# 應用 Rubric 評估程式設計課程學習成效與改善策略

施再繁
朝陽科技大學副教授
e-mail：
tfshih@cyut.edu.tw

張德明
朝陽科技大學研究生
e-mail：
milk.truong@gmail.com

## 摘要

程式設計是資訊領域中一門很重要的課程。在多年教學中，我們觀察到學生此門課的學習成效普遍不佳，大約只有半數學生可以順利一次通過測驗,剩下的學生則常需要反覆進行多次輔導才可以通過，甚至還有小部分仍無法順利通過測驗必須重修課程。透過檢查、分析學生的解答並訪談學生，我們觀察到學生常遇到的學習困難為語法錯誤、不瞭解題意或無法解比較複雜或數學相關類型的題目。綜合這些原因之後，我們覺得必須找出一個可以清楚描述學生在分析題目要求、設計解題演算法和撰寫程式碼三種能力的方法，藉以評估學生學習成果。

rubric為近來評估學習成效的熱門方法，不只可讓老師評估學生能力，還可讓學生知道應該達到哪些能力。在本論文中，我們應用rubric來評估學生學習成效。我們根據歷年學生學習過程所遭遇的各種困難，設計rubric來評量學生的學習成效，首先會以rubric來建考題，然後根據rubric問卷分析的結果，推斷學生的問題並進行合適的輔導，以提高學生之學習成效，同時提供一些訊息來為程度較弱的學生開發合適的教材。

**關鍵詞**：評量指標、 教學評量、 程式設計、成果導向的教育、學習策略。

## Abstract

Programming is a very important course in the field of computer science information technology. Through years of teaching experience in programming, we observed that students learning outcomes of this course is generally poor, only about half of the students can successfully pass the course, the remaining students often need to repeat tutoring, and there's still have a small part that cannot successfully passed the course and must be revised. By checking, analyzing students' codes and interviewing students we found that they have difficulties in learning programming because of syntax error, do not completely understand the problem or cannot solve complex or mathematical related problems.
Considering of these above reasons, we feel that it is critical to find a way to evaluate students' learning outcomes that can clearly describe students' ability to analyze problem requirements, design algorithms and write code.

Recently, rubric is the most appropriate tool to evaluate students learning outcomes which not only allows teachers to assess students' abilities, but also let students know what they need to achieve in the course. In this paper, we design rubric to evaluate student learning outcomes. Based on the difficulties students encountered in learning process over the years, we designed rubric to evaluate students' learning outcomes. First we create test based on rubric, then using rubric to analyze students learning problem and finally provide appropriate training for students to improve their learning outcomes and providing some information to develop appropriate materials for weak students.
**Keywords**: Rubric, Assessment, Programming, Outcome-based education, Learning strategies.

## 1. 前言

在這個資訊技術越來越發達的時代，軟體對日常生活各方面的影響甚鉅，人們不但要會使用軟體，常常需要自己寫出所需的各種軟體。不只在台灣，常常在報紙上看到國內很缺軟體工程師，在世界各地也都普遍存在此問題，所以，各國均很重視程式設計教學，甚至從幼稚園就開始教授一些程式的基本概念，由此可知程式設計的重要性。

在大學，資訊工程系，程式設計是一門必修課。其目的為帶給學生對於程式設計有初步的認識，同時訓練學生寫程式的基本能力。在多年教學中，我們觀察到學生在此門課程的學習成效普遍不佳，大約只有半數學生可以順利一次通過測驗，剩下的學生則常需要反覆進行

多次輔導才可以通過，甚至還有小部分學生在輔導後仍無法順利通過測驗，必須重修課程。透過檢視、分析學生所寫的程式並訪談學生，我們觀察到學生常遇到的學習困難為語法錯誤、不瞭解題意或無法解比較複雜或數學相關類型的題目。綜合這些原因之後，我們覺得必須找出一個可以清楚描述學生在分析題目要求、設計解題演算法和撰寫程式碼三種能力的方法，藉以評估學生的學習成效。

