PROCEEDINGS OF THE
44TH ANNUAL SOUTHERN AFRICAN COMPUTER
LECTURERS ASSOCIATION 2015
(SACLA 2015)

“Renewing ICT teaching and learning:
Building on the past to create new energies”

1-3 July 2015
Johannesburg
South Africa

Editor:
Emma Coleman

Publisher:
University of the Witwatersrand, Johannesburg
Private Bag X3
WITS
2050

REFEREE PROCEDURE

The annual conference of the Southern African Computer Lecturers Association (SACLA) presents lecturers in tertiary departments of Computer Science and Information Systems with an opportunity to share experiences of teaching from undergraduate to doctoral levels. The theme of SACLA 2015 is ‘Renewing ICT teaching and learning: building on the past to create new energies’.

The research papers included in the PROCEEDINGS OF THE 44th ANNUAL SOUTHERN AFRICAN COMPUTER LECTURERS ASSOCIATION CONFERENCE (SACLA 2015) were each double-blind peer reviewed by at least two members of the programme committee. The program committee consisted of both local and international experts in the fields of computer science and information systems education, and having expertise and interest in subjects relevant to the theme of the conference.

All papers were initially screened by the programme chairs for fit to the SACLA conference. Papers were then sent for peer review. A total of 61 academics from 14 local and international institutions constituted the programme committee and reviewed papers for the conference.

The review process followed the double-blind peer review model. Every paper received at least two reviews. The programme chairs solicited additional expert reviews in cases where further clarity was warranted.

Only original, previously unpublished, research papers in English were considered and papers were reviewed according to South Africa’s Department of Higher Education and Training (DHET) refereeing standards. Papers were reviewed according to the following criteria:

- Relevance of the paper to the conference theme
- Originality of the research contribution
- Technical/scientific merit of the research
- Presentation and clarity of the paper

Before accepting a paper, authors were to include the corrections as stated by the peer reviewers. The double-blind review process was highly selective. Of the 55 academic papers received for consideration, 27 papers were accepted for inclusion in the Proceedings after the required changes were made. This constitutes a 49% acceptance rate of contributed papers. The papers accepted cover a wide range of relevant topics within the conference theme, and are reproduced within these proceedings.

Dr Emma Coleman, Prof Judy Backhouse, Prof Jason Cohen
The Program Chairs: SACLA 2015
July 2015

School of Economic and Business Sciences
University of the Witwatersrand, Johannesburg
South Africa
Tel: +27 (0)11 717 8160 Fax: +27 (0)11 717 8139
SACLA 2015 COMMITTEE

SACLA 2015 is hosted by the Division of Information Systems, School of Economic and Business Sciences at the University of the Witwatersrand, Johannesburg.

- Emma Coleman (Conference Chair and Programme Co-Chair)
- Judy Backhouse (Programme Co-Chair and Workshop Coordinator)
- Jason Cohen (Programme Co-Chair and M&D Session Coordinator)
- Suzanne Sackstein (Organising Committee Chair)
- Mitchell Hughes (Organising Committee and HOD Colloquium Coordinator)
- Thomas Grace (Conference Webmaster)
- Nalukui Malambo (Organising Committee)

