程式人

用 20 分鐘搞懂

《系統分析、軟體工程、專案管理與設計模式》

陳鍾誠

2016 年 1 月 20 日

話說、我大學的時候

- 念的是《交通大學資訊科學系》
- 直到畢業為止、都沒有老師開設《軟體工程》和《專案管理》的課程。

我覺得很奇怪

- 為甚麼這麼重要的課程
- 卻沒有安排老師來教呢？

直到我當了資工系的老師

- 我終於瞭解為甚麼了！

原因很簡單

那就是

- 沒有任何老師能教這門課！

你一定會說

- 這些老師也太誇張了，沒辦法教就不教了嗎？

但是、你可曾想過

- 沒辦法教還硬教的情況 . . .

難道

- 那樣會更好嗎？

問題是、為何大學資工系

- 沒有老師可以教軟體工程呢？

這個問題的答案

- 非常的簡單！

請稍微想一分鐘

- 從那些老師到底是怎麼培養出來的 . . .

您就會瞭解了！

還是不懂的話

- 再給個提示、就是那些老師通常都是博士！

是博士又怎樣？

是博士

- 所以通常業界經驗不多！

業界經驗不多又怎樣？

業界經驗不多

- 所以很少有團隊合作的經驗

沒有團隊合作的經驗

- 又會怎樣呢？

沒有團隊合作的經驗

那你還敢讓他們

- 教你軟體工程和專案管理嗎？

有多少大學老師

- 寫過大型專案
- 或目前正在參與軟體專案開發的呢？

所以

- 當初交大資科不開軟體工程和專案管理
- 是對的、是對的、是對的！

但是今天

- 我還是要以一個大學老師的身分，談談《軟體工程、專案管理和系統分析》

因為

- 至少我的經驗比學生多一點點！

而且

- 業界的高手們，都忙著做專案，哪有時間寫文章呢？

話說

- 如果你所寫的程式，從來沒有超過一千行，那是完全不需要《軟體工程》的。

如果你總是一個人寫程式

- 那也根本不需要《專案管理》

如果你的問題

- 一看就知道怎麼寫，那連《系統分析》都可以省了。

這樣的話

- 到底甚麼時候才需要《系統分析》、《專案管理》和《軟體工程》呢？

這個問題的答案很簡單

- 就是把上面的情況都反過來
- 那就是
  - 你不是一個人寫程式
  - 你寫的程式規模不小
  - 你的問題並不是一看就知道怎麼寫了。

在這種情況下

- 你就需要《系統分析》、《專案管理》和《軟體工程》了

因為

- 不分析就不知道該怎麼寫
- 不管理就沒辦法一起寫程式
- 不用軟體工程專案就會失控

舉例而言

- 我大學的時候，寫作業從來不做系統分析

但是有一次

- 資料結構的老師出了一個《運算式微分》的作業。我馬上去寫，結果發現寫不出來。
- 後來仔細靜下心來，把《微分公式》和《鏈鎖規則》寫下來，仔細想想如何用遞迴模擬連鎖規則完成這個作業，後來就順利做完了

