搞笑談軟工

敏捷開發，設計模式，精實開發，Scrum，軟體設計，軟體架構

2011 年 12 月 24 日 星期六

亂談軟體設計（1）：Cohesion and Coupling

December 24 22:41~23:57

最近因為有人問 Teddy 如何設計軟體架構與如何套用 design patterns，讓 Teddy 突然想整理一下所學過的一些軟體設計基本原則，也讓有心想學的鄉民們省一點學習的時間（前提是要先不走火入魔...XD）。這些原則和設計軟體架構與套用 design patterns 也許沒有直接的關係，但是對於『產生軟體設計』與『評估軟體設計』卻是簡單又實用的方法。

在開始講古之前，Teddy 心中一直有一個疑問，那就是：『鄉民們都是如何會設計軟體的？』是學校有教，自己看書，還是工作上學習而來？有空的話也請鄉民們留言分享一下這方面的經驗讓 Teddy 增廣一下見聞，感恩。

今天要談的是兩個基本中的基本觀念：

- **Cohesion（內聚）**：用白話文解釋，把執行某個功能所需用到的程式與資料都塞在某一個模組（function, class, package, etc）之中，使得該模組可被視為一個單獨的個體執行，那麼這個模組的 cohesion 就愈高。內聚，內聚，顧名思義就是把程式，資料這些有的沒的東西都『聚在一起』打包起來。
- **Coupling（耦合）**：如果某個模組跟『其他人（另一個模組）』有關係（例如，使用 global variables 或是接受其他模組傳入的參數）那麼這兩個模組就彼此耦合。

做軟體的人應該都知道，設計一個模組的基本精神，就是要『提高內聚力，降低耦合度』（如果原本不知道也沒關係，至少現在知道了）。提高內聚力的好處就是提高了模組的『獨立性』，也就是說這個模組可以被單獨使用，也可以被單獨修改。這兩點都很重要，因為可以被單獨使，就表示模組被『重複使用（reuse）』的機會變多了；可以被單獨修改，就表示開發人員可以放心大膽的修改模組而不用怕萬一改了模組之後會引發『漣波效應（ripple effect）』影響到其他原本功能正常的模組（就是可以避免不小心改一的地方錯十個地方）。

但是，軟體模組就跟人類一樣是『群居動物』，不太可能生活上食衣住行育樂一個人全部搞定。當然『程式是人寫出來的』，硬要寫出內聚力超高的模組也不是不可能。但是，一個內聚力超高的模組可能會發生以下兩個問題：

- 你的軟體只有一個模組，所有需要的程式碼與資料都在這個模組中。此模組內聚力超高，但是...一共有一萬行。這種程式，應該不太好維護吧。

- 為了不讓模組變得太大但是又要有很高的內聚力，當有新功能要開發的時候，你用 copy and past 的方式產生新的模組。新的模組內聚力很高，可以單獨執行（因為所需的資料與程式碼都被 copy 到新的模組中了），但是，這衍生了 duplicate code（重複程式碼）問題，此為軟體大忌，戒之，戒之。

所以，模組之間還是免不了需要『有關係』，也就是說耦合是難免的。但是不當的耦合（找小三，小四...XD），例如使用全域變數（global variables）就很容易產生一些很難看出來的 bugs（因為人人都可修改全域變數，萬一到時候資料不正確很難找到兇手）。如果是設計單一的 function 或是 method，那麼最低的耦合就是所謂的『資料耦合（data coupling）』，也就是說以參數傳遞的方式作為模組溝通的管道。但是，如果把『眼界放大一點』，看的是 design patterns 或是 architecture 中各個不同『角色（或是參與者）』的溝通方式要採用哪種耦合呢？

鄉民們不知道是否還記的『Design Patterns 分成三大類』這一篇，裡面有提到有一大類的 patterns 是屬於『patterns relying on abstract coupling』，包含了 State， Factory Method， Observer， Bridge， Builder， Command， Visitor， Interpreter， Mediator， Adapter， Prototype， Proxy， Strategy。那麼什麼又叫做『abstract coupling』？簡單的說，就是定義介面（interface），讓需要『有關係』模組透過介面來產生關係。所以 GoF 在 design patterns 書中開宗明義就說：

program to an interface， not an implementation

Abstract coupling 的概念應用很廣，像是 Layers 這個軟體架構上，下兩個 Layers 也是 abstract coupling 的關係。有一句話 Teddy 再強調一次：

模組之間要有關係（耦合）可以，但是最好只能有『~~精神外遇~~ 抽象關係』，不要有『~~肉體耦合~~ 實做耦合（implementation coupling）。』

友藏內心獨白：這一篇算是『Design Patterns 分成三大類』的補充包嗎，還是耶誕禮物？
張貼者： Teddy Chen 於 下午 11:58