但是因為疊床架屋的關係, 修改起來就愈來愈困難了。

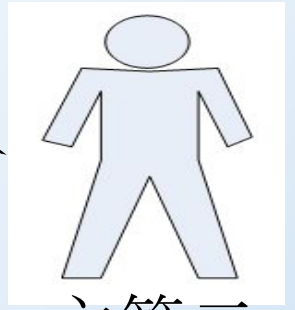
公司決定讓使用者透過網路的方式來調 查使用者對每一個 產品的滿意度，時間相當的緊急，且不容許將系統外包給其他公司來做。



主管一

其實問卷調查系統和投票系統差不多，都是有要求使用者從多個選項中選擇一個

資訊中心的投票系統現在的功能很強大，很多東西都可以透過設定來做，滿有彈性的。而且也是走Web的架構，應該可以用



主管二

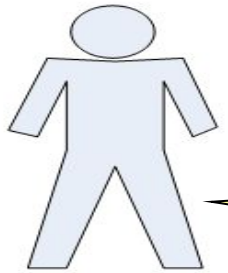
可是他們的系統好像沒有檢查一人一票，安全性也沒有檢查

那就請資訊中心加上這個限制吧，很簡單的，改一改程式就好了

資訊中心的鄭主任把這個工作接下來，並指派給Kimi，要求一個月完成。

Kimi手頭上還兩個新的軟體專案要寫、一個系統導入要進行。雖然他覺得一個月的時間要完成這個專案有點太急，但也只能硬著頭皮接下來。

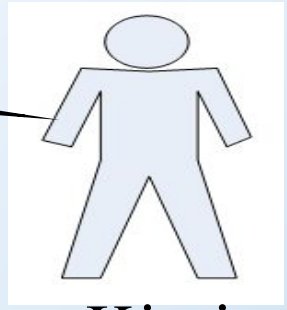
Kimi上網看了幾個問卷調查系統的樣子，發現形式也不少。於是打電話給產品部的江經理，訪談一些問題。



江經理
(業務部)

請問選項需不需要可以複選？需不需要輸入受訪者的年齡、地區、性別等？

應該可以不用吧，這個業務可能要問Paul。或者你把功能做出來，到時候我們要輸入就輸入，不輸入就跳過吧，這樣比較有彈性！



Kimi

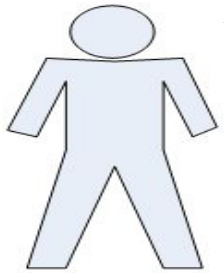
Kimi發現問了等於白問，在時程這麼緊急的情況下，當然選擇把這個功能省略。

五月初系統開發完成。Kimi開始展示給各單位來看

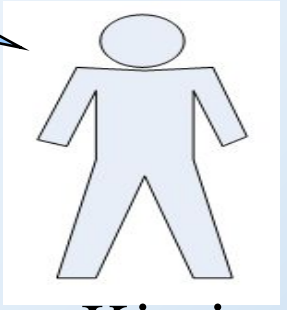
怎麼不能輸入訪員的姓名？這樣我們沒有辦法控管問卷的狀態阿！

可是當初也沒有說要做這個功能啊？

這還需要說嗎？這是common sense吧！



Paul

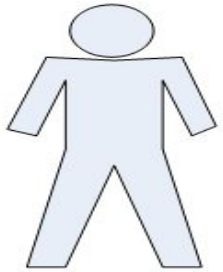


Kimi

Kimi不敢說什麼，只能說是啞巴吃黃連，有苦說不出。一邊翻開筆記做個註解，一邊準備繼續展示系統。

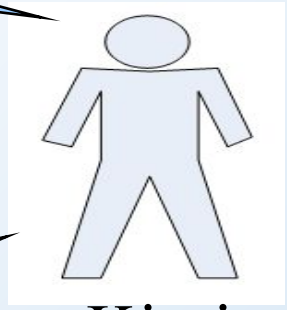
這個問卷只能單選，不能複選？

對啊！（江經理說不用的吧！）



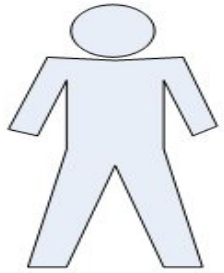
Paul

上次江經理不是有跟你們說要加這個功能嗎？



Kimi

對阿！我有說喔！



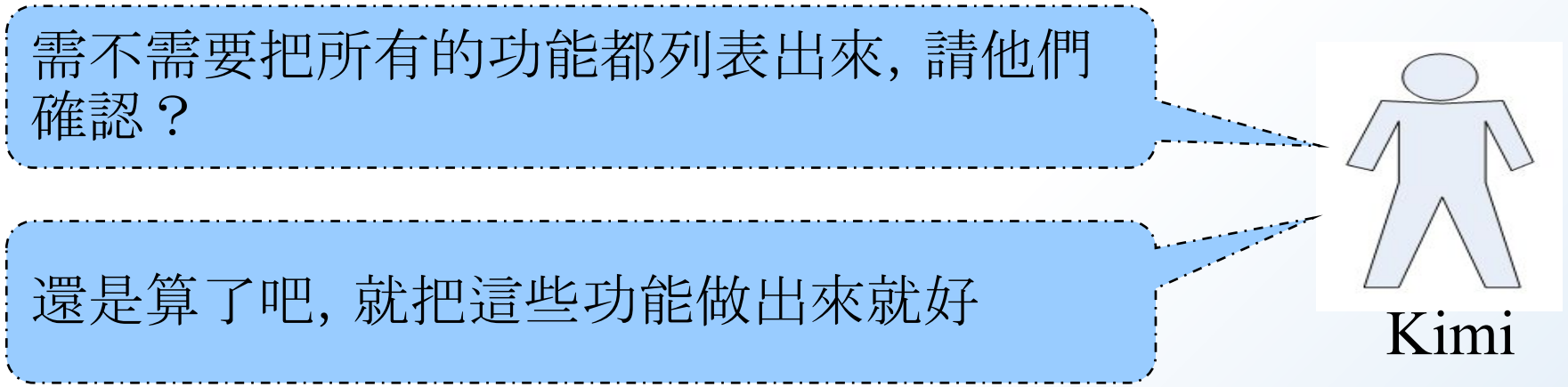
江經理

可是上次江經理是說應該不用做！

我是這麼說沒錯，但是我也說要把它寫出來，到時候用不到就不用

時間這麼趕，這些功能做不出來啦。當初也不說清楚！

Kimi在上級的壓力下還是回去把系統趕了出來。

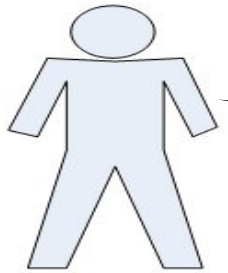


2007年五月底，Kimi再做一次的 demo。
這一次的問題比較少，多半是做一些畫面的調整，各單位對這個系統顯然是很滿意。
於是開始在公司的入口網站公告六月份要開始做市調，並且有抽獎送手機的活動。

系統預計於六月份的第二個星期一上線，星期五的晚上，Kimi收到Paul的電話。

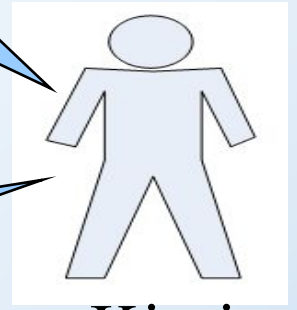
Kimi，幫我一個忙，加上一個可以跳題的功能

什麼是跳題？



Paul

就是某一題如果回答否的話就直接跳到後面第幾題，而不用接著回答。拜託，這個功能很重要



Kimi

可是星期一就要上線了。鄭主任知道這件事嗎？

我還沒有跟他說，我等一下打電話給他，你趕快先改

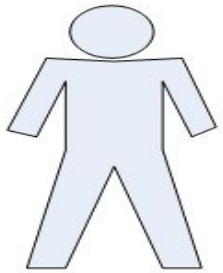
Kimi只好犧牲週末的假期把系統趕出來。經過簡單的測試後，他認為沒有問題，就送個簡訊給 Paul:「系統ok了。」

星期一系統開放後的流量還算正常。但是由於系統的畫面設計的不流暢，許多使用者操作到一半便關掉瀏覽器。到了下午以後，公司開始收到電話，抱怨寫到一半系統就當掉了，關掉重新投票卻又沒有辦法登入。到了晚上，整個系統當機完全沒有辦法運作。

公司總經理因此大怒，把鄭主任叫來臭罵一頓。鄭主任找來Kimi出氣。

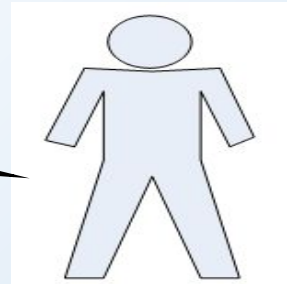
因為前一天臨時改了跳題的功能，功能沒有寫好，好像會有無窮迴圈

什麼跳題？我怎麼都不知道？



鄭主任
(資訊中心)

Paul要我改的，他說會跟您說，要我先改



Kimi

根本沒有跟我說。你的老闆是我還是江經理，人家叫你改你就改，還改到系統當掉。明明一個做好的系統…

Kimi想到犧牲的假期與熬夜的趕工，換來的卻是一頓罵，心中非常不是滋味。

系統最後宣告失敗。

因為這個系統的失敗，鄭主任被調到其他的單位。新任的馬主任是由其他公司轉任，擁有多年專案管理與軟體開發的經驗，

針對投票系統的問題進行分析，發現公司的技術人才沒有一套完整的程序與方法來進行需求的擷取、訪談與分析，所有的功能都是透過急就章的方式片面的拼湊起來，缺乏整體性的思考與討論。進一步的分析，其實不僅是問卷系統如此，公司多半的系統都是如此疊床架屋而起。

資訊中心決定全面檢討整個系統開發的流程，於是找來較為資深的同仁一起討論各類系統的開發模式。

故事暫時落幕 ~~~

想想看
如果你身處這樣的環境，你要怎麼應對呢?

需求分析(Requirement Analysis)

- 了解客戶的需求、分析系統的可行性、分析需求的一致性以及正確性等。
- 重點是”What”。
- 通常會撰寫需求文件(SRS)。

你可以先訂出使用案例

- 透過情境思考，站在使用者操作系統的角度，思考系統該具備怎樣的功能，進而引領需求的分析。
- 先把故事(Scenario)說出來！

接著訂出功能需求

- 具體提出系統應該提供的服務項目。
- 系統是否具備這些功能需求是非常明確的。

功能需求列表範例

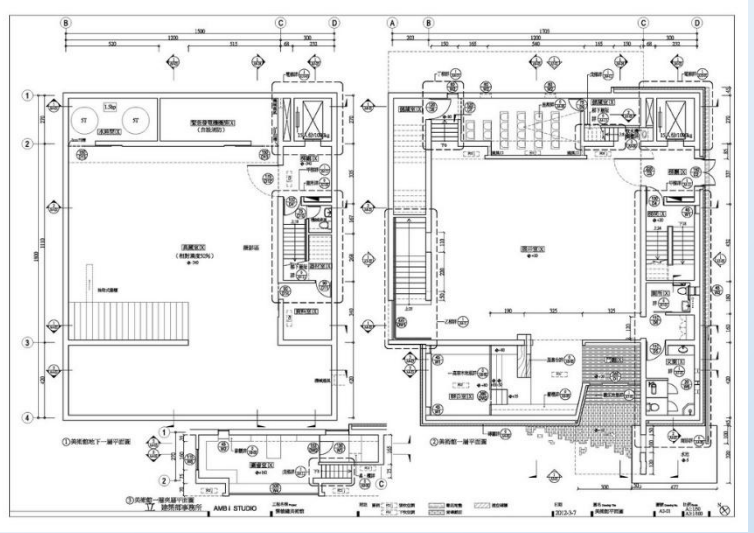
REQ-001	系統管理員可進行投票系統各單位管理帳號建立刪除/修改各帳號權限(系統管理員/單位管理員及承辦人)設定，單位管理員可進行所屬單位帳號建立/刪除/修改各帳號權限(單位管理員及承辦人)設定，單位管理員具有承辦人權限。承辦人員可進行所屬單位建立之所有選舉維護。
REQ-002	承辦人員可建立/修改/檢視選舉基本資料(選舉名稱/選舉時間/投票票數/當選人數/備選人數)。
REQ-003	承辦人可設定選舉的選舉人條件(條件:在職日期/性別)及範圍(可設定包含及除外之範圍，範圍之單位人事資料庫主類別/次類別/單位系所/職稱/姓名)，並根據以上條件產生選舉人名冊，且可匯出成EXCEL。
REQ-004	承辦人員可新增(需註記)/取消 (資料未刪除，但註記不符合選舉人資格原因)選舉人名冊內的選舉人，且可匯出成EXCEL。
REQ-005	承辦人員可設定選舉的候選人條件(條件:在職日期/性別)及範圍(可設定包含及除外，範圍:人事資料庫主類別/次類別/單位系所/職稱/姓名)，並可設定候選人組合，並根據以上條件產生候選人名冊，且可匯出成EXCEL。
REQ-006	承辦人員可新增(需註記)/取消 (資料未刪除，但註記不符合候選人資格原因) 候選人名冊內的候選人，且可匯出成EXCEL。
REQ-007	選舉人可於REQ-001設定的選舉時間登入系統，並顯示可參與之選舉，候選人名冊(含選舉編號/單位/姓名欄位)以單位排序並提供搜尋功能(單位/姓名)，選取候選人進行投票，並需提示選舉人可投票數，且投多票僅限一次投完，不可重新投票。
REQ-008	提供報表讓承辦人員檢視候選人得票狀況(欄位含:選舉編號/姓名/單位/職稱/票數)，於備註標示當選及備選，且可匯出成EXCEL。
REQ-009	提供報表供承辦人員檢視選舉人投票狀況，並顯示已投票人數應投票人數/未投票人數/投票率，且可匯出成EXCEL。
REQ-010	採用PORTAL帳號登入，記錄安全查核資訊，並顯示選舉公告。