這就是

- 一個簡單的《系統分析》案例！

接著你可能會問

- 那《專案管理》又是甚麼呢？

關於這個問題

- 只要做過《專題》的同學都應該有所體悟

我大學的時候

- 專題企圖三個人用 X-Window 合寫一個資料庫管理系統
- 於是就一個人寫視窗、一個人寫資料庫引擎、一個人寫資料庫前端

換句話說

- 就是採用《一個蘿蔔一個坑》的戰略。

結果如何呢？

- 當然可想而知！
- 最後整合的時候，整合不起來
- 好不容易整合起來了，卻很難用，一點都不像資料庫管理系統

這就是因為

- 沒有《管理好專案》的緣故！

接下來

- 就剩下《軟體工程》還沒講了！

到底甚麼是軟體工程呢？

如果你去看教科書

- 很可能會覺得講得好像有道理
- 但就是沒辦法拿來用！

像是那個 CMMI

- 規定了一堆東西，就好像 ISO-9000 一樣。

問題是若你照著做

- 就只會寫出一大堆應付上級的文件！
- 難道這就叫做軟體工程嗎？

靠、當然不是囉！

就是因為 CMMI 沒辦法用

- 所以後來甚麼 SCRUM 和 XP 就跑出來了。
- 至少這兩個還有告訴你該怎麼用！

這樣的話

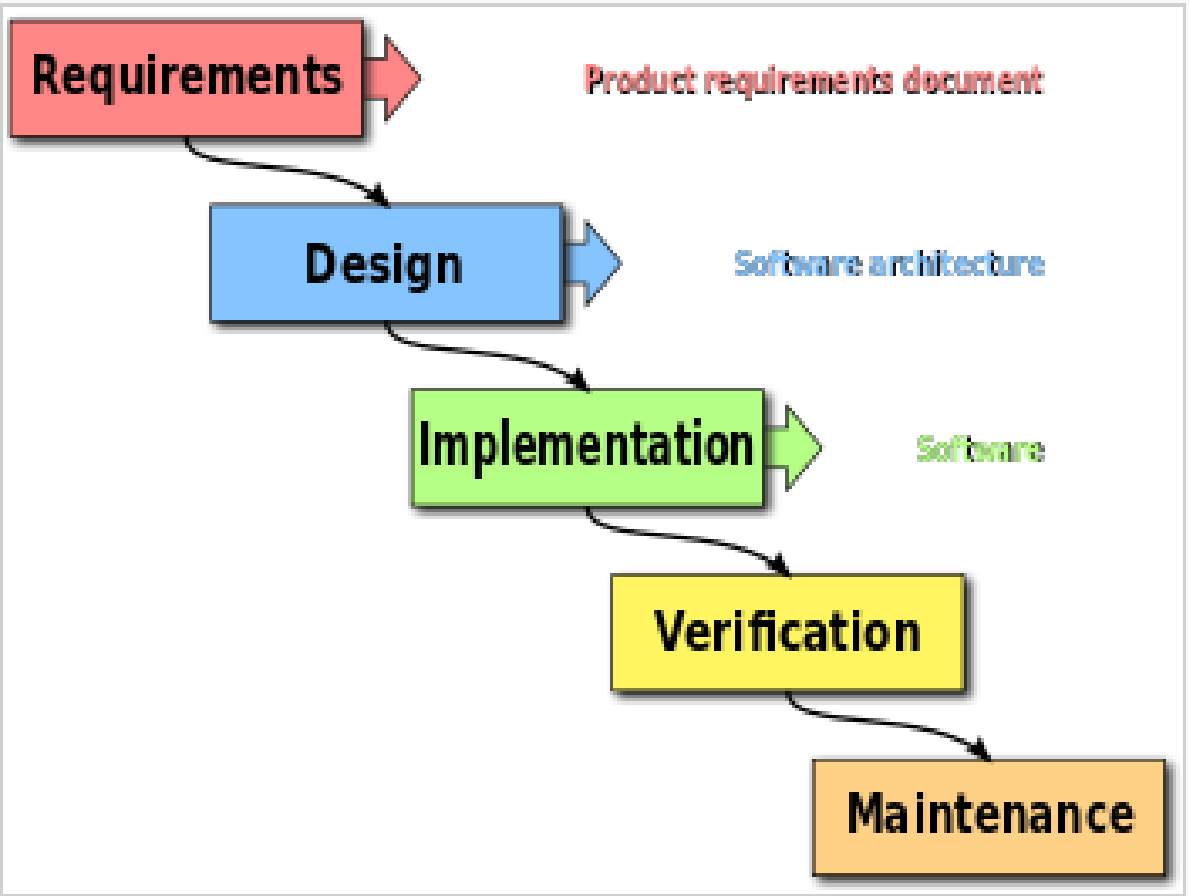
- CMMI 的設計者是腦袋壞掉嗎？

我想也不是如此！

如果你唸過軟體工程

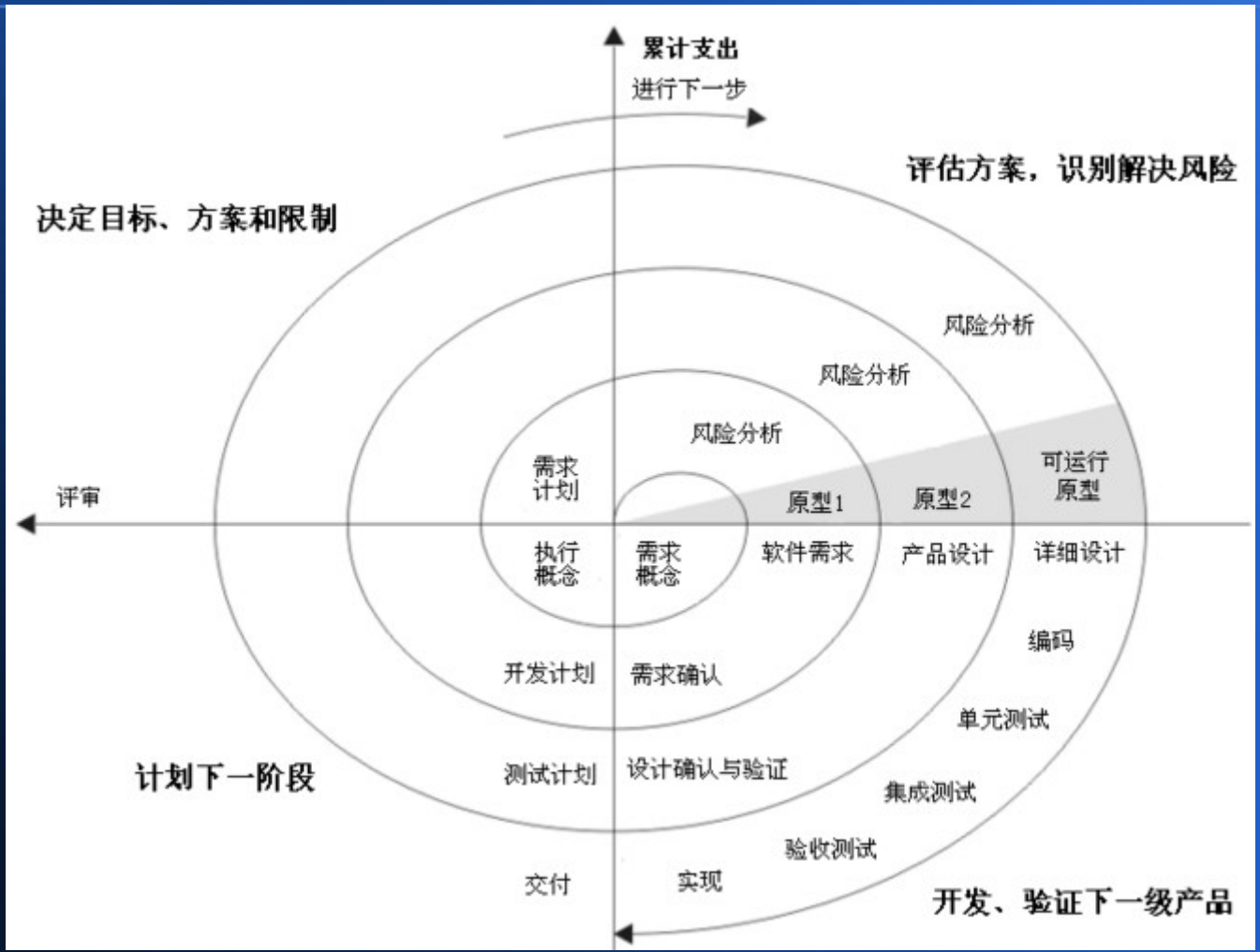
- 應該對《瀑布模式》和《螺旋模式》之類的一堆模式有印象。

瀑布模式



The unmodified "waterfall model". Progress flows from the top to the bottom, like a cascading waterfall.

