一個支援頻繁上機考試的程式練習系統

董少桓、林紹陽
國立雲林科技大學
資訊管理系
Email:{tungsh, M10223011}@yuntech.edu.tw

曾筱倩
資訊工業策進會
數位教育研究所
Email:tseng.tsc@gmail.com

程式設計是需要經過練習，才能學會的學科。因此，「程式練習題」應該是程式設計課程的主要教材之一。本篇論文介紹了一個能夠協助教師出題、學生練習以及頻繁的實施上機考試的程式設計練習系統。初步的評估顯示，頻繁的上機考試對學生的學習成效，有正面的影響。

***關鍵詞*—程式設計教學、上機考試、自動評量**

An exercise management and lab exam system for teaching programming

The way to learn programming is by writing programs and one of the primary learning materials of programming courses should be programming exercises. This paper introduces the PLWeb programming learning and lab exam system that can be used not only as an authoring tool for instructors to compose exercises but also as a novice-friendly editor for students to practice programs and submit solutions. In addition, PLWeb also provides mechanisms to conduct frequent lab exams. Preliminary study reveals that frequent lab exams provide positive impact on students learning outcomes.

Keywords: Programming Education, Lab Exam, Automatic Assessment

一、簡介

對初學者而言，程式設計是一門不易學習的學科。Lahtinen 在一篇研究何種程式設計的教材與教學方式，比較適合學習者學習的論文中指出，上機練習要比聽教師講解有更好的學習效率，而程式設計範例與練習題則是最能幫助學習者學習的教材[10]。然而，教師卻缺乏能協助編輯程式設計的題目說明、測試案例、程式碼提示，並將這些練習透過網路平台有效的傳遞給學生演練的工具，使得有些學生仍然傾向於以聽講與閱讀的方式學習，而不是以更適合程式設計的「練習」的方式學習。

此外，「考試領導教學」也是一個普遍而顯而易見的現象。目前常見的考試方式包括：是非、選擇、填空、問答、申論與口試等。這些考試方式，能夠評量不少學科的學習成效；但是，程式設計的學習目標是會寫程式，而「是否會寫程式」卻難以用是非、選擇等考試方式進行評量[11]。如果教師仍然以上述的方式評量程式設計課程的學習成效，那麼勢必會影響學生的學習方式與準備考試的方法，進而影響學生學會寫程式的目標。

在一些關於考試頻率與學習成效之關係的研究中發現，頻繁的考試，如每週或隔週一至二次的考試，能夠提生學生的學習動機與成效[2,3,12]。可惜的是，由於預算有限、修課人數過多、人工評量耗時耗力、教學負擔過重與缺乏獎勵等原因，使許多程式設計課程仍然沒有使用上機考試來進行評量[9]。而且，在這些限制下，即使進行上機考試，其實施次數也受到限制。

此外，上機考試還需要顧慮學生是否會透過網路搜尋或互傳答案；而如果不用網路，又無法自動的繳交與批改程式碼。總而言之，程式設計課程的上機考試，實在是一件需要教材設計、平時練習與軟硬體設施都緊密配合才能

夠實施的工程。

為了能提供充分的題目，讓學生在平時便能準備上機考試，教師便必須將題目設計妥當，上傳到網路，然後讓學生下載練習。不同於一般學科的練習題，程式設計練習題，需要包含題目說明、測試案例以及某些題目所提供的解答提示幾個部分。如果沒有適當工具的協助，那麼製作這些練習題，並將它們打包與上傳到網路，勢必會對教師的工作產生很大的負擔。此外，教師也需要能夠協助檢測學生的程式是否通過測試的工具。這些繁瑣的過程，如果沒有軟體的協助，那麼不論對教師或是學生而言，都將是相當耗時耗力的工作。

為了解決這些問題，本篇論文介紹了一套能夠支援教師出題，學生練習，同時在監考人員的巡視下，可以頻繁的在電腦教室實施上機考試的程式練習管理系統——PLWeb。PLWeb 的教師出題與學生練習的功能已經使用多年[13]，它的上機考試系統則是在 2014 年新增的功能。本篇論文介紹 PLWeb 的設計與使用，同時也初步評估了頻繁的上機考試對學生學習成效的影響。