再訂出非功能需求

- 強調對於系統品質的要求與限制，或者是說系統應該具備的特性，例如，可靠度、安全性等品質性的要求。

設計(Design)

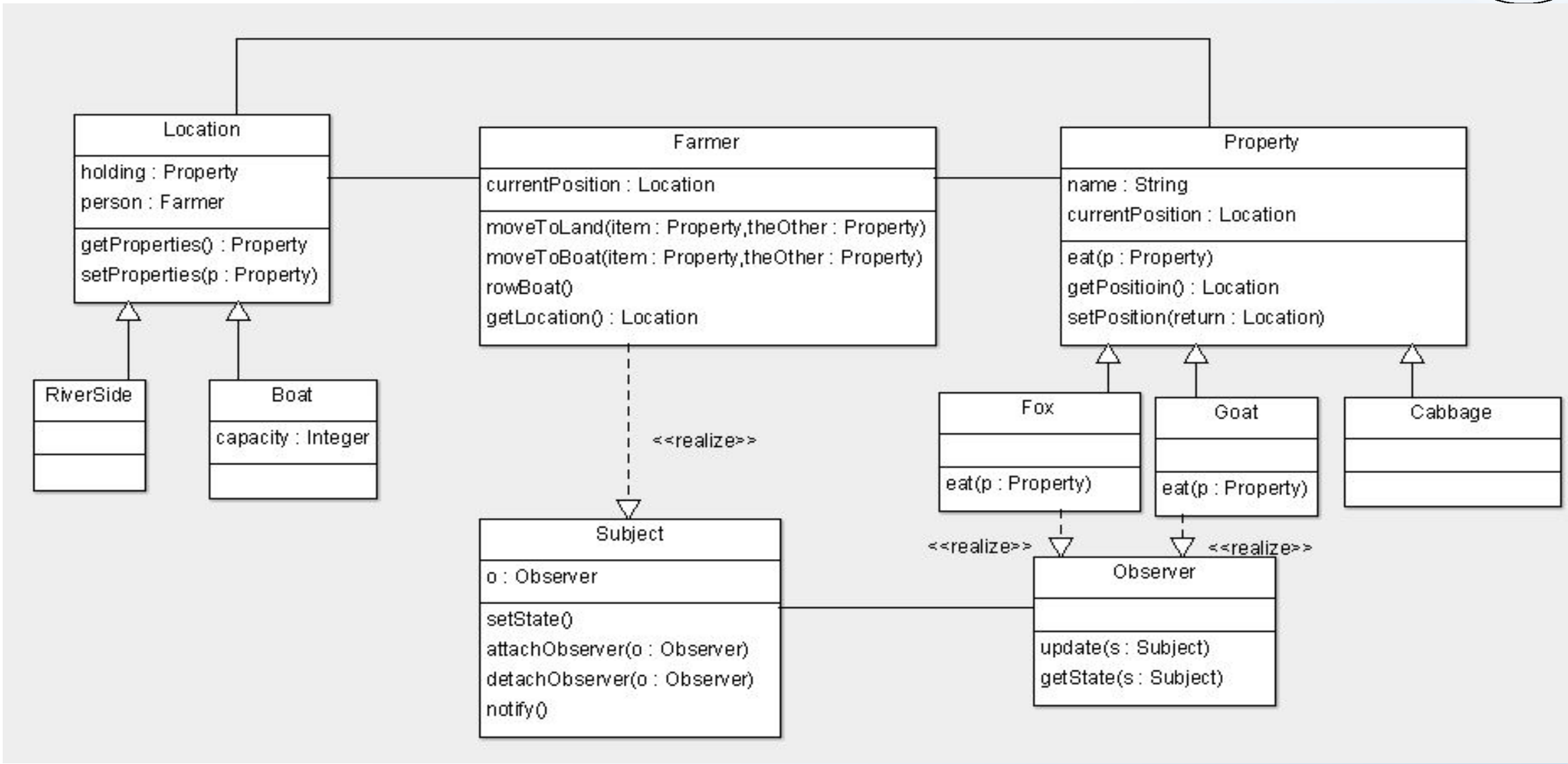
- 將需求轉換為系統的重要過程。
- 包含架構設計、模組間的介面設計、資料庫設計、演算法設計與資料結構設計等。
- 重點是"How"。
- 有時會考慮日後的"Change"。
- 通常會撰寫設計文件 (SDD)。



範例

- One farmer, named Joe, and his belongings(財產), a fox(狐狸), a goat(山羊), and a cabbage(包心菜), are on the one side of a river and want to reach the other side of the river.
- He found that there is a boat parked at the riverbank and can help them cross the river.
- However, Joe can row the boat, and only he can, with only one of his belongings at the same time.
- What is even worse is that the fox eats the goat or the goat eats the cabbage when Joe is not at the scene(不在現場).

Possible Design – Using Class Diagram

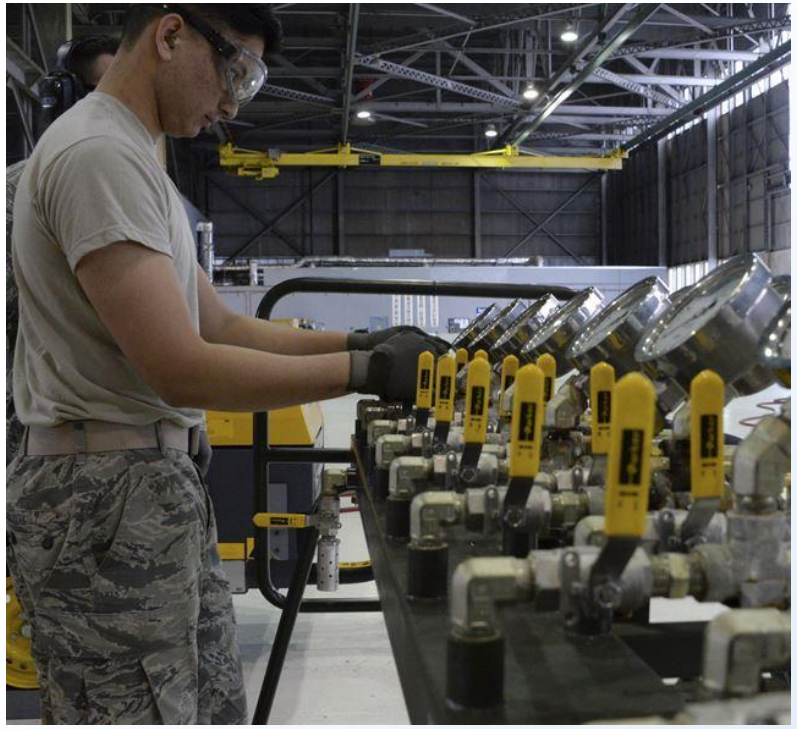


實作(Implementation)

- 透過程式語言將所設計的內容轉化為可執行的軟體。
- 是一般人認為軟體工程師唯一的工作。

測試(Test)

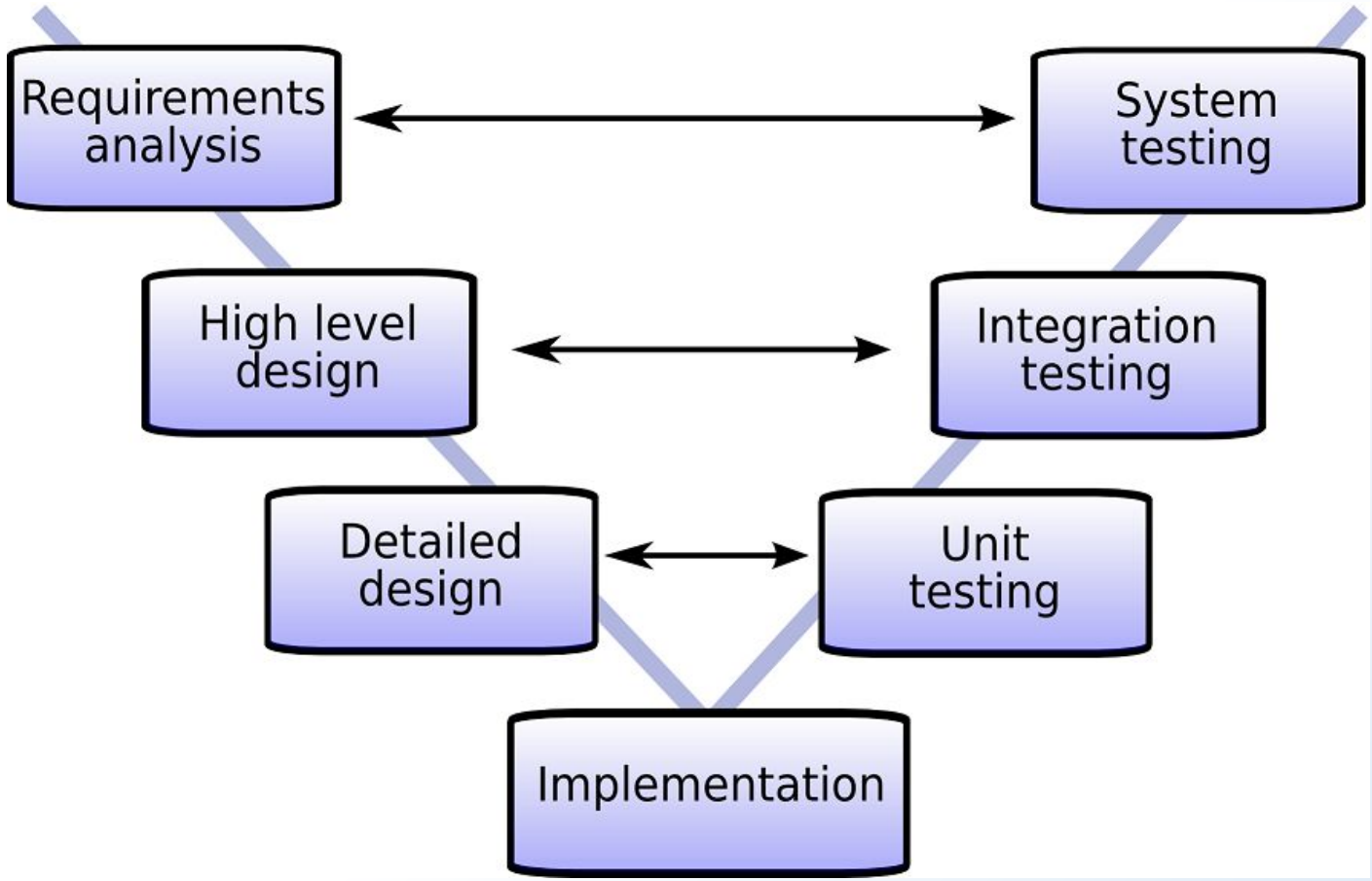
- 測試是對實作活動階段所產出的程式碼模組進行檢測，以檢驗其功能是否正確、效能是否符合要求等。
- 測試案例(Test Case)的設計是測試流程的重點。



測試可以做到

- 發現程式的錯誤(bug)
- 評估程式的品質(quality)
  - 與需求或規格一致(conformance to requirements)
  - 符合使用的目的(fitness for purpose)
  - 功能、效能、安全性、使用者經驗

軟體測試V Model


Requirements analysis
System testing
High level design
Integration testing
Detailed design
Unit testing
Implementation