為了解決上面問題，在本論文中，我們將採用rubric來評估學生學習成效。rubric是一個評分工具、透過它學生可以清楚了解老師對此課程的期待、學生應得的學習成效和能力。首先，我們分析歷年學生在學習過程所遭遇的各種困難，據此設計rubric來評量學生的學習成效。接下來，根據rubric的評量目標來建考題評量學生，我們將考試分成課前考、課後考和補考三種。然後應用rubric進行分析學生考試的結果。分析學生課前與課後解答我們發現大部分學生犯的錯誤是語法。上完課會有一部分學生可以馬上應用教過的敘述解題目(優)，另外一部分雖然可以解題目，但是有部份錯誤(中)，剩下的則為完全無法寫程式的學生(弱)。我們會徵詢程度中等和弱且有意願接受輔導的學生進行課後輔導(因為學生的學習動機為影響學生學習成效的主因，本論文暫不探討這個因素，故先以志願參加輔導者為主)。試驗結果成效良好，證明我們的方法能夠增加教學成效，同時也能提供一些訊息來為程度較弱的學生開發合適的教材。

本研究其他部分包含：相關研究、研究方法、實驗與結果和最後的結論。

## 2.相關研究

造成程式設計學習困難的原因很多，Robins 與 Boulay 在其研究中歸納一些初學者的特性與常遇到的錯誤，提供教師設計教學方法時應該注意的問題[5, 7]。Jenkins 與 Gomes 把各種學習困難的原因依教學方式、學習方式、學生的能力與動機、程式設計本身存在的難度、心理的影響清楚的加以分類[3, 4]。Gomes 發現每個學生解題時需要即時反饋與詳細解釋，但是老師可能無法及時提供這樣的服務、老師的教學策略無法適用全班學生、以靜態教材解釋動態概念或老師太專注教程式語言的語法而忽略應用程式語言解決問題的技巧[4]。Lui 應用建構主義到程式設計課程找出學習不佳的學生在認識與情感方面常遇到的問題，然後提出一些指導方式可以讓這些學生至少達到基礎水平[6]。Lahtinen 與 Milne 透過調查方式找出學生覺得哪些程式設計概念比較難，然後重新設計教材[1, 2]。所以有很多研究克服上面提到的問題，Karsten 用視覺化方式[11]或 Chen 用戲劇方式[14]，讓學生更容易理解程式設計比較抽象的概念。Kalelioğlu，Corral，Seng 與 Howland 使用視覺化程式設計語言讓學生透過玩遊戲學習程式邏輯[8, 9, 12, 13]。Gonzalez 與 Ben-Ari 採用建構主義鼓勵學生建立自己的知識[15, 16]。Brito 透過多次考試讓學生與老師提早知道學習成果，以便學生有足夠時間改善[17]。以上提到的研究，雖然得到初步的成果，但是還沒有深入研究教學原則理面的適性教學(adaptive instruction)的問題，也就是說教學過程需配合學生的能力與學習需求，作因應與導引式調整。適性教學表示在學習過程中教學內容、交給學生的任務(作業、考試)需要符合全班的特徵與個人的特徵。在教學過程中，如果可以確保適性教學就可以提升學生學習動機、讓學生對自己能力更有信心。相反，當教學內容超過學生的程度，容易讓學生對自己的學習能力產生挫折與悲觀，這些都會制約學生的學習智能。了解這個問題後，在教學過程中，我們必須關注全班的平均程度並留意個人的程度。所以，我們必須尋找一個評估學生學習成效的工具，以便即時、正確反應學生學習能力，讓我們可以提出提高學生學習能力的合適策略。因為 rubric 可以清楚描述老師要求學生在此門課需要完成的任務及學到的知識[19, 20]，在多方評估之後，我們選擇 rubric 作為評估工具。

## 3.研究方法

我們對學生學習成果的改善策略是採用教學理論中的適性教學。上課時，老師必須有效掌握學生的學習成果，尤其要隨時關注程度較差之學生的學習狀況，這樣才能及時調整教學活動，以改善學生學習成果。有鑑於此，我們必須先設計出完善有效的 rubric 評量表，作為我們評估學生學習成效的利器，根據 rubric 的評估因子來設計考題測驗學生，每當完成測驗後，我們使用 rubric 評估與分析學生的學習成效，然後，根據分析的結果，對學習成效不佳的學生以適合學生的程度進行輔導。完成輔導之後，再讓學生進行重測並檢視輔導後的結果，圖 1 為整個改善策略流程。

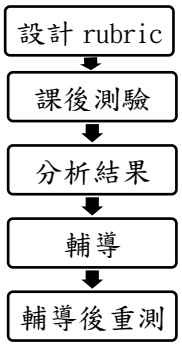


**圖 1. 改善策略**

### 3.1. Rubric 設計原則

rubric 設計首先，必須定義學生於程式設計過程需要達成的各項任務；然後，定義各項任務完成度(level)之評估因子(criteria)、各種評估因子的評分標準與完成度之分級。