螺旋模式


累计支出
进行下一步
评估方案，识别解决风险
决定目标、方案和限制
风险分析
风险分析
风险分析
需求
计划
评审
原型1
原型2
可运行
原型
执行
概念
需求
概念
软件需求
产品设计
详细设计
编码
开发计划
需求确认
单元测试
计划下一阶段
测试计划
设计确认与验证
集成测试
验收测试
交付
实现
开发、验证下一级产品

請您想想

- 甚麼樣的專案適合用瀑布模式，什麼樣的專案適合用螺旋模式呢？

其實答案很簡單

- 如果您是專案負責人，必須決定採用哪一種模式。
- 那麼您該怎麼決定呢？

如果、那種專案

- 您已經做過 n 遍，現在要做第 n+1 遍的話
- 此時、您的目標很確定，經驗很夠，所以
  - 需求 => 分析 => 設計 => 驗證 => 上線 => 維護
  - 完全可以一氣呵成，不需要回頭！

這時你應該採用

- 《瀑布模式+CMMI》的方法
- 也就是正規軍的作戰方式！

如果你的專案

- 是第一次，而且成員都沒有經驗。
- 或者全世界都沒有案例可循，

  是創新型專案

  - 哪麼採用《瀑布模式》你就死定了！

為甚麼呢？

答案很簡單

這就好像

- 你第一次爬山的時候，沒有地圖、沒有嚮導、沒有去過、也沒有去過那座山勘察過，就決定要攀登聖母峰，而且絕不回頭，只能勇往直前，一路向上一樣。

必死無疑！

為了

- 不要帶著整個團隊一起去死！

您必須小心謹慎、步步為營

- 先從簡單的雛型開始，探查客戶的需求，然後在需求愈來愈明確，技術愈來愈成熟之後，逐漸發展出您想要的軟體。

而這種做法

- 就是所謂的《螺旋模式》了！

最後還有一個問題

- 甚麼是《設計模式》呢？

所謂的設計模式

- 就是其他人把《軟體開發與專案經驗》寫下來，發現某些《程式模式》對軟體開發很有幫助的時候
- 這時你直接去拿他們現成的《武功心法》來用，不用重新摸索，就是套用《設計模式》了。