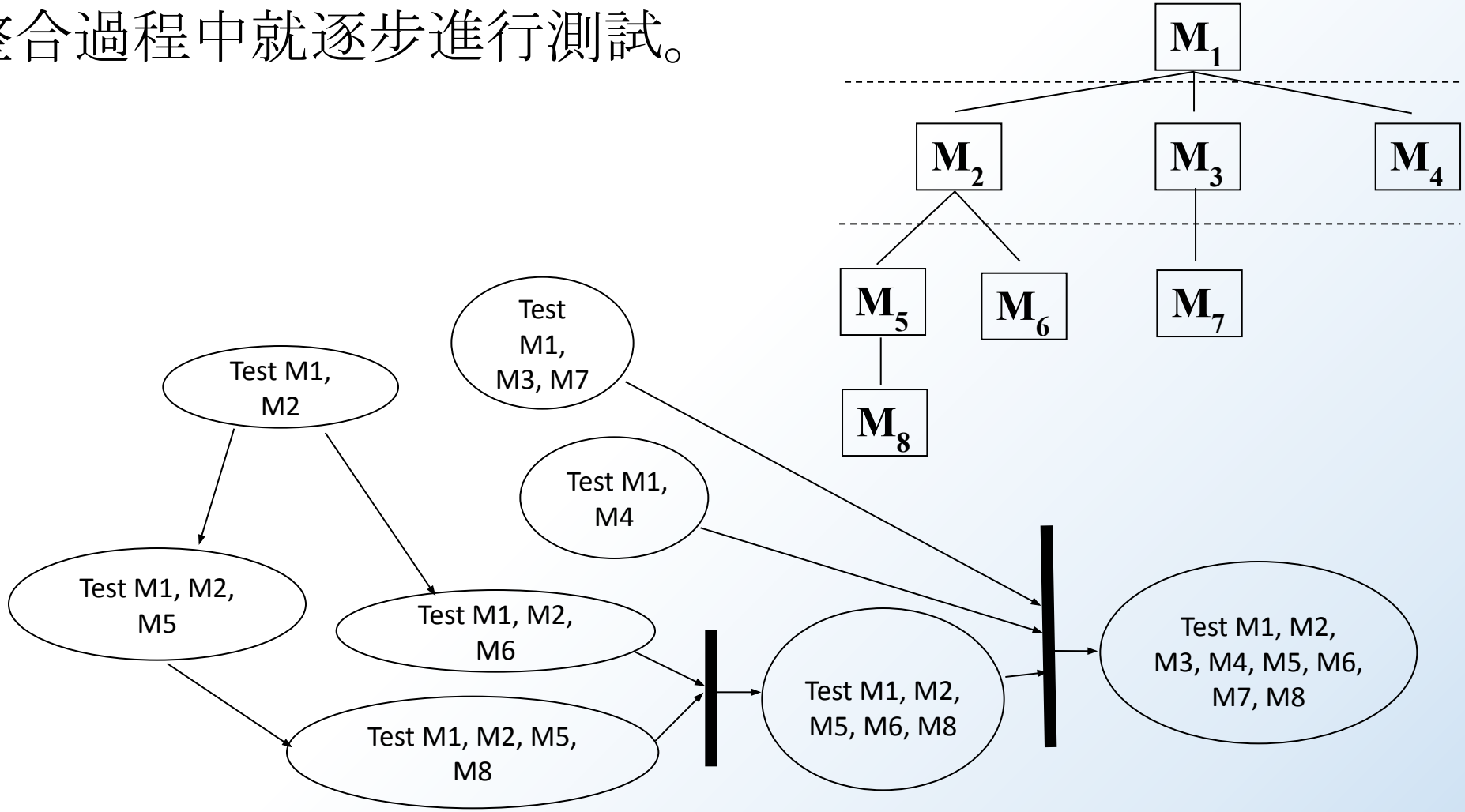
單元測試

- 以程式中最小的邏輯單元為標的，撰寫測試程式，來驗證程式是否正確。
- 忽略單元測試可能造成日後難以除錯的bug。

```
1  @Test
2  public void simpleAdd() {
3      Money m12CHF= new Money(12, "NTD");
4      Money m14CHF= new Money(14, "NTD");
5      Money expected= new Money(26, "NTD");
6      Money result= m12CHF.add(m14CHF);
7      assertTrue(expected.equals(result));
8  }
```

整合測試

- 應在整合過程中就逐步進行測試。



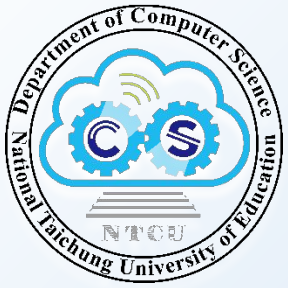
系統測試

- 是對整個系統的測試，將軟體與硬體看作一個整體，檢驗它是否有不符合系統需求之處。
- 包含功能測試、安全測試、壓力測試等。
- 通常由獨立的測試人員測試，而非開發者自行測試。

測試工具

- 你可以運用測試工具來加速測試的進行:
  - Selenium (SideeX)
  - Robot Framework

- Top 10 Automation Testing Tools: https://dzone.com/articles/top-10-automated-software-testing-tools
- Top 10 Most Popular Software Testing Tools
- https://dzone.com/articles/10-popular-software-testing-tools-for-2021



Kanban (看板) & Trello

Kanban Board (看板)

Card

如何有效把現有流程視覺化, 並且改善他!

Trello

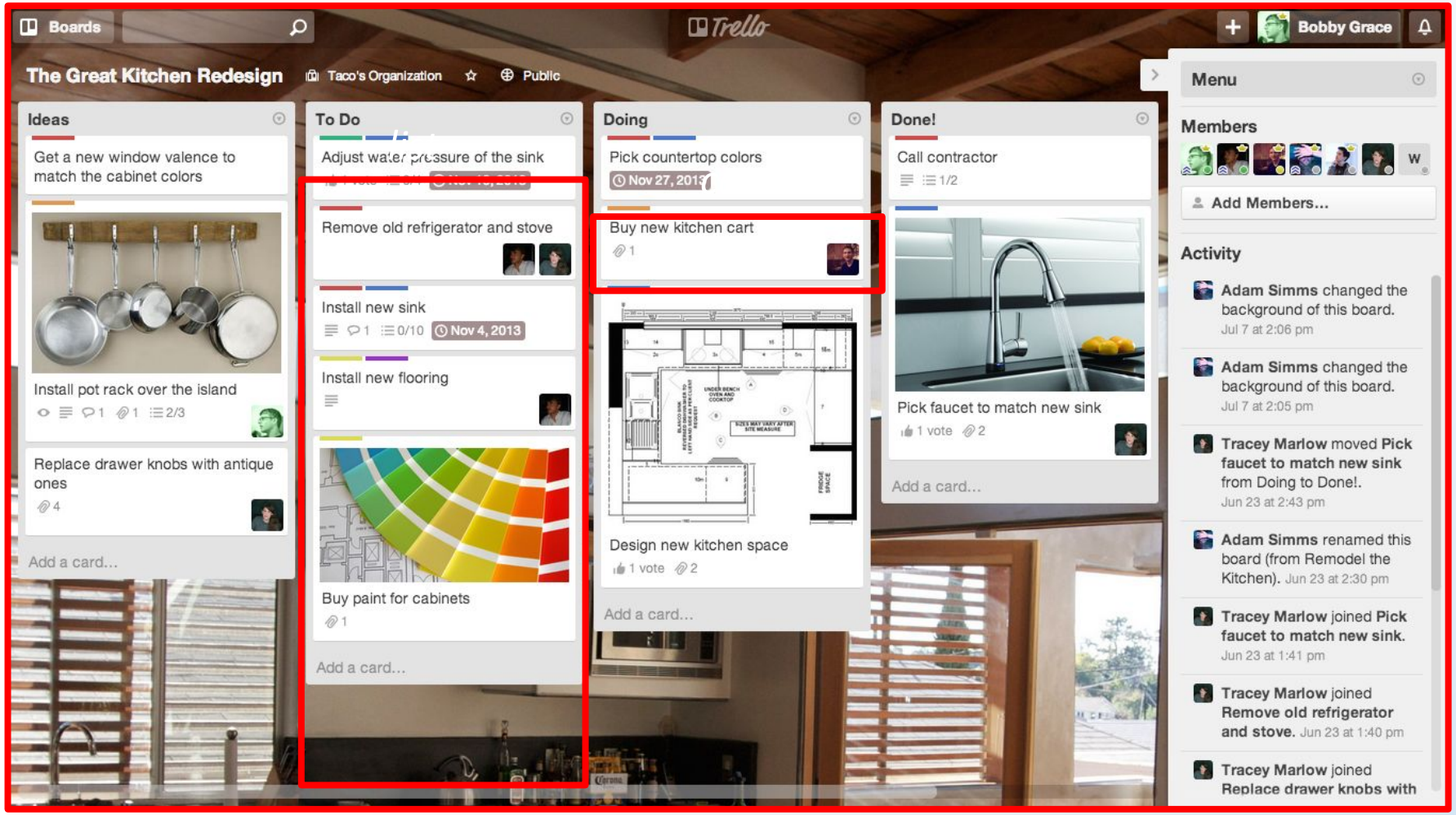
a) 不用錢
b) 適合個人使用
c) 跨平台
d) 簡單易用

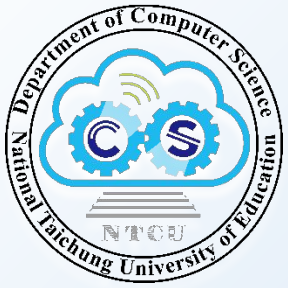
- Trello 實現了看板方法
- 如何使用Trello?
  - https://trello.com/b/3ZTiEXNt/welcome-board

- 我們可運用Trello進行專案管理
  - 分工
  - 追蹤進度

Board, List, and Card

board



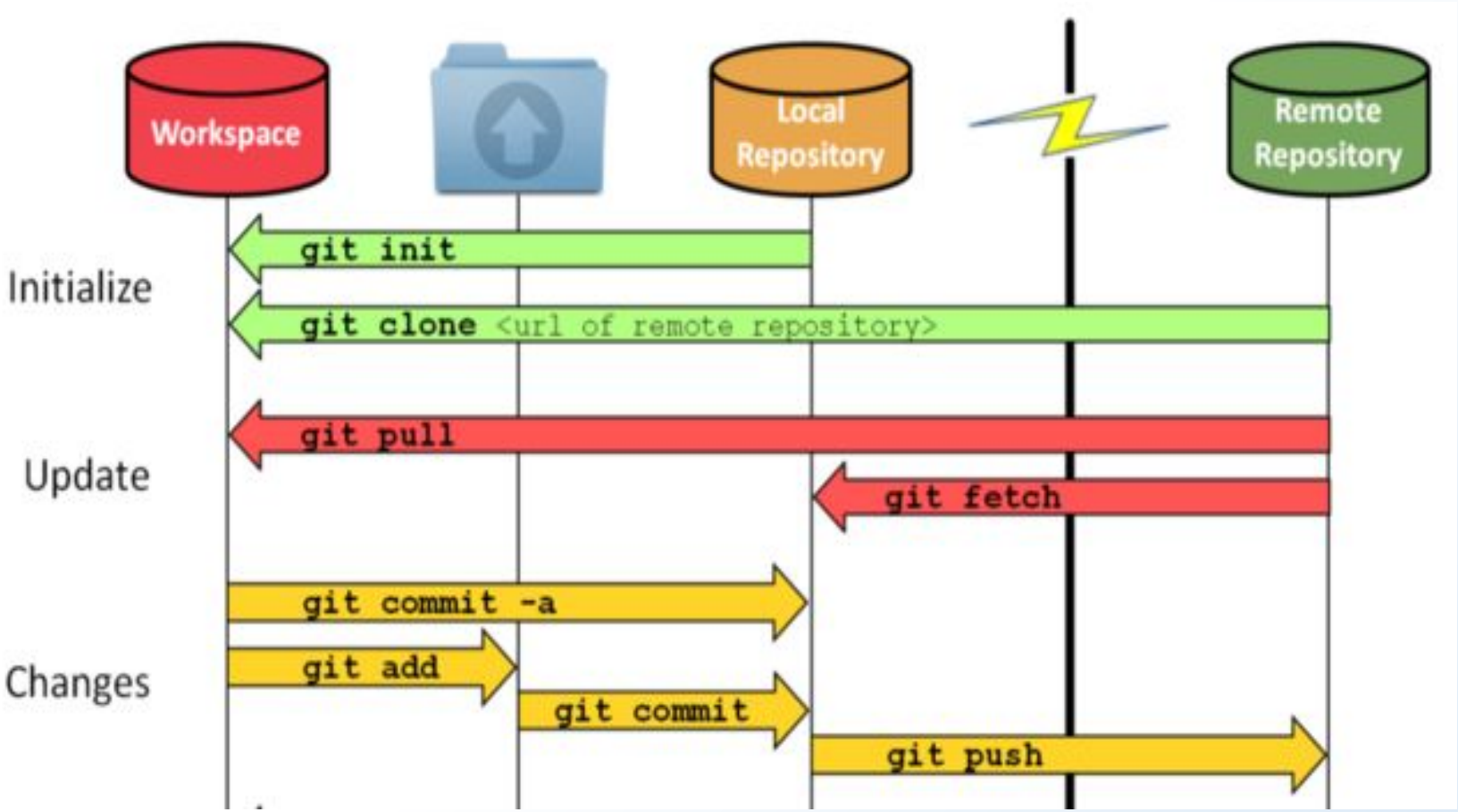

Department of Computer Science
National Taichung University of Education
NTCU

Git

甚麼是Git

- 目前最為主流的**分散式版本控管軟體**
- 每個使用者都有一份完整的儲存庫
- 不需要伺服器端的支援就可以運作
- 提交版本都僅提交到本地的儲存庫
- 因為是在本地提交變更, 不會有任何權限限制
- Github是基於Git的原始碼託管服務(hosting service)