#### 3.1.1.定義程式設計過程需達成之任務

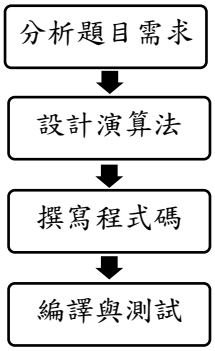


**圖 2. 程式設計過程需達成之任務**

程式設計過程需達成如圖 2 所示之四個任務，每個任務前後面均有密切關係。當一個任務沒有正確完成，都會影響到最後的結果，當順利完成四個任務則代表整個程式設計完善，故我們以逆向分析檢測各個任務的完成度，來評估學生的學習成效，以此判斷學生在學習過程中，在那一個階段遇到瓶頸，並據加以輔導提升學習成效。

當學生都是初學者，優秀學生可以自己學會與完成上面四個任務，中等學生需要老師幫一點忙，而程度較弱的學生則可能需要慢慢帶學。學生若不清楚整個過程，當程式碼有問題就不知道哪裡出問題，將無法自行修改。

分析題目需求：根據題目提供的訊息，決定輸入輸出需求

設計演算法：依題目要求設計解題演算法，包括解題邏輯、運算式、進行步驟。

撰寫程式碼：以正確有效的敘述實作出所設計的演算法。

編譯與測試：編譯時如果碰到編譯錯誤、超過時間、記憶體限制要反覆除錯修正至可執行，當程式可以執行時，必須仔細檢查測試不同輸入資料之輸出直至完全正確為止。

#### 3.1.2.定義任務完成度評估因子與評分標準

完成程式設計過程需達成任務之定義後，接下來則必須定義評估任務完成度之評估因子與評分標準，列示如表 1。

**表 1 各項任務完成度之評估因子**

| 任務項目 | 評估因子 |
|---|---|
| 分析題目需求 | 輸入與輸出分析之完成度 |
| 設計演算法 | 解題演算邏輯設計之完成度 |
| 撰寫程式碼 | 資料儲存結構定義宣告與程式碼之完成度 |
| 編譯與測試 | 編譯與測試之完成度 |

實際評分時使用的 rubric 表及評分標準，如表 2。我們將各項任務之完成度(Level)分為三個等級：

- L1：全部未完成
- L2：部分完成
- L3：全部完成

**表 2 Rubric 評量表及評分標準**

| Level / Criteria | L1 | L2 | L3 |
|---|---|---|---|
| C1-輸入輸出分析 | 輸入或輸出敘述不完整 | 輸入輸出敘述格式或輸出結果錯誤 | 完全正確 |
| C2-設計演算法 | 完全錯誤 | 部份錯誤 | 完全正確 |
| C3-資料儲存結構定義與宣告 | 完全錯誤 | 部份錯誤 | 完全正確 |
| C4-撰寫程式碼 | 完全錯誤 | 部份錯誤 | 完全正確 |
| C5-編譯與測試 | 編譯錯誤 | 編譯正確測試錯誤 | 完全正確 |

### 3.2. 課後測驗原則

為了驗證學生可否達成前述4個任務和所具備的程式能力，我們根據所設計的 rubric 設計課後考題，每回總共有 5 題：其中 2 題測驗基本語法，2 題語法進階應用測驗及 1 題測驗邏輯思考。

### 3.3. 分析測驗結果

課後測驗結束後，我們以 rubric 進行成績不及格學生的學習成效分析，明確定出學生於 4 個任務 5 個評估因子的完成度，作為設計輔導方式的依據。

### 3.4. 輔導機制

以 rubric 進行學生個人學習成效分析後，可確定學生於各任務的完成度，作為設計個別輔導方式的依據。本節將針對各任務之評估因子，依不同的完成度訂定相對應的輔導機制，一一說明如下：

- **C1-輸入輸出分析**

  L1：重新講解輸入輸出敘述、資料儲存概念(型態、變數、陣列、...等)，然後帶學生練習從題目找出輸入輸出及決定合用的資料儲存方式。確認學生做對之後，再讓學生自己反覆練習題目。