PROGRAM COMMITTEE SACLA 2015

1. Adele Botha (CSIR)
2. Anitta Thomas (UNISA)
3. Arash Raeisi (University of Salford, UK)
4. Beula Samuel (UNISA)
5. Carina De Villiers (University of Pretoria)
6. Colin Pilkington (UNISA)
7. Conrad Mueller (UNISA)
8. Desiree Hamman-Fisher (University of the Western Cape)
9. Elmarie Kritzinger (UNISA)
10. Elsje Scott (University of Cape Town)
11. Emma Coleman (Wits University)
12. Guillaume Nel (Central University of Technology)
13. Helene Gelderblom (University of Pretoria)
14. Henk Pretorius (University of Pretoria)
15. Hima Vadapalli (Wits University)
16. Hugo Lotriet (UNISA)
17. Imelda Smit (North West University)
18. Jaco Pretorius (University of Pretoria)
19. Jan Kroeze (UNISA)
20. Jane Nash (Rhodes University)
21. Jason Cohen (Wits University)
22. Jean-Marie Bancilhon (Wits University)
23. Jean-Paul Van Belle (University of Cape Town)
24. Judy Backhouse (Wits University)
25. Judy van Biljon (UNISA)
26. Karen Renaud (University of Glasgow)
27. Ken Halland (UNISA)
28. Ken Nixon (Wits University)
29. Kerry-Lynn Thomson (Nelson Mandela Metropolitan University)
30. Laurette Pretorius (UNISA)
31. Laurie Buttgerieit (Nelson Mandela Metropolitan University)
32. Leda van der Post (Nelson Mandela Metropolitan University)
33. Liezel Nel (University of the Free State)
34. Linda Marshall (University of Pretoria)
35. Linda Spark (Wits University)
36. Lisa Seymour (University of Cape Town)
37. Lynn Futcher (Nelson Mandela Metropolitan University)
38. Mac van der Merwe (UNISA)
39. Machdel Matthee (University of Pretoria)
40. Mariana Gerber (Nelson Mandela Metropolitan University)
41. Marlien Herselman (CSIR)
42. Martina Jordaan (University of Pretoria)
43. Maureen Tanner (University of Cape Town)
44. Mitchell Hughes (Wits University)
45. Nompilo Tshuma (Rhodes University)
46. Patient Rambe (University of the Free State)
47. Patricia Lutu (University of Pretoria)
48. Ray Kekwaletse (Wits University)
49. Reuben Dlamini (Wits University)
50. Romeo Botes (North West University)
51. Ronell van der Merwe (UNISA)
52. Roxanne Piderit (University of Fort Hare)
53. Susan Benvenuti (Wits University)
54. Suzanne Sackstein (Wits University)
55. Thomas Grace (Wits University)
56. Thomas van der Merwe (UNISA)  
57. Trish Alexander (UNISA)  
58. Turgay Celik (Wits University)  
59. Vincent Horner (UNISA)  
60. Wallace Chigona (University of Cape Town)  
61. Walter Uys (University of Cape Town)  

SPONSORS

SACLA 2015 was made possible through the generous sponsorship of our partners.

- SAP
- Oracle
- AOS
- Deloitte
- KPMG
- School of Economic and Business Sciences, Wits University
CONTENTS – PEER REVIEWED PAPERS

Guillaume Neel and Liezel Neel, Improving Program Quality: The Role of Process Measurement Data ........................................ 1
Helene Gelderbloom and Alta van der Merwe, Applying Human-Computer Interaction (HCI) Design Principles and Techniques in HCI Course Design ............................................................................. 7
Lindokuhle Gomuna, Lynn Futcher and Kerry-Lynn Thomson, Integrating Information Security into the IT Undergraduate Curriculum: A Case Study ........................................................................ 19
Thabang Serunola and Lisa F Seymour, Factors Affecting Students Changing Their Major to Information Systems .................................................................................................................. 27
Andre Calitz, Margaret Cullen and Jean Greyling, South African Alumni Perceptions of the Industry ICT Skills Requirements ................................................................................................................. 36
Ruth Wario and Marianne Viljoen, Attitudes Towards Computer Usage as Predictors of the Classroom Integration of ICT at a Rural South African University ......................................................... 48
Patient Rambe, Technology for Interactive Engagement or Tools for Differential Academic Participation? Using Google Groups for Collaborative Learning at a South African University .................................................. 57
Laurie Butgereit, An Experiment in Using Gamification in an Information Technology Distance Classroom ................. 69
Jane Nash, Using Video Podcasts to Teach Procedural Skills to Undergraduate Students ................................................. 77
Judy Backhouse and Mitchell Hughes, An Ecological Model of the Information Behaviour and Technologies of Undergraduate Students in a South African University .............................................................. 85
Baldreck Chipangura, Judy van Biljon and Adele Botha, Evaluating Mobile Centric Readiness of Students: A Case of Computer Science Students in Open-Distance Learning ........................................ 92
Walter F. Uys and Wallace Chigona, Strengthening the Teaching-Research Nexus in a Large First Year Information Systems Class .................................................................................................................. 100
Ian Sanders, Colin Pilkington and Wynand van Staden, Errors Made by Students when Designing Finite Automata ........................................................................................................................................... 110
Laurette Marais and Laurette Pretorius, Precise Machine Translation of Computer Science Study Material ............... 119
Janet Liebenberg, Pair Programming in the Introductory Programming Class ................................................................. 129
Ken Halland, Anitta Thomas and Colin Pilkington, Teaching OOP Using the Qt Framework ............................................ 137
McDonald van der Merwe, Automated Language Processing and Semantic Classification: A Preliminary Effort to Examine the Relationship between Distance Student Online Discussion Forum Discourse and Learning Outcomes ............................................................................................................. 144
Craig Marais and Karen Bradshaw, Problem-Solving Ability of First Year CS Students: A Case Study and Intervention ............................................................................................................................................ 154
Vreda Pieterse and Lisa Thompson, Investigating the Applicability of Belbin Roles on Participatory Levels in IT Student Teams ...................................................................................................................... 161
Nompilo Tshuma, Assessment for Learning: Using Formative Assessment to Scaffold Students’ Fragile Knowledge .................................................................................................................................................. 170
Judy van Biljon and Karen Renaud, Do Visualizations Ease Dissertation Assessment? .................................................... 177
Suzanne Sackstein, Linda Spark and Tana Masenda, E-textbook Usage by Lecturers: A Preliminary Study ............... 186
Thomas Basirat Abinbola, Botes Elizabette Pranto, Imelda Smit and Roelien Goede, How Can We Help You? Identifying High Risk 1st Year IT Extended Students .............................................................................. 191
Romeo Botes and Imelda Smit, To NoSQL or Not to NoSQL: That is the Question .......................................................... 196
Patricia Lutu, Big Data and NoSQL Databases: New Opportunities for Database Systems Curricula ....................204
Vreda Pieterse and Hannes Janse van Vuuren, Experience in the Formulation of Memoranda for an
Automarker of Simple Programming Tasks .................................................................................................210
Stefan Gruner, On the Future of Computer Science in South Africa: A Survey amongst Students at University ..215
Experience in the Formulation of Memoranda for an Automarker of Simple Programming Tasks