Git運作流程


Workspace
Local Repository
Remote Repository
Initialize
git init
git clone <url of remote repository>
Update
git pull
git fetch
Changes
git commit -a
git add
git commit
git push

Git基本指令與概念

- **git pull**: 遠端儲存庫有更新，要拉至本地端
- **git add**: 選擇欲加入本次版本的變更檔案
- **git commit**: 將本次版本進行提交(至本地端)，建立檔案快照
- **git push**: 從本地儲存庫上傳檔案至遠端

Example 1

Roader: 高速公路與快速道路即時路況社群及語音導引 APP

創新動機1

- 在國道上收聽警廣，卻始終聽不到前方路況
- 廣播資訊囊括全國各式道路，難以過濾所需資訊
- 行進間想撥專線詢問路況或回報，卻影響行車安全

創新動機2

- 出發前想查看路順不順
- 塞車時想了解原因
- 長途旅程時, 不知道該怎麼走最快

Roader系統目標

適地性即時路況

路況回報社群

語音互動與導引

操作概念

路況事件回報

上高架後小寶感受到強勁的陣風，於是輕觸螢幕觸發 Roader 接收語音，小寶說「風很大」，Roader 收到後說「謝謝您的回報！」

路況事件連署

回報之後 Roader 將資訊推播給在高架端上前後 3 公里的小華進行連署，小華的 Roader 詢問：「附近駕駛回報此處風大，是否同意？」小華回答「是」，Roader 在完成連署之後將此狀況擴大散播，讓未上高架的人可收到此消息。

接近路況事件提醒

小寶因為擔心風大發生意外，於是在五股轉接道(32K)開下高架，小寶下轉接道後看到前方壅塞，Roader 馬上告知小寶：「700 公尺前車道發生車禍事故，前方回堵 300 公尺。」

使用案例

*Use case No.	Roader-UC-010	
*Use case name	回報事件	
Summary	在駕駛或副駕駛模式中回報新事件	
*Actors	使用者	
Pre-Conditions	1. 登入狀態 2. 駕駛或副駕駛模式中	
*Description	Actor Actions：	System Responses：
	駕駛模式： 1.點擊螢幕 3.回答事故語句	 2.觸發語音接收 4.確認或者取消 5.謝謝您的回報! 6.將此回報事件納入待連署事件中
	副駕駛模式： 1.點擊回報 ICON 3.選擇欲回報種類 5.填寫資料送出	 2.顯示所有可回報種類 4.根據種類跳出視窗請使用者回報詳細資訊 6.謝謝您的回報!

功能需求

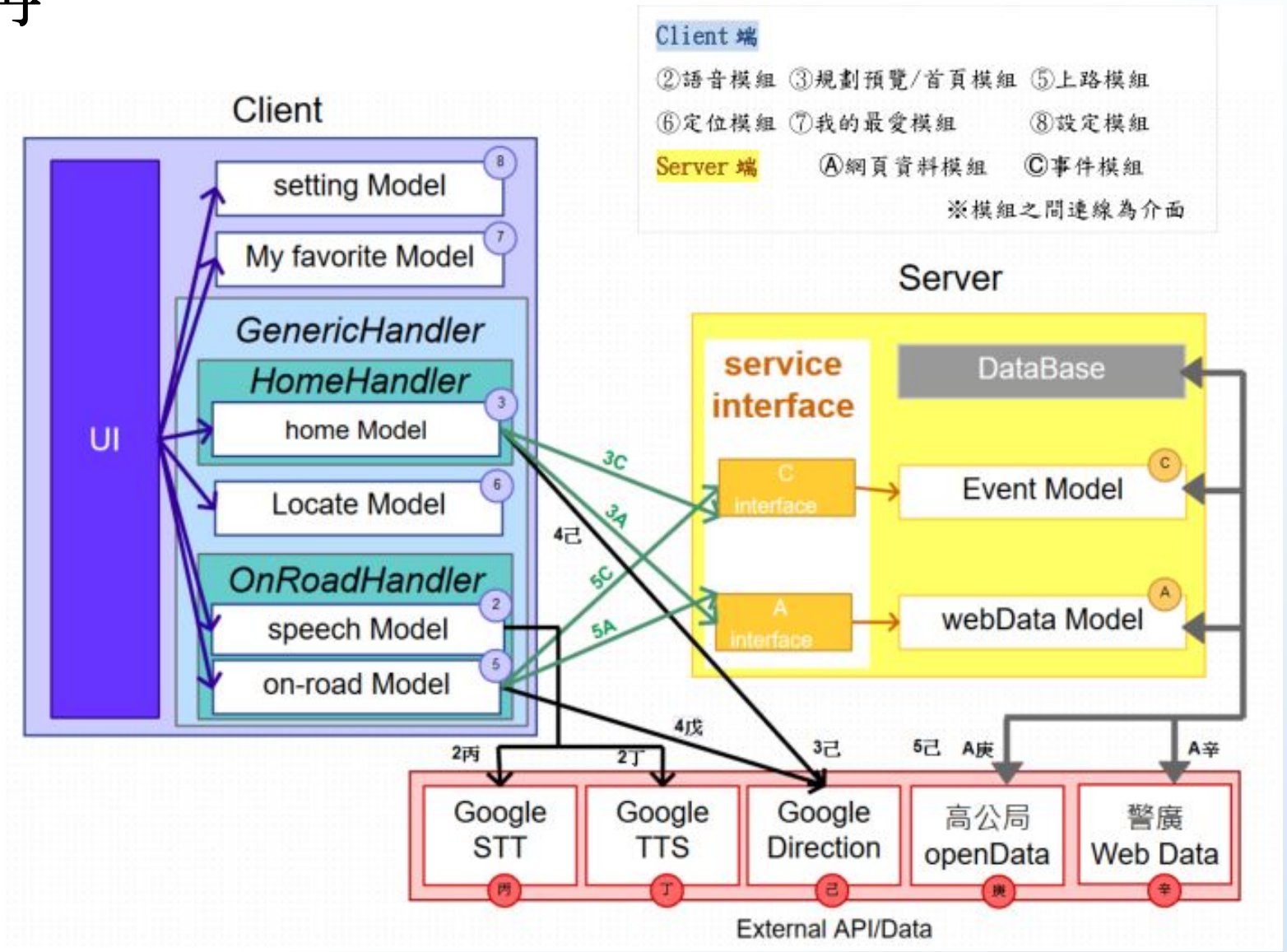
Roader-FR-DM008	語音報告	規劃Roader：語音倒數正在接近的路上事件、告知固定式測速照相位置、車速/最高低速限提醒、接近目的交流道提醒。 自由Roader：語音倒數正在接近的路上事件、告知固定式測速照相位置、車速/最高低速限提醒。
Roader-FR-DM009	事件連署	當使用者的位置進入其他使用者回報路況事件的範圍，即出現連署畫面，讓使用者進行連署，同時也播放語音，告知使用者可以使用語音指令進行連署，一定時間過後會自動離開該畫面。
Roader-FR-DM010	回報事件	使用者可使用語音指令回報事件(事件分級只有一層)。
Roader-FR-AM011	切換至副駕駛模式	整個畫面向右(手指向左滑)，即可切換至副駕駛模式。
Roader-FR-AM012	自動結束	離開終點交流道之後依照設定決定是否自動結束Roader。

非功能需求

Roader-NF-001	路徑計算的結果應在3秒內完成。
Roader-NF-002	Server端應容許1000位使用者同時存取資源。
Roader-NF-003	安全的使用環境： 由於在道路上使用手機釀成意外的事件層出不窮，需達到道路上使用最少化，需擁有駕駛語音模式，在道路上時不讓駕駛使用者進行太繁瑣的動作。
Roader-NF-004	不給予不必要資訊： Roader需將路況資訊經過篩選後，根據使用者的位置給予相對應的路況資訊，讓使用者隨時獲得身旁最新資訊。
Roader-NF-005	動態提供最佳路徑： 在駕駛上路後只要接近系統交流道時，Roader會自動依據當時路況動態判斷是否有更快速到達目的地的路徑。

系統架構

軟體架構


Client 端
②語音模組 ③規劃預覽/首頁模組 ⑤上路模組
⑥定位模組 ⑦我的最愛模組 ⑧設定模組
Server 端 Ⓐ網頁資料模組 Ⓒ事件模組
※模組之間連線為介面
Client
UI
setting Model
My favorite Model
GenericHandler
HomeHandler
home Model
Locate Model
OnRoadHandler
speech Model
on-road Model
Server
service interface
C interface
A interface
DataBase
Event Model
webData Model
3C
3A
4己
5C
5A
4戊
2丙
2丁
3己
5己
A庚
A辛
Google STT
Google TTS
Google Direction
高公局 openData
警廣 Web Data
External API/Data

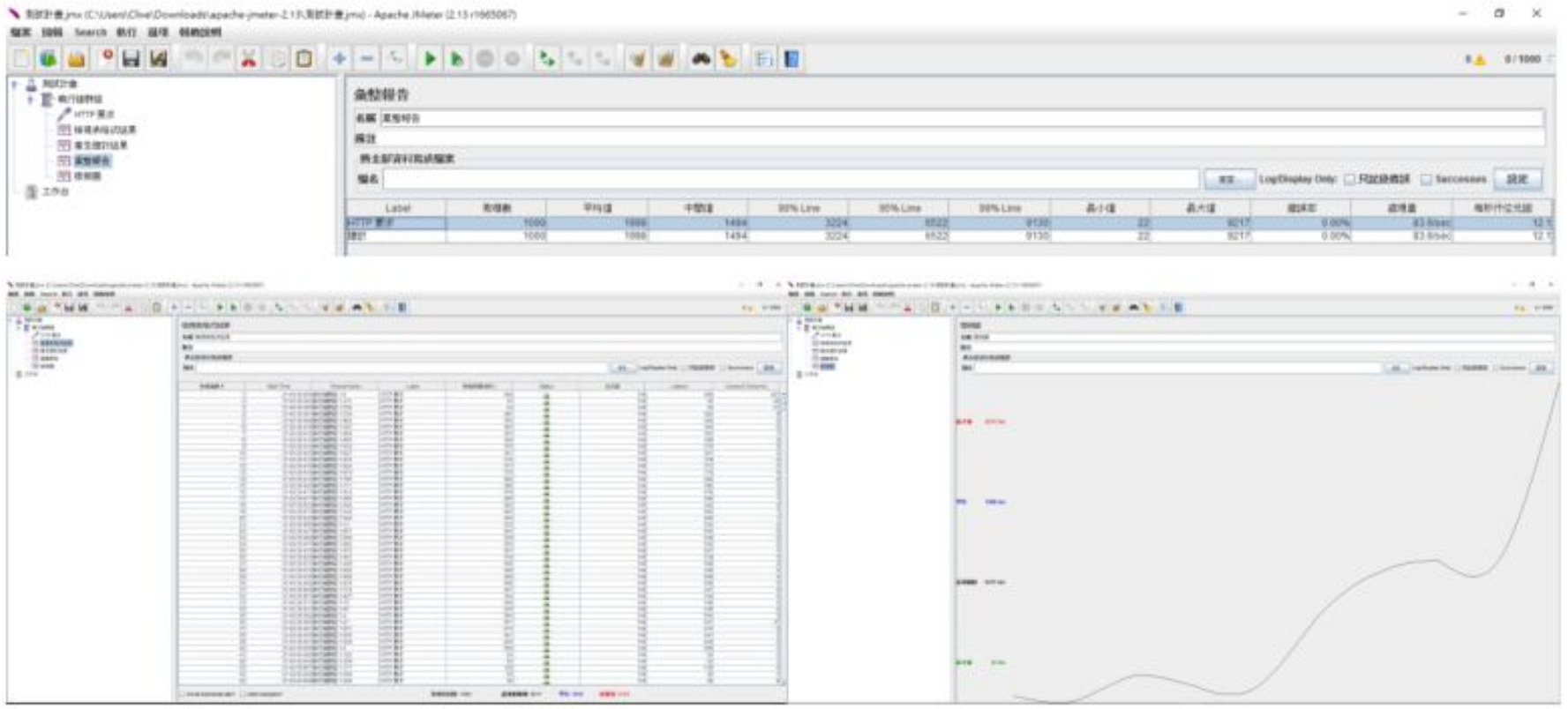
測試案例與測試結果

ID	Roader-TC-003
Name	設定路線提供一或多條參考路徑
Tested target	測試計算一至多條可能路線，並依照路程時間排序
Reference	Roader-FR-PM003
Severity	High
Instructions	1.點擊設定路線 2.選擇基隆端(國道一號-0.6K)為起點 3.選擇旗山端(國道十號 33.8K)為終點
Expected result	顯示一至多條路線並依時間排序給予選擇