二、相關研究

不同於單純的紙筆測驗，程式設計課程的上機考試，需要軟硬體與行政程序的緊密配合，才能實施。因此不同的學校，常常因為各自的狀況不同，而有不同的上機考試方式。在Cutts、Barnes、Bibby 和 Brown 等人發表的一篇綜合報告了七所英國大學如何進行上機考試的論文中，作者依照考試的時間與試題的份數，將這些學校的考試方式，分成了以下三種[4]：

1. 同一時間考試、考題未公布：這種方式類似傳統的方式，也是最單純的方式。
2. 不同時間考試、考題未公布：這種方式需要設計難度類似，但是版本不同的試題，以防止已考過的學生將試題傳遞給尚未考的學生。
3. 不同時間考試，考題已公布：這個方式的解答程式碼，必須在沒有充分練習與瞭解的情況下，無法輕易的完成。

這些考試方式的可靠性、有效性與可擴充性，並不完全相同。可靠性高的考試考題相同，或難度類似，而且學生不易在監考人員的巡視下作弊。就以上三種方式來說，「不同時間、考題未公布」由於不同的試題可能使成績產生差異，因此可靠性最差。有效性是指考試的狀況是否與程式設計師實際的工作狀況一致，例如：是否可以如同程式設計師一般的查考書籍、上網找資料或詢問其他程式設計師。可擴充性是指在班級人數增加的狀況下，考試是否仍然能順利運作，而「不同時間、考題未公布」的擴充性最受限制。雖然有這麼多不同的方式，但是受限於考試的成本，這些學校在一學期中，最多只實施了一或兩次的上機考試。

由於上機考試不易管理與實施，因此，Farrow 和 King 開發了一套使用 BlueJ 的 Submitter 套件與 JUnit 來實施上機考試的系統。為了能讓學生在一至三小時的考試時間內，完成考試，這個系統的考題，提供了部分的程式解答，然後要求學生以改正錯誤或擴充程式的方式完成考試。然而，這套系統需要與網路連線才能完成繳交與評分的動作。然而，因為沒有提供防弊機制；所以，只能「相信」學生不會藉由網路作弊[8]。

為了解決這個問題，英國的 Brighton 大學發展了一套讓學生只能連接考試主機下載考題，並在完成考試後，上傳答案的考試系統[7]。然而，這套系統並沒有提供教師製作程式練習與考題的工具，因此不易推廣。

以上實施上機考試的方式與系統，可以應

付在一學期的課程中實施少數次數的上機考試。然而，如果要在一個學期中，頻繁的在每週或隔週便實施上機考試，那麼除了要能更方便的支援考試現場的需要之外，還必須讓教師在平時便可以提供學生充分的練習以準備考試。整體而言，一個能頻繁實施上機考試的系統，必須包含以下幾個功能：

1. 提供教師用的「程式練習題製作工具」（authoring tool），以幫助教師將出題時，所要撰寫的題目、程式碼、提示、測試案例以及上傳主機等工作，更有效率的進行。
2. 幫助學生練習寫程式的「程式練習環境」，讓學生能方便的下載並練習程式設計練習題，以預備上機考試。
3. 能讓學生利用網路繳交程式，但又無法作弊的「考試用作業環境」，以減輕上機考試的成本。
4. 能夠繳交與偵測學生的程式是否通過測試的「自動評量工具」，以減輕教師閱卷的負擔[1,5,6]。

不同於只能提供以上四種功能之一部份的上機考試系統。PLWeb 的特色是能夠同時支援以上四種功能。不但如此，PLWeb 還提供了教師在電腦教室上課時監控學生答題狀況，讓教師可以針對學生的需要以提供個別指導的功能。

