How To Solve It (HTSI)支援工具

專題編號：101-CSIE-S016
執行期限：100 年第 1 學期至 101 年第 1 學期
指導教授：鄭有進
專題參與人員： 98590316 張竣翔
98590342 李旻憲

一、摘要

協助程式初學者在開發 C/C++ 程式過程中可以依照計畫與有目的的撰寫程式，養成良好的撰寫習慣。此專題利用 Eclipse 開發一個外掛模組讓程式設計師進行開發工作時結合 How To Solve It 來解決在撰寫過程中的各項問題，此系統並結合 Cute C++ Unit Testing Easier 外掛模組來協助程式設計師檢測 TODO 是否有被實作及撰寫對應測試，除此之外也提供 TODO 所對應之測試的通過與失敗數量列表給程式設計師確認 TODO 執行成效做為參考。

關鍵詞：Eclipse、解題方法、工作規劃、單元測試、軟體工程實務技術

二、緣由與目的

許多程式初學者，因無經過妥善思考規劃或誤解題目意思，導致撰寫出的程式不符合預期，程式初學者可以藉由使用 How To Solve It 的解決方法來培養良好撰寫程式習慣 [3] 。

本專題旨在開發一個支援 How To Solve It 解決方法之協助工具，簡稱為 HTSI 工具。HTSI 工具的開發平台為基於 Eclipse Plugin 的設計架構[1, 2]，整合至 Eclipse 整合式開發環境，並結合其他 Ecllipse 工具，如 Eclipse CDT (C/C++ Development Tooling)[4], Cute C++ Unit Testing Easier [5]等工具，讓使用者在開啟專案後可以替專案建立專屬的問題描述(Problem Statement)，在每項問題描述中可以擁有各自的工作列表，每項工作(TODO)可指定一至多個測試案例。

三、系統需求

HTSI (How To Solve It)是一套解決問題的方法，如表一所示，可分為了解問題、訂定計畫、實行計畫及回顧四個步驟。

表一: HTSI (How To Solve It)方法[7]

1. 了解問題
   仔細閱讀題目，充分了解題目所表達的意思。
2. 訂定計畫
   2.1 依據問題，列出工作清單
   2.2 依據問題設定工作的優先權
3. 實行計畫
   3.1 一次只處理一項工作
   3.2 撰寫每項工作的單元測試
   3.3 若工作對應的測試皆通過代表此項工作完成
4. 回顧
   4.1 重看一次問題及工作清單
   4.2 更新工作事項（如：新增事項，刪除不需要的事項等等）
   4.3 重複步驟直到工作清單全部完成

經由上述 HTSI 之方法，擬出使用者在套用 HTSI 開發程式的方法可能的操作與所需之介面，訂定 HTSI 的需求如下：

(1) 作為使用者，我可以在 Eclipse 的專案上啟動 HTSI Plug-in 的主畫面。
(2) 作為使用者，我可以將 TODO 選擇對應到多個 Test Method，並能分別統計其通過與失敗的數量。
(3) 作為使用者，我可以針對不同的專案建立不同的 Problem 來處理問題。
(4) 作為使用者，我可以在 HTSI Plug-in 中，當所有的 TODO 都已標記完成

後，Problem 的狀態變更為 Solved 狀態。

(5) 作為使用者，我可以在 HTSI Plug-in 上管理 Problem。

(6) 作為使用者，我可以在 HTSI Plug-in 上管理 Problem statement。

(7) 作為使用者，我可以在 HTSI Plug-in 上檢視所有的 TODO，並可依照各項 column 作為排序。

(8) 作為使用者，我可以在 HTSI Plug-in 上新增/刪除/修改一個 TODO。

四、專題設計與實作

在開發此 HTSI 工具中我們將整個 plugin 主要區分為兩大區塊，一個是可以對應到所選擇專案的 Problem statement，另一個則是所對應的 TODO。使用者可以在開啟專案後建立許多個 Problem statement 來記錄所需要解決的大項目問題，在此 Problem statement 中便可以新增許多需要完成的 TODO，每一個 TODO 可以擁有描述、對應測試類別、對應物件、檢核狀態等屬性。

我們基於 Eclipse 的設計框架，建立自己的 TODO View 以及 ProblemStatment View，並且套用許多 Design Patterns[6] 至 HTSI 工具中。例如，我們套用 Observer Pattern，使得 Model 改變時，會以 Observer 通知兩個 View 更新。而在 TODO view 中所指定的各項測試則是透過 Eclipse CDT 中 ITranslationUnit 來取得專案的程式碼所對應的 AST Tree[8] (由 Eclipse CDT 自動編譯程式碼所建立)。AST Tree[8]包含每個類別中的所有函式，在此我們套用 Visitor Pattern 來尋訪以擷取出我們所要的部分，並分析該函式是否已被實作，以及是否有對應的測試類別。

在整合 Cute C++ Unit Testing Easier，提供圖形化介面的測試介面，並且測試結束時，會送出測試結果到 TODO View 中，以確認 TODO 所對應的 Test 已全數通過，並且至對應到一個以上的函式，則使用者可以將 TODO 設置為完成狀態。

五、參考文獻

[1] Erich Gamma, Kent Beck, *Contributing to Eclipse*, Canada: Pearson Education, Inc., 2004

[2] Eclipse, http://help.eclipse.org/indigo/

[3] George Polya, *How to Solve It*, Princeton University Press, 1957

[4] Eclipse CDT, http://www.eclipse.org/cdt/

[5] Cute C++ Unit Testing Easier, http:// http://cute-test.com/

[6] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison Wesley Professional, 1994

[7] 鄭有進、陳偉凱，「融入軟體工程實務於物件導向程式設計課程」，**第二十一屆物件導向技術及應用暨第六屆台灣軟體工程研討會論文集**，桃園，2010/7/22~2010/7/23, P16-P30

[8] Eclipse AST Tree, http://www.eclipse.org/articles/article.php?file=Article-JavaCodeManipulation_AST/index.html