Roader-TC-030	Fail	Google Direction 有機率行經一般道路，造成路線判斷錯誤
Roader-TC-031	Pass	
Roader-TC-032	Fail	未能精確倒數
RATE	81.25 %	

效能測試結果

採用 JMeter 對 Server 端做 1 秒同時 1000 次的 HTTP 要求



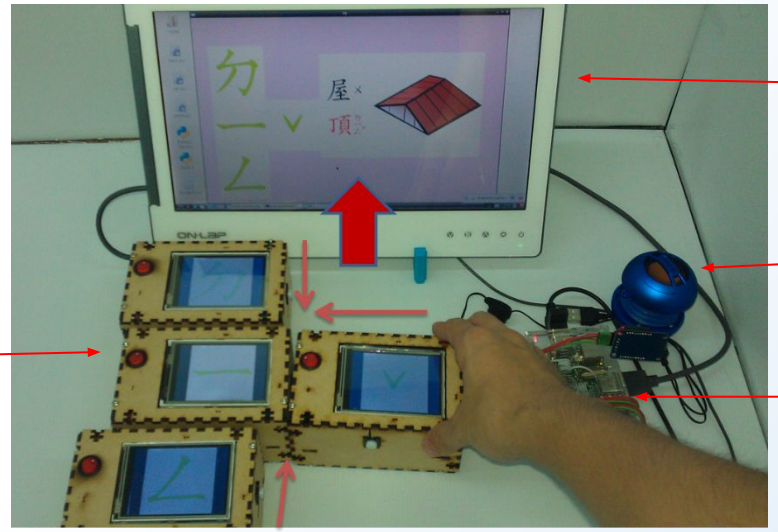
測試結果：失敗率 0.00%，平均回應時間 1998 微秒

Example 2

數位互動式積木系統

需求描述1

- 數位互動式積木系統
  - 系統功能:供兒童使用之注音符號學習系統。

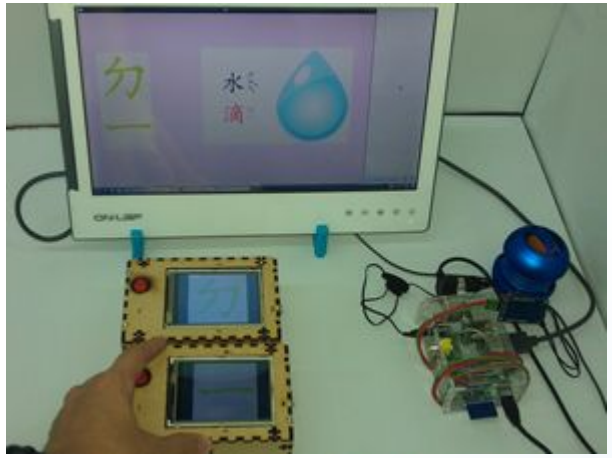


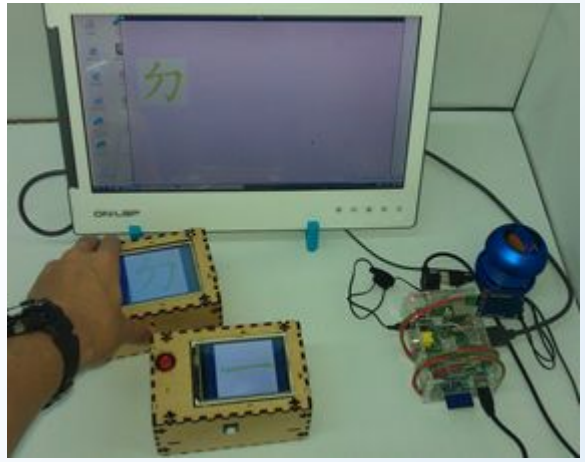
需求描述$_2$

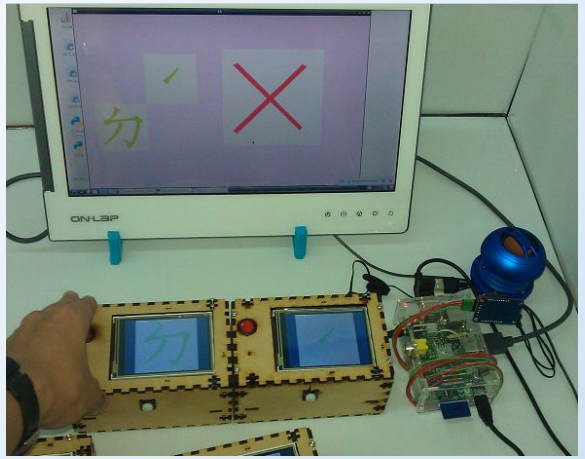
- 操作方式
  - 兒童將數位積木向鄰近數位積木靠攏，觸發感應開關，將積木螢幕顯示注音符號和四方感應開關接觸狀態，傳回主機。
  - 主機接收注音符號與積木開關資訊，運算式否符合注音符號聲母、韻母擺放方式。
  - 主機將積木注音符號顯示於主機螢幕上左方，相關拼音語詞顯示螢幕右方。

需求描述$_3$

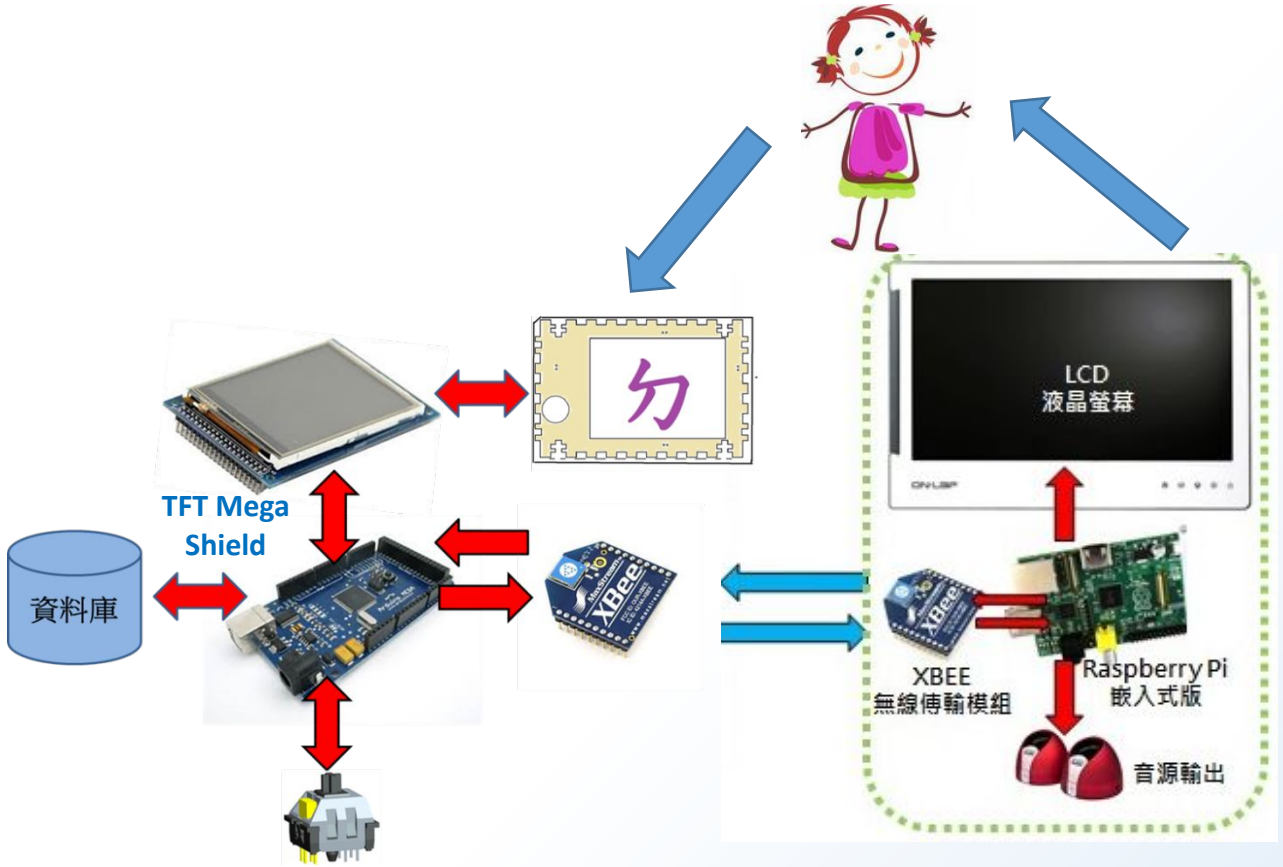
- 操作方式
  - (左)聲母"ㄉ"、韻母”ㄧ”開關觸發
  - (中)挪移積木, 僅聲母"ㄉ"開關觸發
  - (右)錯誤顯示, 聲母"ㄉ"、聲符"ˊ"開關觸發