三、系統功能簡介

PLWeb 有二個主要的部分：伺服器與程式編輯器。程式編輯器是從一個以 Java 撰寫的開放源碼程式編輯器——jEdit，修改並延伸而來。程式編輯器中包含四個區域（圖 1）：題目描述區（左上）、編輯區（右上）、輸出樣本區（左下）、以及測試區（右下）。題目描述區是一個迷你瀏覽器，它可以將題目的內容顯示出來。如果該題目提供部分解答，則部分解答會被自動的載入編輯區中。而學生可以將編輯區中的內容以填入空格、延伸功能或自行全部編寫的方式來完成練習題。

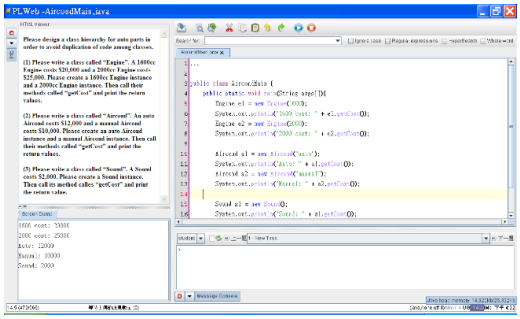

圖 1 程式編輯器

當學生在網頁中按下「開始練習」按鈕時，系統會將程式編輯器伴隨著一組練習題從伺服器下載到客戶端。學生可以依照題目的描述，在編輯區中開始練習。學生若按下「執行」按鈕，程式編輯器就會呼叫預先安裝在學生電腦中的編譯器，將編輯區中的程式進行編譯與執行。程式編輯器會自動比對學生程式的輸出和輸出樣本區的內容是否相同，以判斷學生的程式是否通過測試。而測試的結果會被自動地送至伺服器，並儲存在資料庫中。

程式編輯器不僅可以讓學生練習程式設計，還可以協助教師製作練習題目。當程式編輯器處於作者模式時，它會在編輯區中同時開啟三個頁籤：題目描述、解答以及部分解答。教師可以任意切換並編輯這三個頁籤的內容。當教師按下「執行」按鈕後，便可以在命令提示視窗中輸入測試資料。而命令提示視窗中的內容會自動地被儲存起來，當作輸出樣本區的內容。在結束練習題的建立之後，練習題作者可以按「右箭頭」按鈕，繼續建立下一個練習題。在建立所有練習題之後，練習題作者可以按下「上傳」按鈕，將練習題打包並上傳至伺服器。

為了支援上機考試，PLWeb 也提供了教師製作隱藏的測試案例的功能。在考試時，學生可以在通過輸出樣本區中測試案例的測試後，再「提交」考題，這時隱藏的測試案例，可以接著檢測學生的程式是否確實達到題目的要求，

而不只是以使用 printf 或其它不合題目要求的方式印出輸出樣本區答案。

PLWeb 的「考試用作業環境」讓學生在監考人員的巡視下無法作弊。這個考試用作業環境在 Windows 作業系統的實現方式，是藉著下載的程式編輯器在使用者端取得系統權限後，便將所有視窗桌面上的程式全部關閉。而為了防止學生使用 Ctrl-Alt-Delete 三鍵同時按下，並使用工作管理員自行啟動其他軟體，程式編輯器會每隔一秒便自動關閉可能已被啟動的工作管理員。這個方式讓學生除了使用仍然與主機相連的程式編輯器外，沒有任何其他的應用程式可以使用，因而得到了一個簡易、容易使用而且實施成本低的考試用作業環境。

四、使用經驗與評估

PLWeb 提供了以練習題方式呈現的 Scheme、C 與 Java 的完整教材。每份教材均包含了 150 題以上的中小型(5-300 行)的程式設計練習題。教師的授課方式是以練習為主，講授為輔。在課堂中也配置了助教以協助學生解決練習過程中所遭遇的問題。

由於正式評估頻繁的上機考試對學習成效的影響，需要一個學期的時間，而在同一個科系中以不同的方式對參與的學生進行授課與考試，會影響到學生分數的公平性；因此，我們以跨年度比對的方式，為雲林科技大學資訊管理系，大一上學期開設的「程式設計概念與方法」的 2012～2014 年三年期間，修課同學的期初與期末考答題的資料進行了比較與分析。