好了

- 這就是我所理解的《軟體工程》、《系統分析》、《專案管理》與《設計模式》了

但是

- 知道這些就夠了嗎？

喔！

- 當然不是

你知道

- 系統該怎麼分析嗎？

雖然

- 不管是結構化分析、或者是物件導向分析，只要能讓你抓到需求，設計好系統，就是好分析。
- 但是、學習正規的系統分析，還是有些幫助的。

系統分析前

- 你必須先抓到使用者的需求
- 這段稱為《需求分析》

但是

- 要怎麼抓到使用者需求呢？

方法有很多

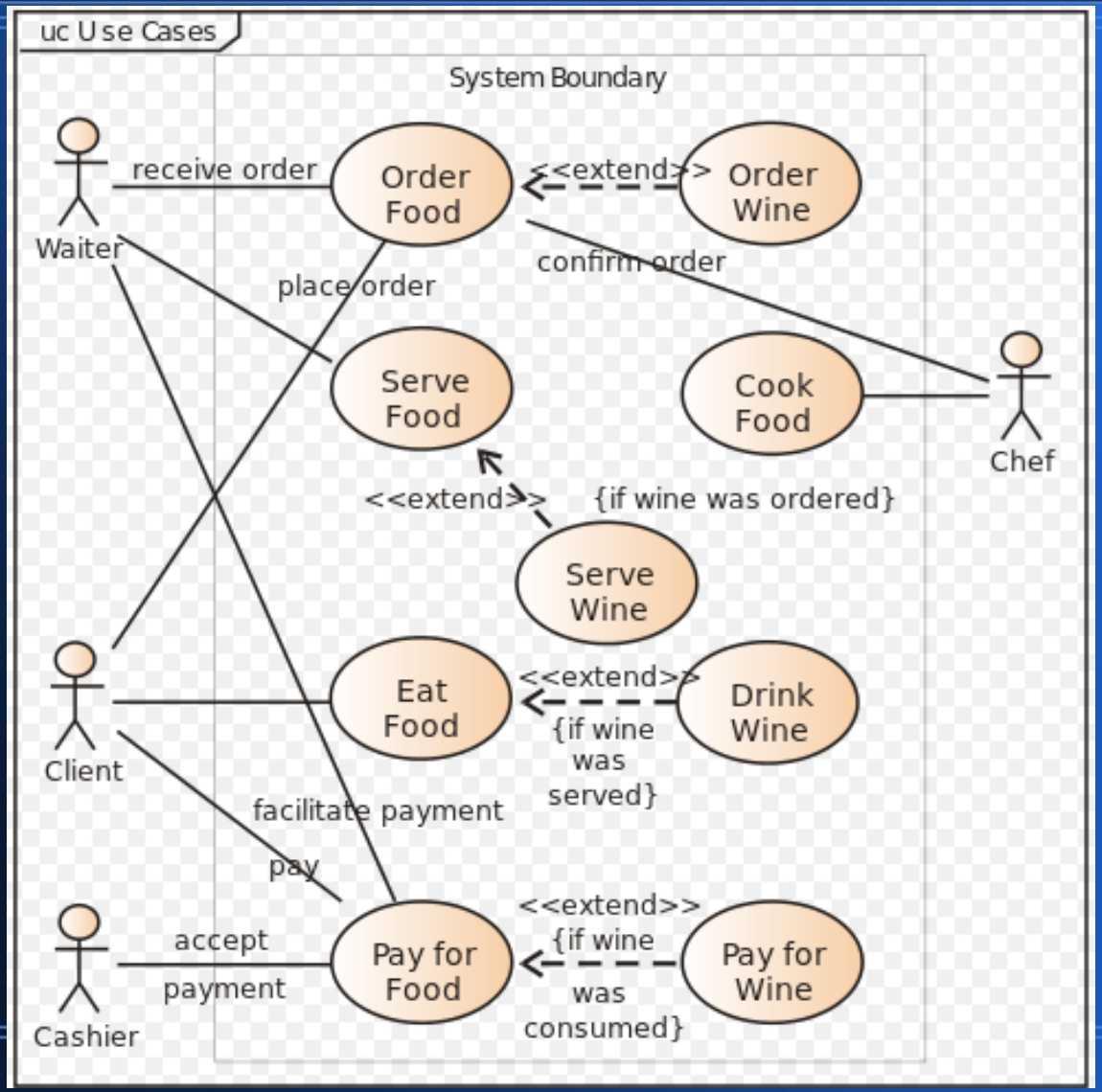
- 像是從《使用個案》下手
- 或者從《雛型介面》下手

都是可行的

從使用個案下手

- 就是假裝你是使用者
- 那每一步會做甚麼事情
- 而系統每一步要顯示甚麼畫面，執行甚麼動作，回應甚麼訊息等等
- 把這些互動用《使用個案圖》畫出來。

使用個案圖


uc Use Cases
System Boundary
Waiter
receive order
Order Food
<<extend>>
Order Wine
confirm order
place order
Serve Food
Cook Food
Chef
<<extend>>
{if wine was ordered}
Serve Wine
Client
Eat Food
<<extend>>
Drink Wine
{if wine was served}
facilitate payment
pay
Cashier
accept payment
Pay for Food
<<extend>>
{if wine was consumed}
Pay for Wine

當然

- 除了畫《使用個案圖》之外，如果還能將《使用者介面》的雛型也做出來，就能更進一步的確認是否符合使用者的需求。
- 這就開始有點像是《雛型法》了。

問題是

- 要設計使用者介面，會不會花太久時間呢？這樣值得嗎？
- 其實、你不需要設計會動的使用者介面，只要把樣子做出來和使用者溝通就好了。

所以分析用的使用者介面

- 你可以用視覺化界面拉一拉
  - 例如 Visual Studio
- 也可以用 PowerPoint 做個樣子
- 或者用專業的快速雛型工具來設計
  - 例如 Axure RP 原型工具