系統架構

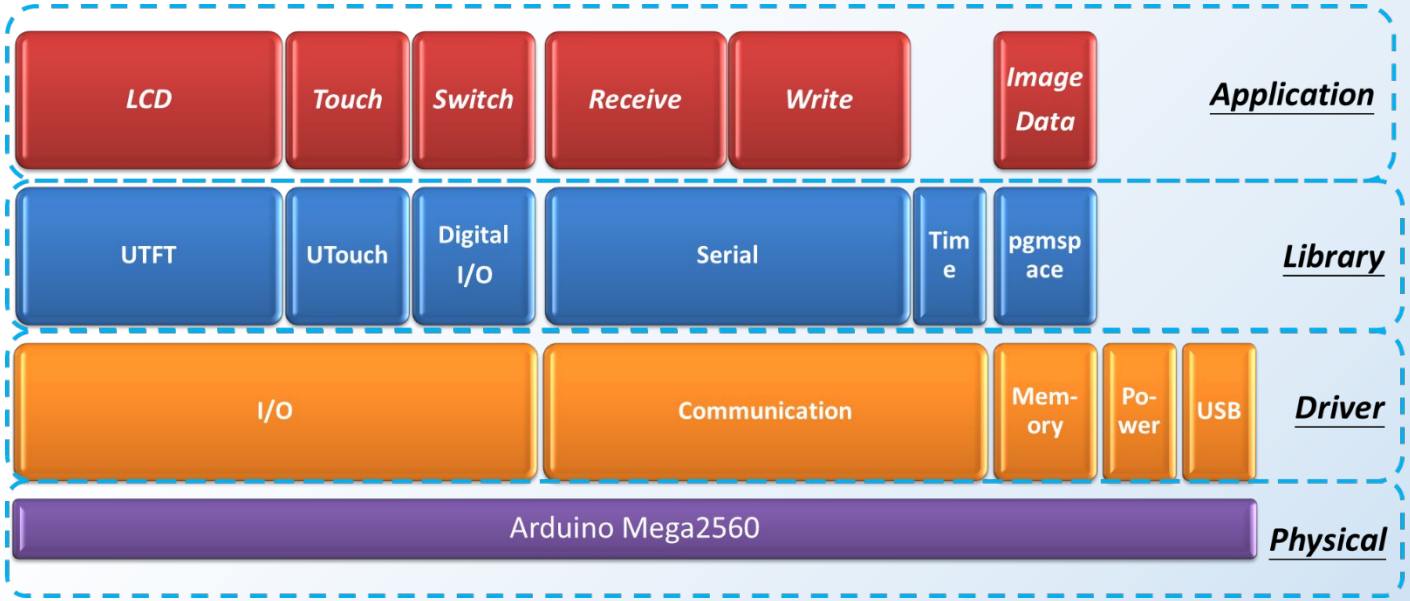


i-Block數位積木

Raspberry Pi多媒體系統

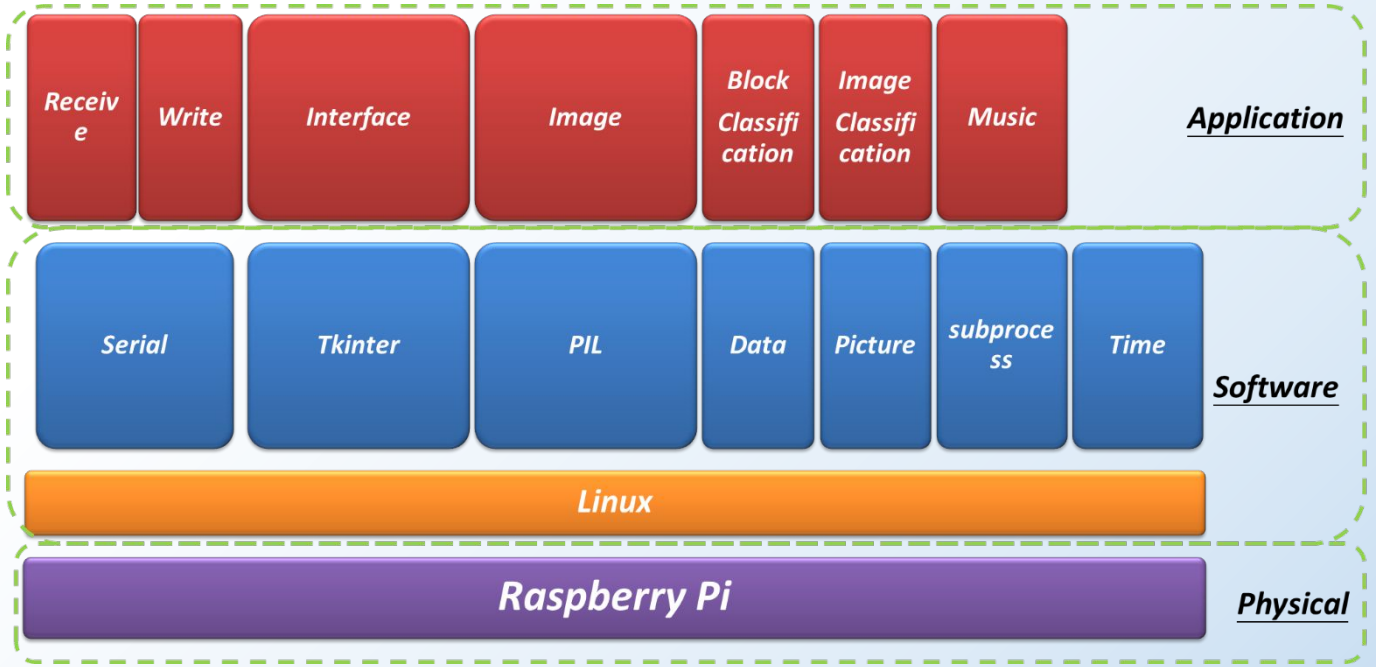
軟體架構設計 - 數位積木

- 分硬體(Physical)、驅動(Driver)、C語言函式庫(Library)、應用(Application)四層。
- 程式撰寫在函式庫與應用層，有積木通訊規則與互動規則。
- 影像傳輸(LCD)、觸控感測(Touch)、感應開關(Switch)、無線資料接收(Receive)、無線資料傳送(Write)、影像資料庫(Image Data)。



軟體架構設計 - Raspberry Pi

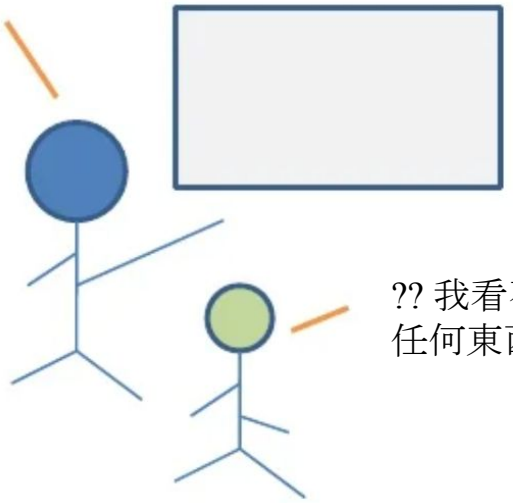
- 分硬體(Physical)、軟體(Software)、應用(Application)三層。
- 程式設計於軟體及應用層上，應用層以Python撰寫。
- 軟體模組分七區塊:無線接收(Receive)、無線傳送(Write)、使用者界面(Interface)、圖像輸出(Image)、積木資料分類(Block classification)、圖像資料分類(Image classification)、音訊輸出(Music)。



測試

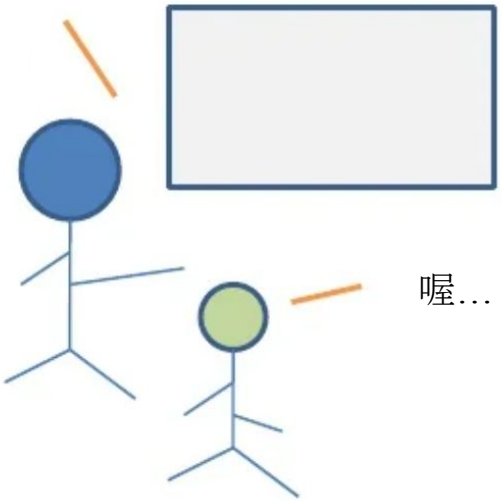
Arduino UNO與觸控螢幕連線測試
觸控螢幕畫線、寫字、按鈕測試
觸控螢幕圖片點陣圖顯示測試
觸控螢幕與Arduino底層函式庫修正測試
觸控螢幕圖片其他格式圖片顯示測試
Arduino UNO 控制紅外線收發模組測試
接觸式按鈕測試
Arduino UNO圖像最大化限制測試
Arduino MEGA圖像最大化限制測試
樹梅派GPIO腳位、RS232通訊系統
樹梅派廣播最大容忍值測試
電子積木測試
電子積木BUG修正測試
互動系統穩定度調整測試
軟硬體系統整合測試

我們的產品應該看起來像這樣子



?? 我看不到任何東西啊

這有什麼問題，我們一會兒就開工。你就這樣開始就對了

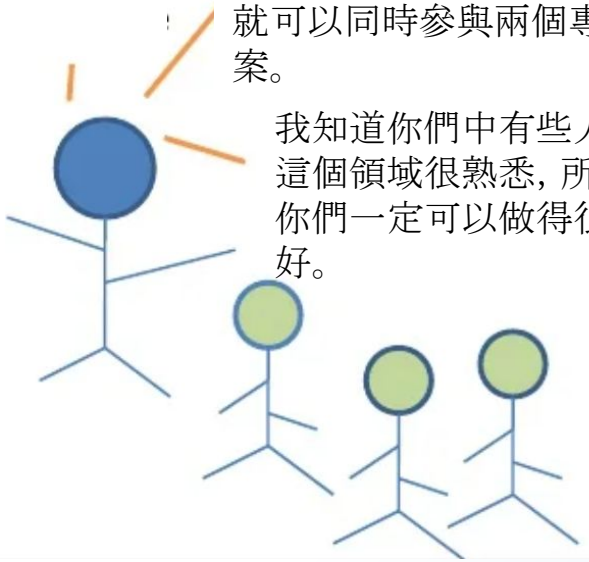


喔…

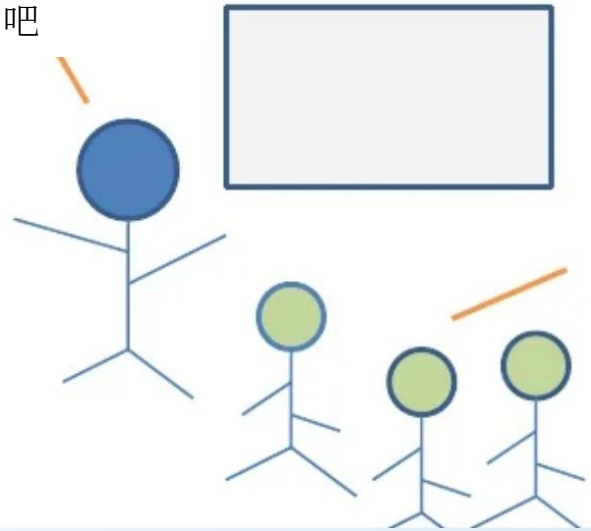
我會再增加資源，我們一定要在結案日前完工。

我會將你們中的一部分人分成兩組，這樣他們就可以同時參與兩個專案。

我知道你們中有些人對這個領域很熟悉，所以你們一定可以做得很好。

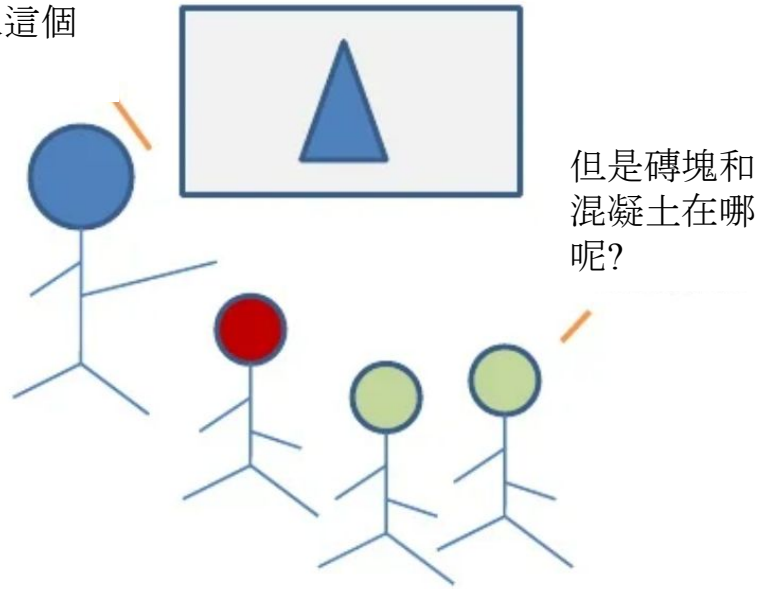


現在你有這麼多人力了，你應該可以更快的完成了吧



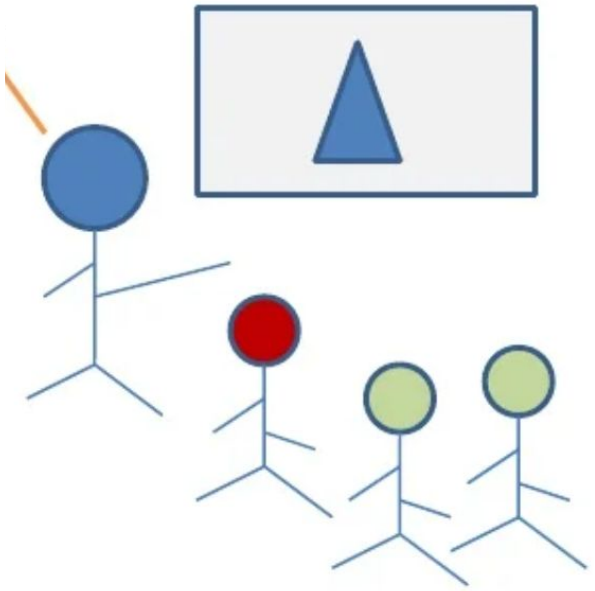
???…

我會持續將詳細的需要加入這個白板上。

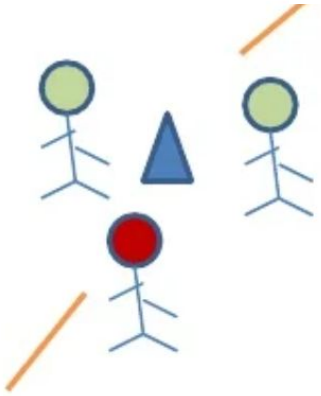


但是磚塊和混凝土在哪呢?

