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 25 日 星期日

亂談軟體設計（2）：Open-Closed Principle

今天原本應該要講第二個重要的設計原則『program to an interface』，不過這個原則之前 Teddy 已經講過了（請參考這篇 Program to an interface），所以直接跳下一個：

The Open-Closed principle (OCP)

OCP 是 Teddy 相當喜歡而且經常應用的一個設計原則，Teddy 第一次看到這個原則是在 Bertrand Meyer 所寫得長達一千兩百多頁的巨著 Object-Oriented Software Construction, 2nd，57 頁。後來 Robert C. Martin 在 Agile Software Development 這本書第 99-109 頁有重新詮釋，這本書的解釋比較容易理解。Teddy 引用一下 Martin 對於 OCP 的解釋：

OCP: the Open-Closed Principle

> *Software entities (class, modules, functions, etc.) should be open for extension, but closed for modification.*

一個軟體個體應該要夠開放使得它可以被擴充，但是也要夠封閉以避免不必要的修改。這樣解釋鄉民們應該還是看不懂，接著看 Robert 在書中所說得這段話就會很清楚了 (p. 99)：

> *When a single change to a program results in a cascade of changes to dependent modules, the design smells of Rigidity. The OCP advises us to refactor the system so that further changes of that kind will not cause more modifications. If the OCP is applied well, then further changes of that kind are achieved by adding new code, not by changing old code that already works.*

鄉民們知道開發軟軟體最怕的是什麼嗎？請花 30 秒想一下…

開發軟體最怕的不是『寫程式』或是『不會寫程式（寫不出程式）』，最怕的是『改程式（改那些原本已經寫好測試過可以動的程式）』。先談一下既然程式已經寫好了，為什麼還要去改它呢？兩個主要的原因：

- 要增加功能。
- 要修正問題（bugs）。

那為什麼改程式那麼可怕？因為要修改一個已經可以動的系統的某個模組，很容易造成『*a single change to a program results in a cascade of changes to dependent modules*』，這也就是昨天提到的因為模組之間會有 coupling（耦合），因此產生『牽一髮而動全身』的效應。如果鄉民們的系統都有足夠的『自動化測試案例』來驗證『此次修改沒有破壞原本可以正常動作的功能』，那麼相對來講這種『漣波效應（ripple effect）』其實也就沒有那麼可怕。但是實務上大家都知道『自動化測試永遠都不夠多』，所以如果能從『設計面』著手從而避免掉這種牽一髮而動全身的漣波效應，則軟體系統將會變得很容易修改，擴充，了解，與維護。

所以，套用 OCP 的最高境界就是做到：

藉由增加新的程式碼來擴充系統的功能，而不是藉由修改原本已經存在的程式碼來擴充系統的功能 (If the OCP is applied well, then further changes of that kind are achieved by adding new code, not by changing old code that already works.)

Teddy 第一次看到這段話的時候真的是百思不解，奇怪，要增加新的功能不就是要把程式拿出來改嗎？不改原本的程式，光靠『增加新的程式』要如何擴充系統的功能啊？

先解釋一下為什麼 OCP 最高境界要求鄉民們『藉由增加新的程式碼來擴充系統的功能，而不是藉由修改原本已經存在的程式碼來擴充系統的功能』。因為如果能夠不修改原本已經存在的程式，那麼就可以完全避免掉『*a single change to a program results in a cascade of changes to dependent modules* 這種牽一髮而動全身的漣波效應』...乍看之下覺的有點逃避的鴕鳥心態...XD。

那麼要如何做到藉由增加新的程式碼來擴充功能？講穿了也是很間單，又回到昨天所說得『abstract coupling』與『Program to an interface』這些原則上面。Robert 在書中有舉了一個淺顯易懂的例子並附上範例程式，有興趣的鄉民們請自行參考。Teddy 要講的是不用看程式碼就可以讓鄉民們了解的例子-- Plug-in architecture 。

如果有用過 Eclipse，對於 plug-in architecture（外掛架構 or 插件架構）應該很了解。不會寫程式的鄉民也沒關係，像 Firefox 或是 Chrome 這些瀏覽器，也都有支援所謂的 plug-in architecture。Plug-in architecture 就是一種十分符合 OCP 的設計。因為在 Plug-in architecture 架構中，開發者是藉由『撰寫新的 plug-ins 來擴充系統的功能』，使用者也不太可能（也不需要）去修改 Eclipse 來擴充功能。

類似的例子還有很多，例如現在很流行的 Dependency Injection（例如 spring framework），還有古早用查表方式，透過 service registry 或是 service look-up 的分式，以及透過 ini 或是 XML 設定檔來改變系統行為，甚至是 Windows 系統的 hook 程式 Teddy 認為都可以算是某種 OCP 的實踐。

總之，OCP 的精神『further changes of that kind are achieved by adding new code, not by changing old code that already works.』這句話背下來放在心裡面就對了。

友藏內心獨白：要學『美味關係』每天都寫一篇部落格文章還真是很拼耶。

張貼者： Teddy Chen 於 下午 10:12