  L2：針對錯誤的部份加強，先讓學生自己試著找出錯誤並修正，若學生無法克服，則由老師講解出錯的原因，再讓學生重做，以加深印象。然後，讓學生自己反覆練習題目至精熟。

- **C2-設計演算法**

  L1：重新講解分析題目與解題相關技巧，訓練學生將題目要求完成的任務，先以紙筆列出完整解答步驟，再將之轉為虛擬碼或流程圖表示。此乃解題邏輯的訓練，需要的訓練時間較長，一旦學生熟練解題邏輯將抽像問題具像化之後，即可跨越解題轉換的門檻。讓學生自己反覆練習類似題，小幅修改保有原題目的延續性，學生較容易完成及得到成就感，可提高對程式的興趣，必須系統化設計漸進式導引教學的題庫供學生練習。

  L2：先讓學生試著自己找出錯誤並修正，若學生無法克服，才針對錯誤的部份予以協助，再讓學生重做以加深印象。然後，提供大量類似題，讓學生練習至精熟。

- **C3-資料儲存結構定義與宣告**

  L1：重新講解資料儲存結構定義與宣告的方式，並帶學生做題目，使其由解題過程理解相關概念，最後，指定相關題目讓學生練習。

  L2：針對錯誤的部份予以協助，提供大量類似題讓學生練習。

- **C4-撰寫程式碼**

  L1：重新講解如何利用 C1~C3 所準備的轉換為程式碼，要求學生先熟記各種敘述的用途和語法，並帶學生做題目，使其由解題過程理解相關概念，最後，指定相關題目讓學生練習。此部份和解題邏輯一樣，需要長時間以漸進導引式的題庫加以訓練。

  L2：針對錯誤的部份予以協助，讓學生自己能重新找出錯誤並修正，提供大量類似題讓學生練習。

- **C5-編譯與測試**

  L1：重新講解除錯工具的用法及各種錯誤訊息的意義與解決方式，再讓學生檢查並修改自己的程式碼，直到可自行解決問題為止。

  L2：教導學生如何找出所有必要測試的輸入資料，同時確保各種輸入資料對應之輸出結果正確無誤。

經多年教學的觀察，發現程式設計學不佳的學生，大都不清楚各任務所需達成的目標及方法，所以，我們的 rubric 設計 5 個評估因子，以評估學生所遇到的學習困難，由實驗中得知，學生主要的困難為設計演算法(C3)和撰寫程式碼(C4)，所以，我們以 C3 和 C4 兩者的分析為主，以減少 rubric 分析時間。

### 3.5. 輔導後重測

輔導後將進行重測以證明整個機制的有效性，重測的題數、難度與各種規定與課後測驗一樣，但題目則加以修改的類似題。

## 4. 實驗與結果

本研究的實驗對象為大學部日間部一年級 1A、1B 兩班成績不及格的學生。我們讓兩班成績不及格，但有意願接受輔導的學生自由

報名參加。分別於期中考後與期末考後兩個階段進行實驗，參加人數統計如表 3 所示。

**表 3 參加實驗人數統計**

| 班級 | 實驗階段 | 原始及格人數 | 參加輔導人數 | 輔導後及格人數 |
|---|---|---|---|---|
| C1A (59 人) | 期中 | 28 | 17 | 16 |
| | 期末 | 11 | 30 | 19 |
| C1B (64 人) | 期中 | 21 | 22 | 18 |
| | 期末 | 17 | 29 | 17 |

從表 3 可以看到兩班期末考及格人數均比期中考減少，C1A 減少 17 位，C1B 減少 4 位。及格人數減少的原因是因為期末考範圍有指標，動態陣列，函數傳遞一指標參數，這些是屬於比較難的主題，所以較多學生對這些敘述的語法與應用概念感到困難，無法順利解題。在 Lahtinen 或 Milne 與 Rowe 的研究中，提到指標的概念被學生評為程式設計最難學的概念[1, 2]。但是由表 3 可看到參加輔導人數呈現增加的狀況，代表大多數的學生還是有學習意願，只是因為他們學習能力有限，所以無法自己學習，需要透過輔導加強學習能力。