等到需求分析完成

- 就要進一步分析那些看不到的部分
- 需要那些物件、模組、API、裝置才能完成系統的要求

這時候就進入了

- 系統分析階段！

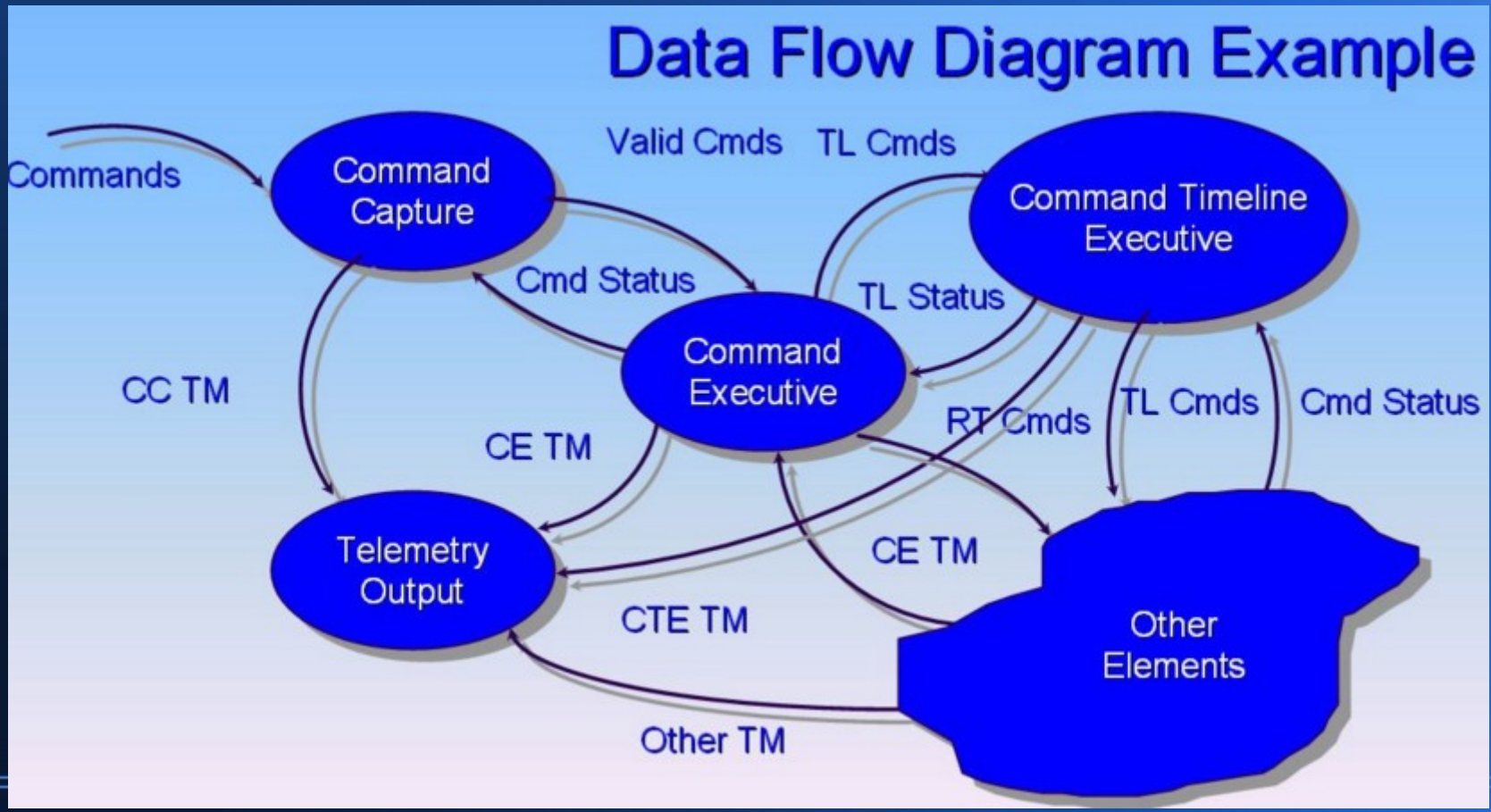
在系統分析階段

- 您可以用結構化分析的
  - 資料流：《資料流程圖》
  - 資料庫：《實體關係圖》

  等工具來分析