「程式設計概念與方法」使用 Scheme 語言進行授課。Scheme 是 Lisp 的一種，它的特色是可以用少數的語法表達多樣的程式設計概念。這門課的授課內容包括：條件判斷、迴圈、遞迴、高階程序、陣列、排序與二元搜尋樹等。

在 2012 與 2013 年的課程中，共進行了三次分別佔學期總成績 30%、30%與 40%的上機考試；而在 2014 年，則進行了 11 次總共佔學期成績 70%的平時考與一次佔學期成績 30%的期末考。2012 與 2013 年的閱卷方式是由監考人員在學生繳交之後，比對學生電腦上的程式是否與教師所要求的撰寫方式相同來驗證學生的答案。而 2014 年的考試，則使用含有隱藏式測試案例的自動評分系統進行評分。

在這三年的課程中，教師都使用同樣的題庫供學生練習，而練習的解答也在題目開放練習後 3-4 天，便開放給學生參考，而學生也被事先告知有些考題會與題庫中的題目相同。雖然題目相同，但是如果學生在不理解授課內容與這些題目的演算邏輯的情況下，幾乎不可能只靠記憶便在考試中複製相同的程式解答。因此即使使用了與題庫相同的考題，還是能相當可靠的測驗出學生是否理解這些題目的演算邏輯與撰寫程式的過程。

在這樣的方式下，我們比較了考試範圍同樣是遞迴程式設計的 2012 與 2013 年期初的第一次大考與 2014 年的第三次小考中題目相同的兩個題目。這兩題是 product-sum 及 average>=60，product-sum 是以一個遞迴程序呼叫另一個遞迴程序以計算結果的程式；average>=60 則是以遞迴計算一個含有數筆學生成績資料的 list 中，平均成績大於或等於 60 的學生姓名。表格 1 是這兩題在 2012、2013 與 2014 年，不含重修生的答對率統計：在 product-sum 的答對率統計中可以發現，2012 年的答對率為 0.66、2013 年的答對率為 0.60，而 2014 年則提升為 0.77；在 average>=60 的答對率統計中可以發現，在 2012 年的答對率為 0.59、2013 年的答對率為 0.40，而 2014 年則提升為 0.67。推測 2014 年第三次小考的答對率略有提升的原因是在這次考試之前，先進行了更基礎的「資料型態」及「條件判斷式」兩次的上機考試，而在 2012 與 2013 年的第一次考試前，並沒有進行這兩次的上機考試。

此外，我們也比較了這三年的期末考中四題相同的題目。這四題分別是用遞迴撰寫的 insertion sort，應用高階程序觀念的 map，同時應用了遞迴、陣列與高階函數的

vector-compute 及二元搜尋樹。表格 2 與 3 我們分別以 t 檢定比較了 2012 年與 2014 年以及 2013 年與 2014 年，不含重修生的修課同學，答對這四題的人數佔該班人數的比率，而結果顯示頻繁的上機考試顯著的提升了學生的學習成效，且皆達統計上的顯著，其中 2012 與 2014 年的比較中，在題目二元搜尋樹中，學生的答對比率從 0.125 提升至 0.819，為最具提升的考題；而在 2013 與 2014 年的比較中也有相似的結果。數據顯示頻繁的上機考試，讓大部分的同學在平時便需勤加練習，以至於能夠更完整的累積課程中所學習到的知識與經驗；因此，在期末考面對難度較高的教材與考題時，學生比較不易放棄，因而能夠有比較好的成績。

五、結論

本篇論文介紹了一個能夠支援教師出題、學生練習、自動繳交、自動評分與上機考試的程式練習系統。這些功能可以幫助教師實施頻繁的上機考試。而初步評估顯示，頻繁的上機考試對學生的學習成效有正面的影響。

致謝

本研究依經濟部補助財團法人資訊工業策進會「104 年度資策會創新前瞻技術研究計畫(1/1)」辦理。

參考文獻

[1] K. M. Ala-Mutka, “A survey of automated assessment approaches for programming assignments”, Compute Science Educaton, Vol. 15(2), pp. 83-10, 2007.