完成輔導之後，我們讓學生進行重測，以驗證輔導後的成效。圖 3 顯示 C1A 期中、期末考的原始與重測及格率比較，其中及格率=及格人數/應考總人數。

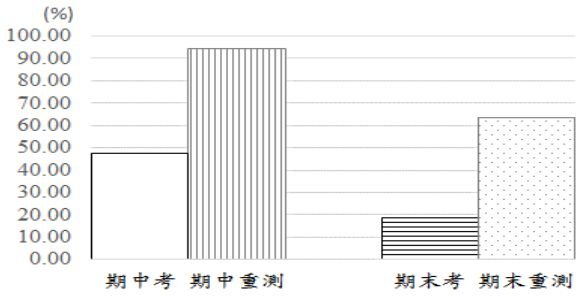


**圖 3. C1A 各測驗及格率**

C1A 期中考原始及格率為 47.46%，重測及格率為 94.12%，期末考原始及格率為 18.64%，重測及格率為 63.33%。由此可看到進行輔導後，學生學習成效均大幅提高，其中期末重測比期中重測及格率低，造成這個現象的原因，除了前面所提到的考試主題難度較高之外，另一個原因是參加輔導學生變多，造成時間、人力不夠，雖然我們找到學生學習的問題，但是，沒有充裕的時間、人力可協助每個學生克服所有的問題，在 C1B 也發生同樣的狀況，所以，這也是教學上必須思考如何改善的地方。C1B 輔導後的成效如圖 4 所示，C1B 期中考原始及格率為 32.81%，重測及格率為 81.82%，期末考

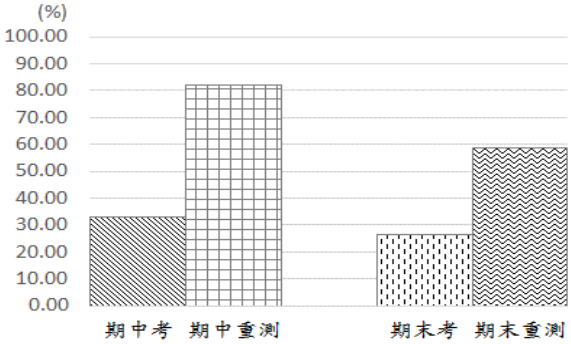


**圖 4. C1B 各測驗及格率**

原始及格率為 26.56%，重測及格率為 58.62%。

C1A 與 C1B 期中輔導成效比較如圖 5 所示，由圖 5 可以看出 C1B 原本期中考及格率比 C1A 低，期中重測也是。雖然各班學習成效有提高，但是 C1B 輔導後並沒有變的比 C1A 好，這和班級的學習風氣恰好吻合，證明學生的學習態度對學習成效有很大的影響。

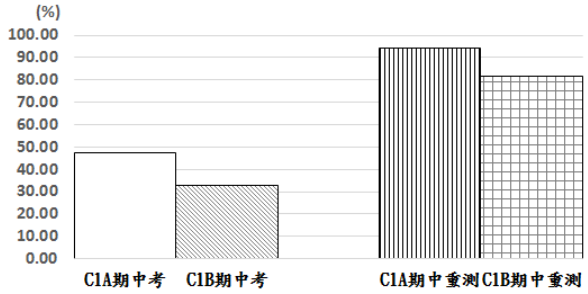


**圖 5. C1A & C1B 期中考及格率比較**

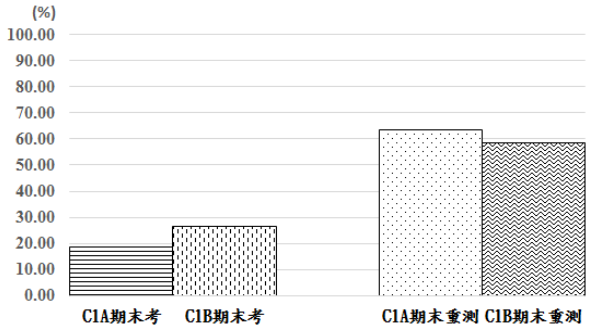


**圖 6. C1A & C1B 期末考及格率比較**

C1A 與 C1B 期末輔導成效比較如圖 6。從圖 6 可以看到，到期末階段兩班的表現與期中不一樣。C1B 期末考原始及格率比 C1A 高，但是重測時變成 C1B 比 C1A 低，應該是 C1A 再挫敗之後較積極接受輔導補救。兩班各次測驗的平均分數統計如表 4。

表 4 各考場平均分數統計

| 班級 | 考試類型 | 平均分數 |
|---|---|---|
| C1A | 期中考 | 27.65 |
| | 期中重測 | 77.06 |
| | 期末考 | 14.29 |
| | 期末重測 | 72.86 |
| C1B | 期中考 | 24.55 |
| | 期中重測 | 74.09 |
| | 期末考 | 18.72 |
| | 期末重測 | 64.53 |