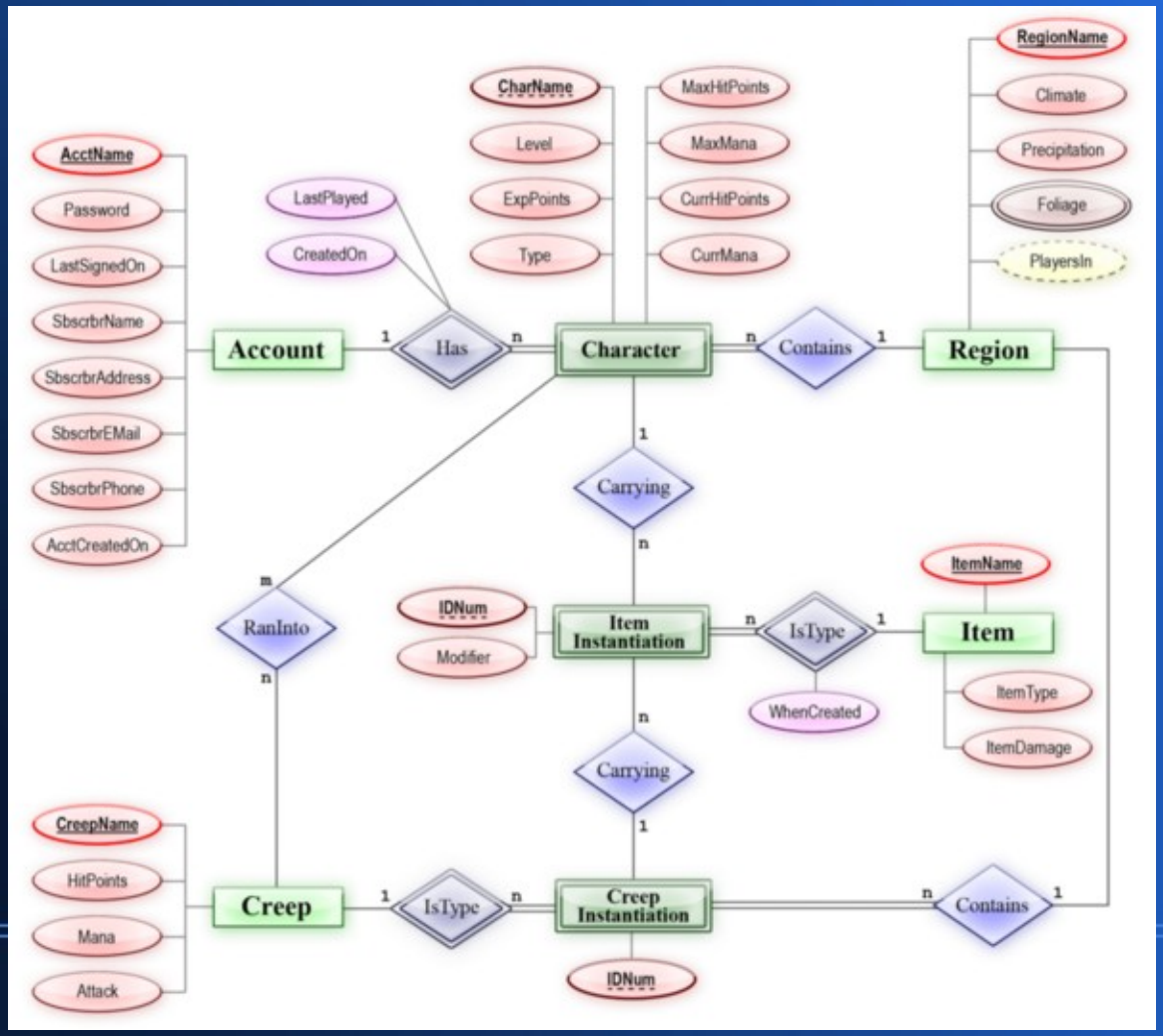
資料流程圖表達的是

- 各模組的輸入輸出與資料流向



實體關係圖則是用來表達

- 資料庫上各個表格的欄位連結關係



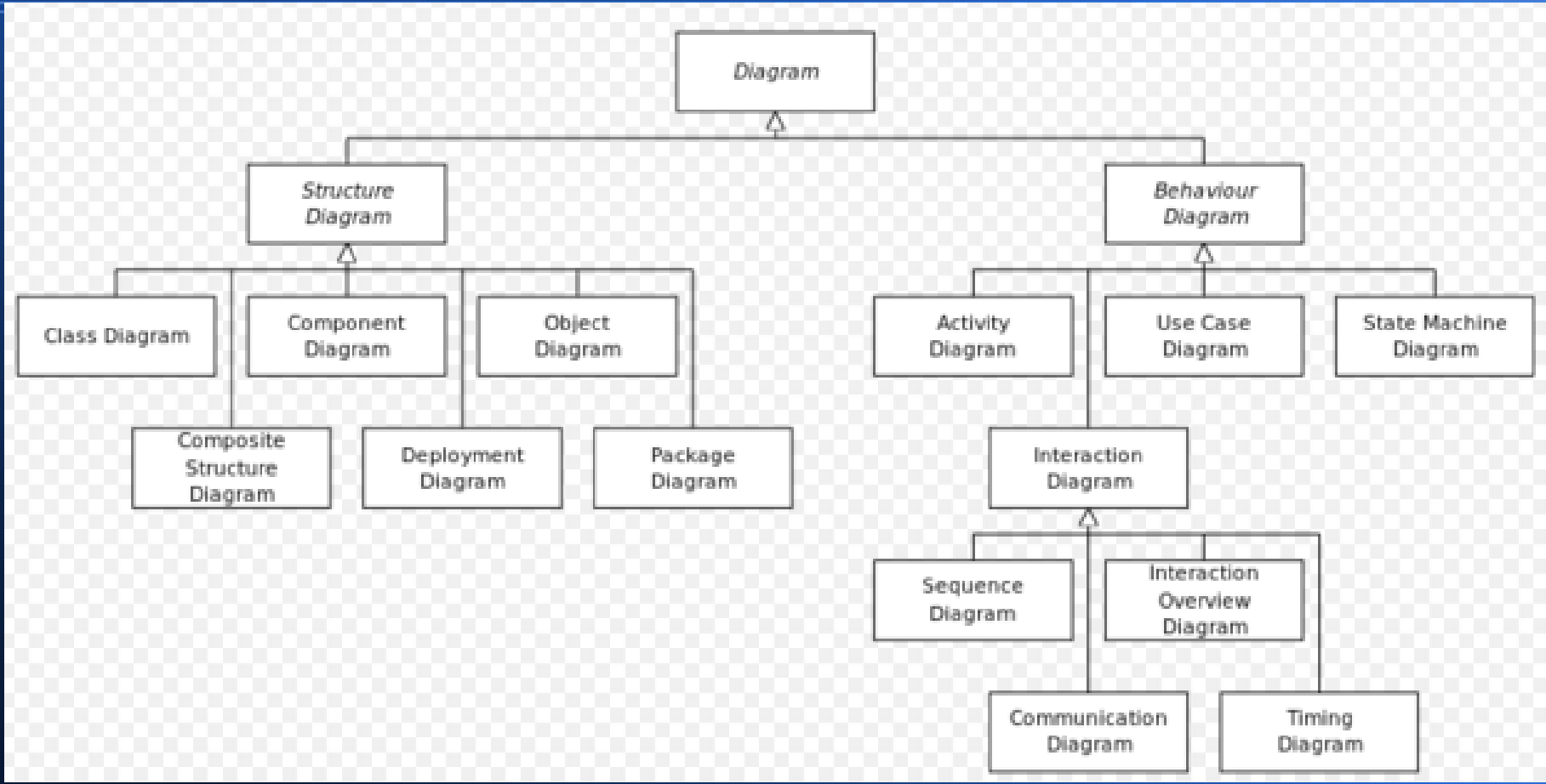
完成之後就可以劃分模組

- 然後進入設計階段