[2] S. M. Brookhart and J. G. De Voge, “Testing a theory about the role of classroom assessment in student motivation and achievement”, Applied Measurement in Education, 12(4), 409-425, 1999.

[3] A. C. Butler and H. L. Roediger, “Testing improves long-term retention in a stimulated classroom setting”, European Journal of Cognitive Psychology, Vol. 19(4/5), pp. 514-527, 2007.

[4] Q. Cutts, D. Barnes, P. Bibby, J. Bown, V. Bush, P. Campbell, and C. Whyley, “Laboratory exams in first programming courses”, In 7th Annual Conference of the Higher Education Academy for Information and Computer Sciences, pp. 29-31, Dublin, Ireland, 2006.

[5] C. Daly and J. Horgan, “A technique for detecting plagiarism in computer code”, The Computer Journal, Vol. 48(6), pp. 662-666, 2005.

[6] C. Douce, D. Livingstone, and J. Orwell, “Automatic test-based assessment of programming: A review”, Journal on Educational Resources in Computing, Vol. 5(3), 2005.

[7] J. English, *Experience with a computer-assisted formal programming examination.* The 7th Annual Conference on Innovation and Technology in Computer Science Education, pp. 51-54, University of Aarhus, Denmark, 2002

[8] M. Farrow and P.J.B King, “Experiences with online programming examinations”, *IEEE Transactions on Education*, Vol. 51(2), pp. 251-255, 2008.

[9] W. L. Kuechler and M. G. Simkin, “How Well Do Multiple-Choice Tests Evaluate Student Understanding in Computer Programming Classes?”, Journal of Information Systems Education, Vol. 14(4), pp. 389-400, 2003

[10] E. Lahtinen, K. Ala-Mutka and H.-M. Järvinen, “A study of the difficulties of novice programmers”, ACM SIGCSE Bulletin, Vol. 37(3), pp. 14-18, 2005.

[11] S. Sheard, J. Sheard, A. Carbone, D. Chinn, M.

Laalso, T. Clear, and G. Warburton, "Introductory programming: examining the exams", In Proc. Australasian Computing Education Conference (ACE2012), pp. 61-70, Melbourne, Australia, 2012.

[12] H. Shirvani, "Examining an assessment strategy on high school mathematics achievement", American Secondary Education, Vol. 38(1), pp. 34-45, 2009.

[13] S. H. Tung, T. T. Lin, and Y. H. Lin, "An exercise management system for teaching programming", Journal of Software, Vol. 8(7), pp. 1718-1725, 2013.

表格 1 2012、2013 與 2014 期初考試答題率的差異

年份與考試\考題	product-sum	average>=60
2012 第一次	37/56 = 0.66	33/56 = 0.59
2013 第一次	37/61 = 0.60	25/61 = 0.40
2014 第三次	47/61 = 0.77	41/61 = 0.67

表格 2 2012 與 2014 期末考答題率的統計表

	2012 年未實施頻繁上機考			2014 年實施頻繁上機考			
考題	人數	平均	標準差	人數	平均	標準差	顯著性
Insertion sort	56	.321	.471	61	.819	.387	.000***
map	56	.321	.471	61	.508	.504	.041*
Vector-compute	56	.178	.386	61	.590	.495	.000***
二元搜尋樹	56	.125	.333	61	.819	.387	.000***

* $p < .05$，** $p < .01$，*** $p < .001$

表格 3 2013 與 2014 期末考答題率的統計表

	2013 年未實施頻繁上機考			2014 年實施頻繁上機考			
	人數	平均	標準差	人數	平均	標準差	顯著性
Insertion sort	61	.295	.459	61	.819	.387	.000***
map	61	.098	.3	61	.508	.504	.000***
Vector-compute	61	.262	.443	61	.590	.495	.000***
二元搜尋樹	61	.213	.412	61	.819	.387	.000***

* $p < .05$，** $p < .01$，*** $p < .001$