磚塊和混凝土之後將會依序提供給你們

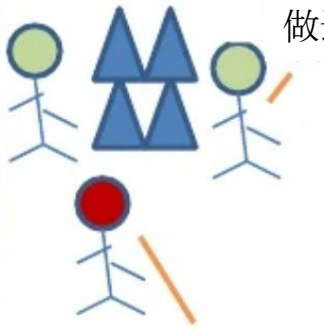


???…



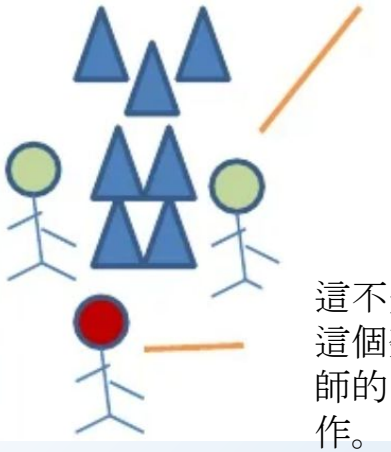
在這邊我們需要用@#@#@這個樣式來做,但是用的時候風險自負啊

我之前從來沒做過

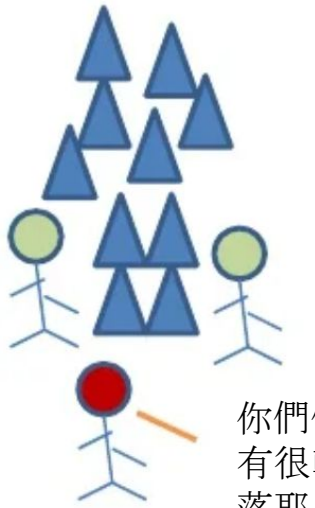


我也沒做過啊,但應該很簡單啦

好像有點不穩, 可以幫忙看看嗎



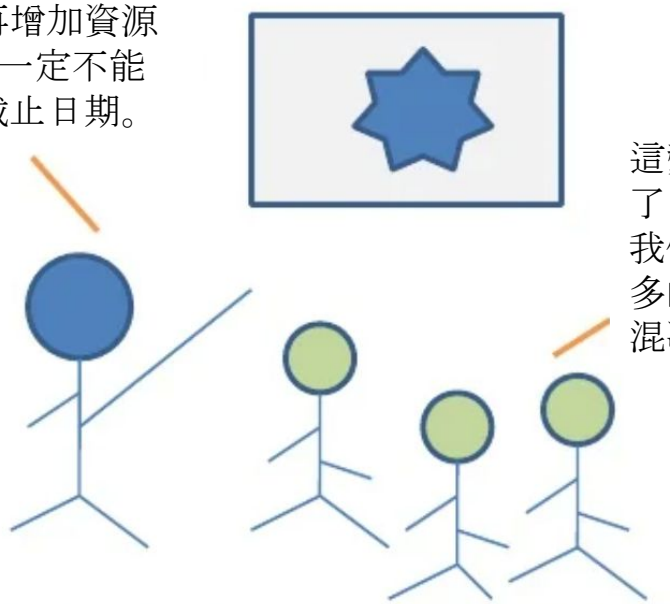
這不是我這個架構師的工作。



你們做的沒有很乾淨俐落耶

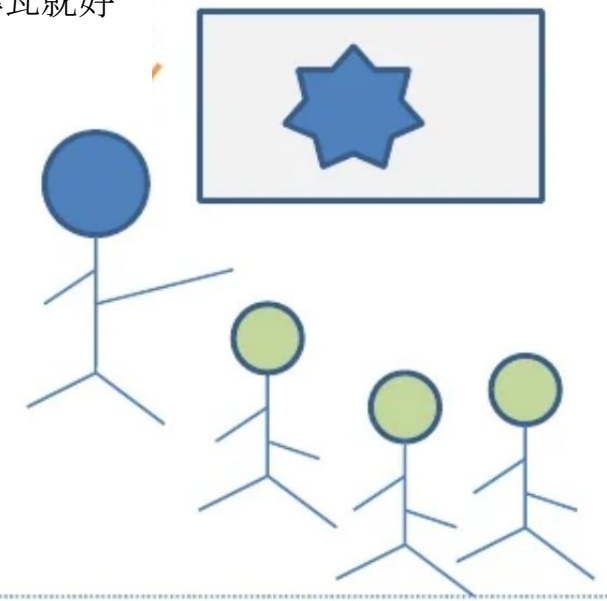
https://www.slideshare.net/amazedsaint/cartoon-software-development-process

目前有些變更，我會再增加資源，我們一定不能超過截止日期。

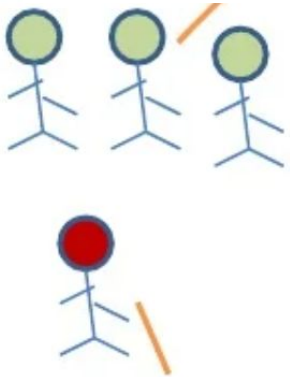


這變動太大了！
我們需要更多的磚塊和混凝土。

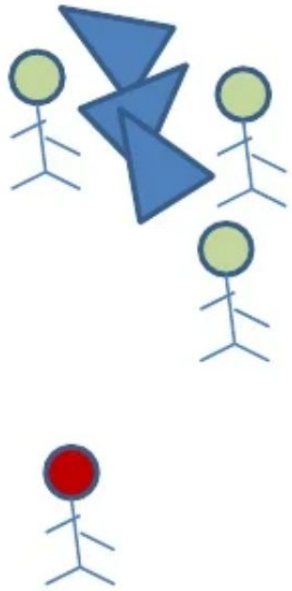
你們這群傻瓜，手上有那些磚瓦就好好運用啊



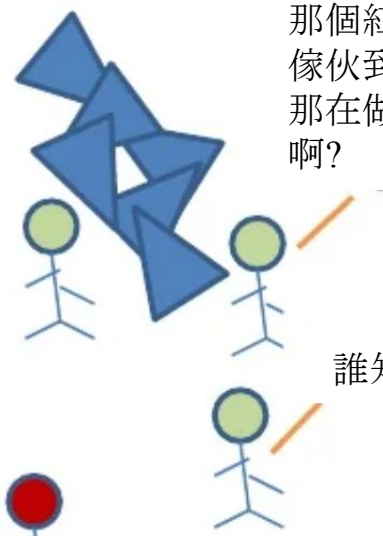
這些變更到底怎麼做啊?



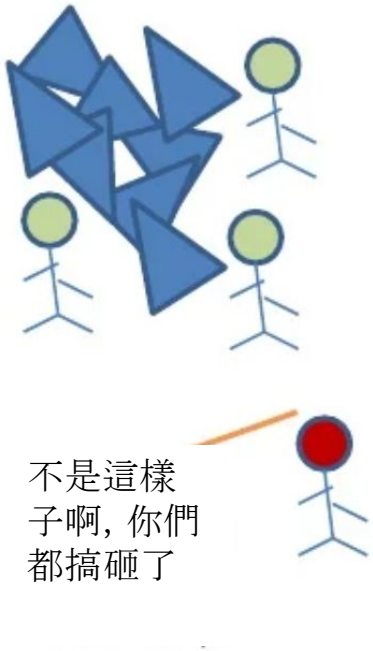
看起來我們可以把每個地方轉一下，變形一下，擠壓一下就行了



那個紅頭的傢伙到底在那在做什麼啊?



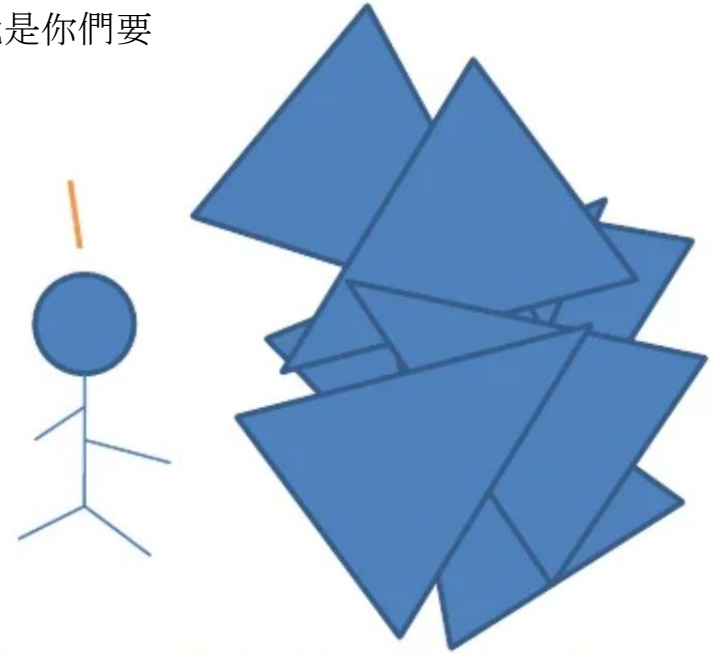
誰知道呢



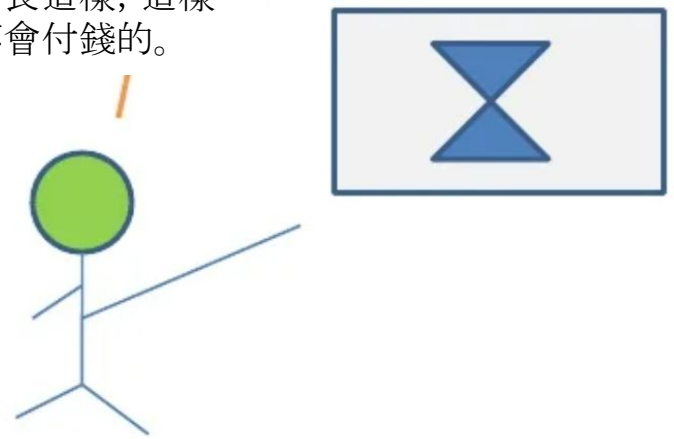
不是這樣子啊，你們都搞砸了

https://www.slideshare.net/amazedsaint/cartoon-software-development-process

我想這個就是你們要的?



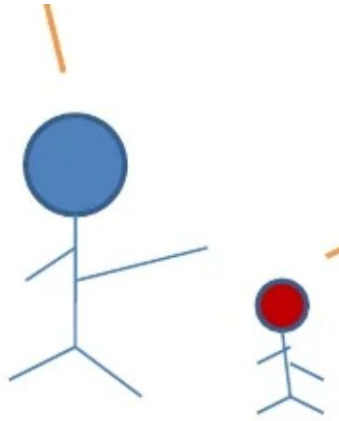
這是甚麼東西啊!完全就不是我要的, 我要的東西長這樣, 這樣我是不會付錢的。



你們搞砸了!!我要把你們開除!

說的對,
開除他們

下次我們要找更好的人力來做這個



是啊, 我們下次一定可以做得更好。我們可使用一下額外的技術樣式。

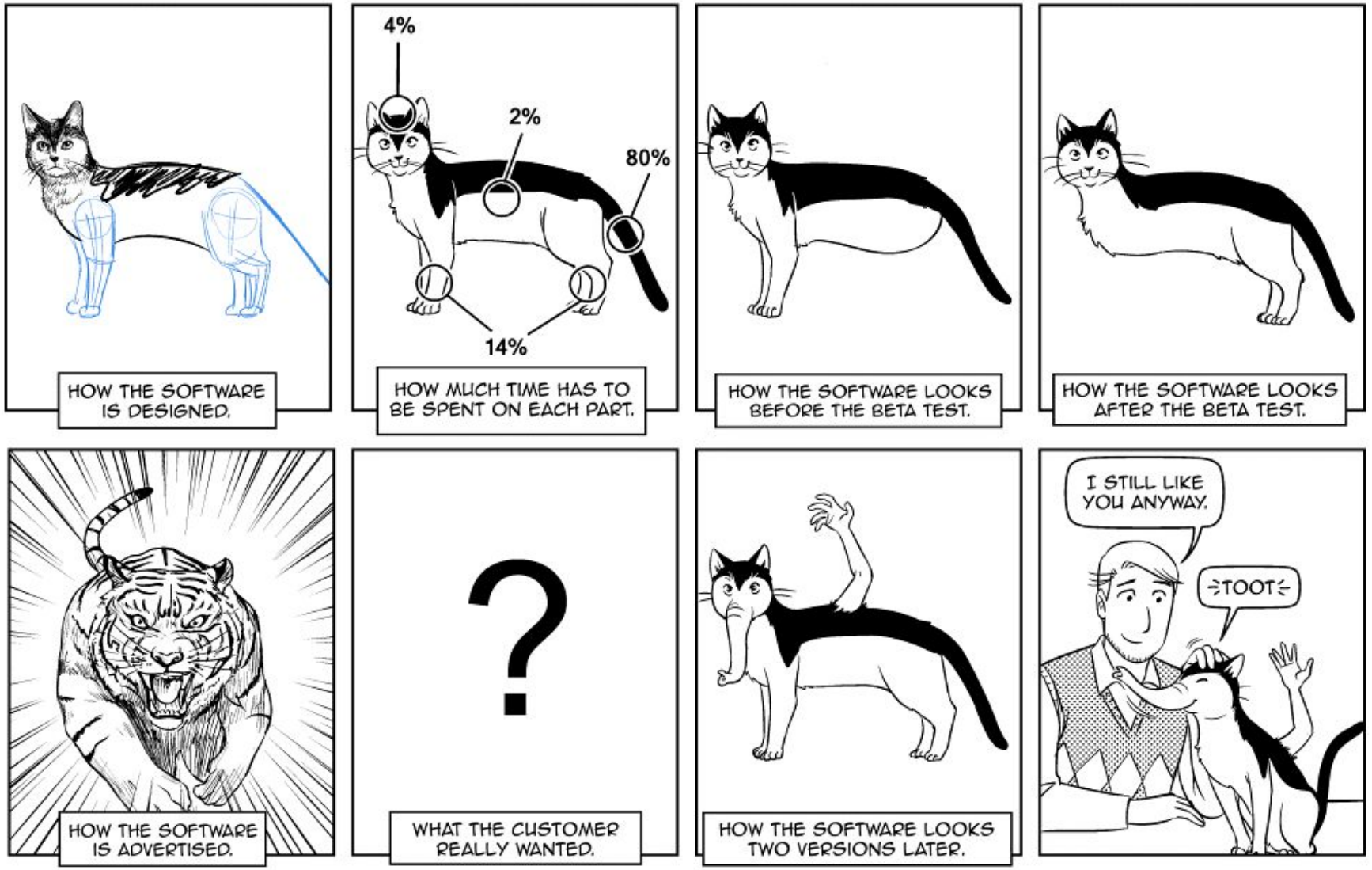
https://www.slideshare.net/amazedsaint/cartoon-software-development-process