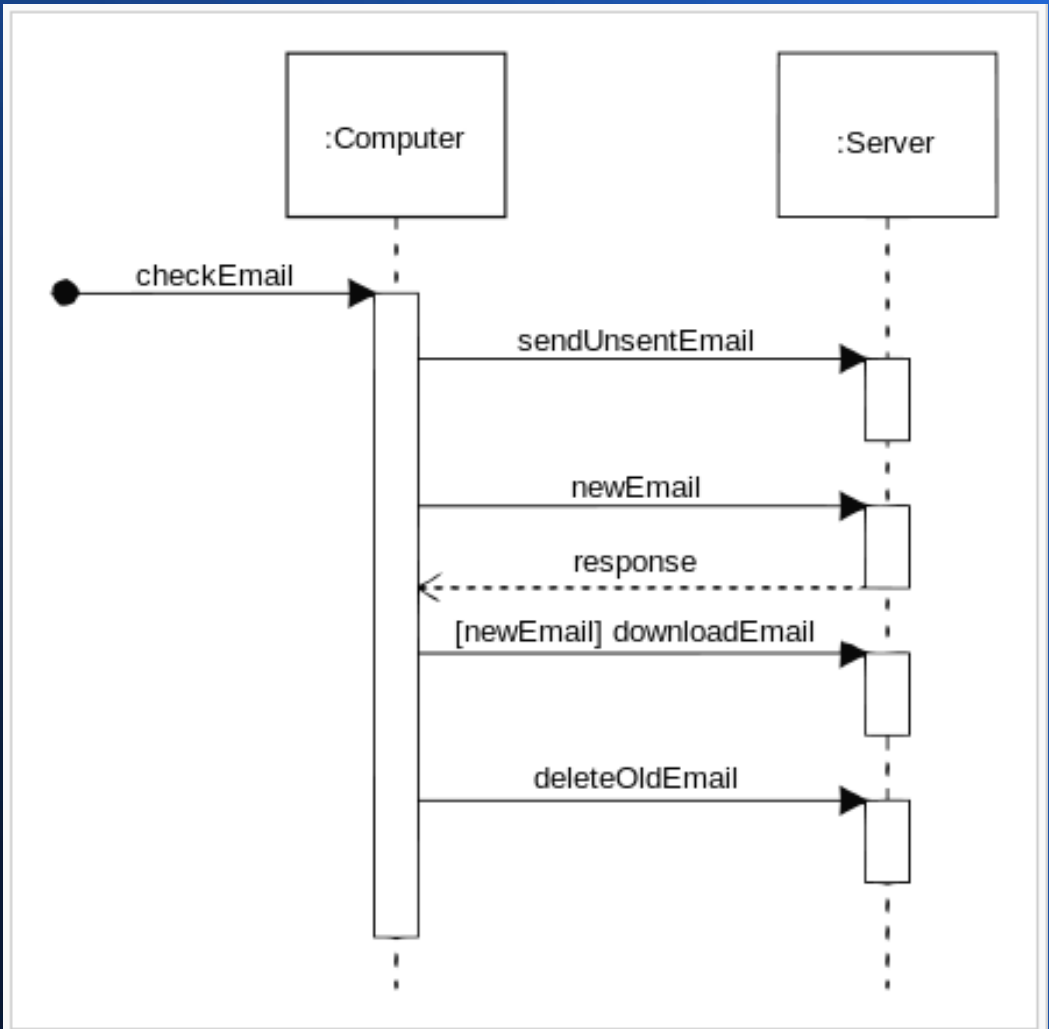
如果您用物件導向技術

- 先用《類別圖》描述類別與關係，並用《物件圖》描述某時刻的靜態物件關係。
- 然後用《循序圖》來表達物件間的互動行為
- 也可用《狀態圖》來表達物件狀態的轉移情況。
- 最後用《元件圖》描述程式檔案的布屬，用《部署圖》描述個機器與網路的真實配置情況。