## 5. 結論

在本論文中，透過實際應用 rubric 分析學生學習成效，實驗證明這個方式可以詳細分析學生目前達到的成果，讓老師、助教等可進行有效的輔導、提高學生學習效果。在輔導過程中，我們發現可以透過 rubric 分析、找出學生的問題，然後提供適合解決方法，所以學生學習動機提高且更積極練習。但是，我們發現應用 rubric 分析需要花很多時間，所以，未來必須朝自動進行分析的方式努力，例如：設計演算法時，可採用流程圖表示，設計好流程圖元件，讓學生如組裝積木的方式將流程圖組合起來。

## 參考文獻


[1] Lahtinen, Essi, Kirsti Ala-Mutka, and Hannu-Matti Järvinen, "A study of the difficulties of novice programmers," Acm Sigcse Bulletin, Vol. 37, No. 3, 2005.

[2] Milne, Iain, and Glenn Rowe, "Difficulties in learning and teaching programming-views of students and tutors," Education and Information technologies, Vol. 7, No. 1, p.55-66, 2002.

[3] Jenkins, Tony, "On the difficulty of learning to program," Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences, Vol. 4, 2002.

[4] Gomes, Anabela, and António José Mendes, "Learning to program-difficulties and solutions," International Conference on Engineering Education–ICEE, 2007.

[5] Robins, Anthony, Janet Rountree, and Nathan Rountree, "Learning and teaching programming: A review and discussion." Computer science education, Vol. 13, No. 2, pp.137-172, 2003.

[6] Lui, Andrew K., "Saving weak programming students: applying constructivism in a first programming course," ACM SIGCSE Bulletin, Vol. 36, No. 2, pp.72-76, 2004.

[7] Du Boulay, Benedict, "Some difficulties of learning to program," Journal of Educational Computing Research, Vol. 2, No. 1, pp.57-73, 1986.

[8] Kalelioğlu, Filiz, "A new way of teaching programming skills to K-12 students: Code. Org," Computers in Human Behavior, Vol. 52, pp.200-210, 2015.

[9] Corral, José María Rodríguez, "A game-based approach to the teaching of object-oriented programming languages," Computers & Education, Vol. 73, pp.83-92, 2014.

[10] Hundhausen, Christopher D., and Jonathan Lee Brown, "An experimental study of the impact of visual semantic feedback on novice programming," Journal of Visual Languages & Computing, Vol. 18, No. 6, pp.537-559, 2007.

[11] Karsten, Rex, and Shashidhar Kaparthi, "Using dynamic explanations to enhance novice programmer instruction via the WWW," Computers & Education, Vol. 30, No. 3, pp.195-201, 1998.

[12] Seng, Wong Yoke, and Maizatul Hayati Mohamad Yatim, "Computer game as learning and teaching tool for object oriented programming in higher education institution," Procedia-Social and Behavioral Sciences, Vol. 123, pp.215-224, 2014.

[13] Howland, Kate, and Judith Good, "Learning to communicate computationally with Flip: A bi-modal programming language for game creation," Computers & Education, Vol. 80, pp.224-240, 2015.

[14] Chen, Lim Fung, "The effectiveness of using role play in conducting programming classes," International Journal of Asian Social Science, pp.2074-2083, 2013.

[15] Gonzalez, Gracielo, "Constructivism in an introduction to programming course," Journal of Computing Sciences in Colleges, Vol. 19, No. 4, pp.299-305, 2004.

[16] Ben-Ari, Mordechai, "Constructivism in computer science education," Journal of Computers in Mathematics and Science Teaching, Vol. 20, No. 1, pp.45-74, 2001.

[17] Brito, Miguel A., and Filipe de Sá-Soares, "Computer programming: fail fast to learn sooner," Technology Enhanced Learning. Quality of Teaching and Educational Reform. Springer Berlin Heidelberg, pp.223-229, 2010.

[18] Becker, Katrin, "Grading programming assignments using rubrics," ACM SIGCSE Bulletin, Vol. 35, No. 3, 2003.

[19] Jonsson, Anders, and Gunilla Svingby, "The use of scoring rubrics: Reliability, validity and educational consequences," Educational research review, Vol. 2, No. 2, pp.130-144, 2007.