這次我們能拿下這個成績了

感謝聆聽

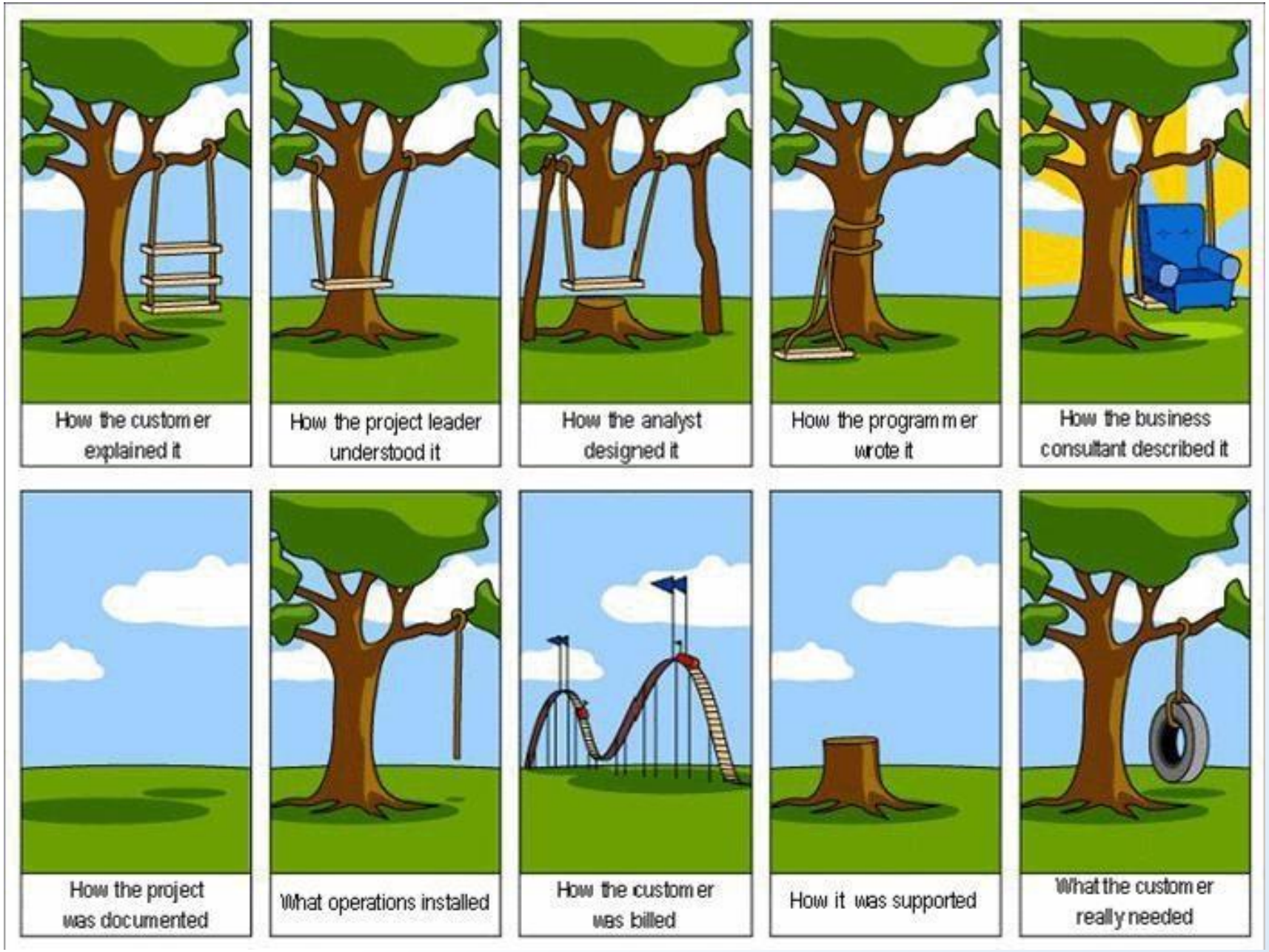
You can find me at:
glenn@mail.ntcu.edu.tw

Richard's guide to software development



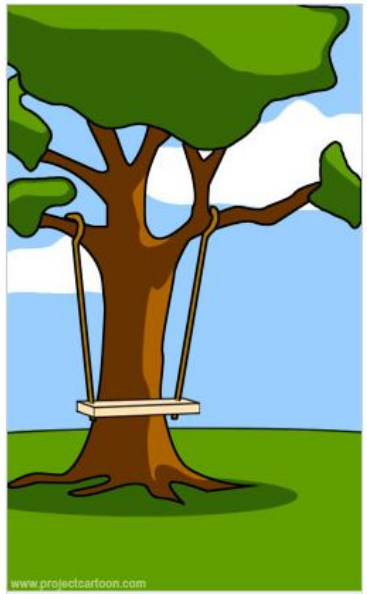
Sandra and Woo by Oliver Knörzer (writer) and Powree (artist) – www.sandraandwoo.com

https://www.quora.com/What-are-the-best-programming-comic-strips



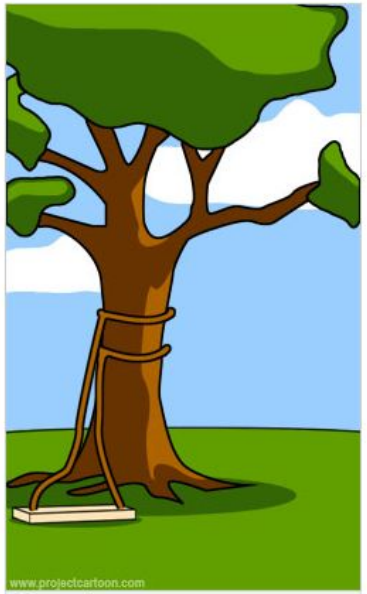
https://medium.com/@thx2001r/the-project-cartoon-root-cause-5e82e404ec8a

How the customer explained it

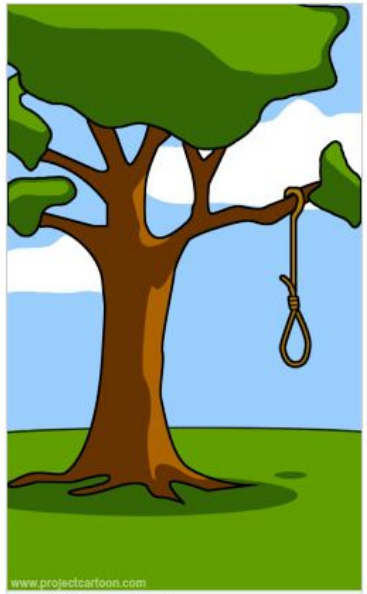


How the project leader understood it

How the analyst designed it

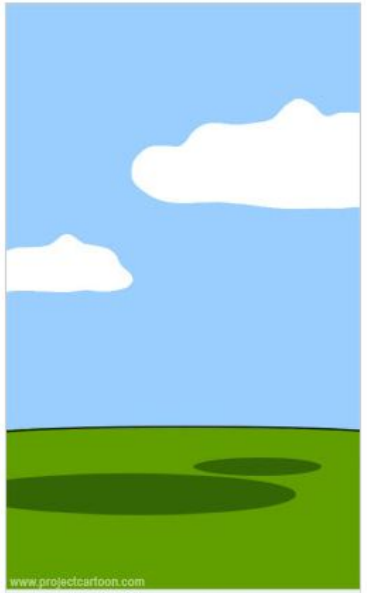


How the programmer wrote it

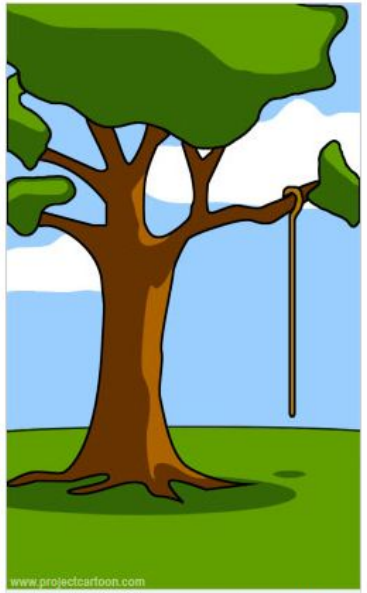


What the beta testers received

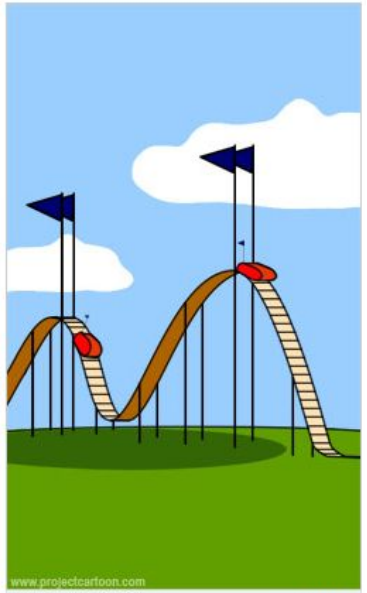
How the business consultant described it



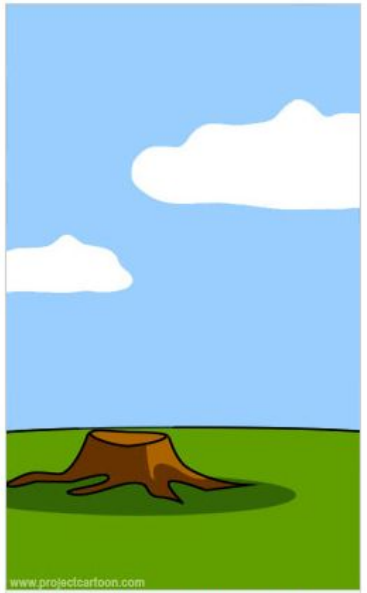
How the project was documented



What operations installed



How the customer was billed

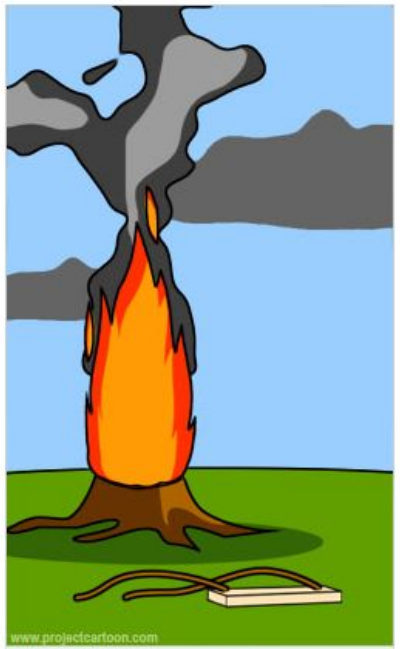


How it was supported

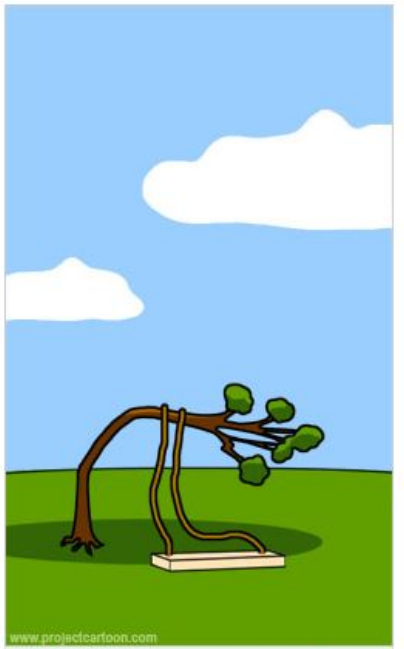
What marketing advertised

What the customer really needed

When it was delivered



What the digg effect can do to your site



The disaster recover plan