類別圖（Class Diagram）

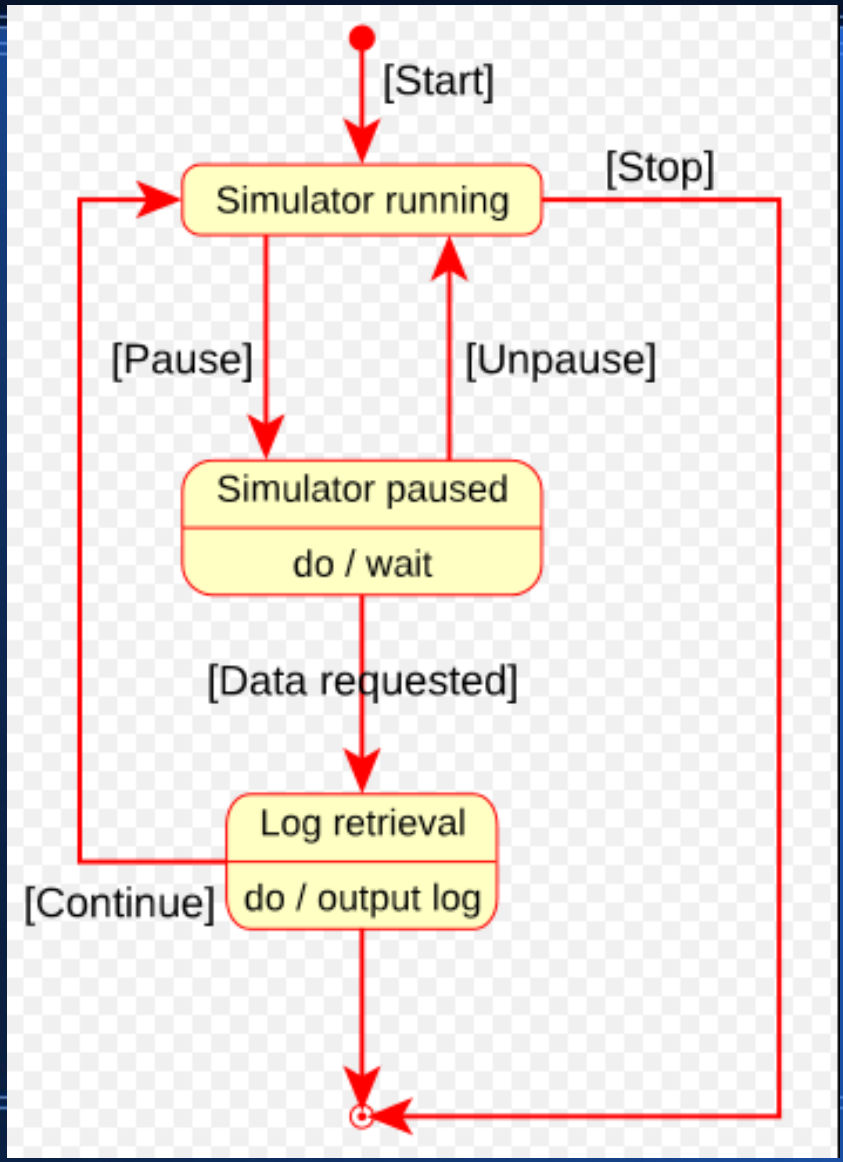


循序圖（Sequence Diagram）

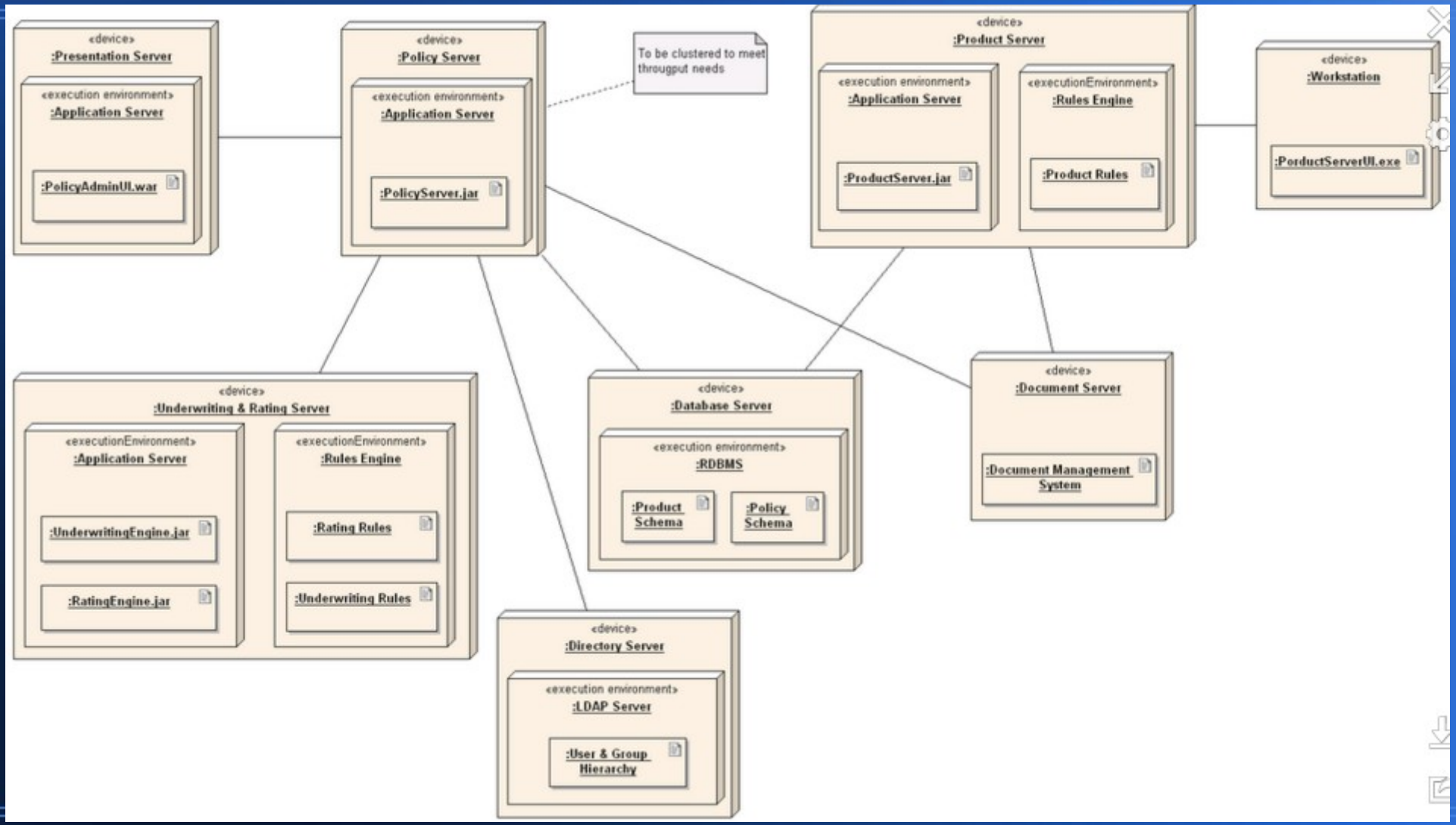


Sequence diagram of e-mail message sequence

狀態圖（State Diagram）



布署圖(Deployment diagram)



同樣在完成這些分析之後

- 就可以進入《系統設計》階段，開始訂定 API 函數介面了。

API 介面都定義完成之後

- 最好寫一份那些人負責實作那些函數的文件
- 或許加上時程表與甘特圖
- 然後就可以開始寫程式了

程式設計時需要的人力可能會比分析時多很多

- 像是 Joel Spolsky（約耳趣談軟體的作者）就曾描述他在 Excel 5.0 專案的分析階段只有不到十個人，但是當他們完成規格書之後，進入程式設計階段後曾經最多到達 500 人。
- 在設計完成並進行完單元測試之後，人力編制又回到了 10 個人的情況。

當然

- 在您進行開發的過程當中會用到很多工具
- 像是《除錯工具、錯誤管理、版本管理、單元測試、持續整合測試、壓力測試》等等。
- 所以您可能會需要用到 debugger, git, mocha, selenium, tarvis, jenkins 等各式各樣的工具
- 必要時還得開發獨門的特製工具。

而這些

- 就和您所使用的《語言、平台、和領域》等等都有密切的關係了。

軟體開發

- 絕對不是一件簡單的事情！

而這也是為何

- 人月神話的作者 Frederick P. Brooks 會說：《軟體開發沒有銀彈》的原因了！

希望

- 這份延伸為二十分鐘的十分鐘系列投影片
- 能夠提供您比較多一點的訊息。

或許這對您有所幫助！

我們下回見囉！