Vreda Pieterse
Department of Computer Science
University of Pretoria
Pretoria, South Africa
vpieterse@cs.up.ac.za

Hannes Janse van Vuuren
Department of Computer Science
University of Pretoria
Pretoria, South Africa
hjvvuuren@cs.up.ac.za

ABSTRACT
We have gained experience over the past ten years while teaching an introductory programming module where the assignments are automatically assessed. Our experience of using automatic assessment improved our skills in setting clearer assignments of a higher quality, which are easier to assess. We use an example to describe how we set an assignment. This example assignment and its memorandum were the starting point for the design of a new memo format for our automarker. The strategies used for setting the assignment were developed for setting assignments suitable for automatic assessment. We explain how these strategies can be used for setting an assignment and formulating the instructions for its manual assessment. This is a guideline for creating quality assignments of this kind.

Categories and Subject Descriptors
K.3.2 [Computers and Education]: Computer and information Science Education; K.3.1 [Computers and Education]: Computer Use in Education.

General Terms
Human Factors

Keywords
Programming assignment, assessment, automatic assessment

1. INTRODUCTION
The Programming assignments have been assessed automatically for almost the same length of time as programming has been taught. Automated assessment is employed in many programming courses at tertiary education institutions [6, 10, 12].

In February 2016 we will celebrate the tenth anniversary of our own system, called Fitchfork, which we use for the automatic assessment of programming tasks. When assessing a program, Fitchfork compares the output produced by a student’s program with a specified expected result.

Each program submitted is executed several times, each time with different test data as stipulated in a memorandum.

Regular expressions are used for the specification of the expected output for each test case. Students submit their programs by uploading via a web interface and receive their feedback on the same interface.

The quality of the assessment and feedback given as a result of the assessment depend on the rubric used for the assessment. The formulation of assessment instructions is an art, whether the assignment is intended to be assessed using an automarker such as Fitchfork or intended to be assessed manually. In this paper we present a tutorial on how to create quality programming assignments. To this end, we describe the design of an assignment and the specification of its assessment rubric. The rubric is aimed at automatic assessment but could easily be formulated for manual assessment.

Section 2 describes a sample assignment as a means of illustrating the different aspects of a rubric for assessment. The formulation of rubrics and memoranda is closely linked to the principles of software testing. In Section 3 we illustrate how testing theory informs the way we design a test suite for a memo. Section 4 emphasises feedback as an important aspect of formative assessment and indicates how it should be incorporated in a memorandum for automatic assessment. An often neglected aspect of assessment is balancing the difficulty level of questions with the allocation of marks to ensure fairness. In Section 5 we describe our rule of thumb for doing this. Table 2 is the outcome of a typical balancing act which we usually apply before setting a memo for automatic assessment. Section 6 gives an example of how this is translated into assessment instructions for the manual evaluation of the assignment described in Section 2.

2. THE ASSIGNMENT
Assessment is intended to measure the students’ competence with specific programming concepts. When one has to set an assignment to achieve this goal, one has to identify the learning outcomes, formulate the tasks aimed at testing the learning outcomes, select a theme and design an assignment based on these elements.

Our sample assignment is designed to provide experience in the manipulation of C-String type arrays – null terminated
character arrays. Typically one would require the students to be able to use an appropriate variable to hold a C-String, to initialise the variable with a value which is read from the keyboard, and to apply some of the functions listed in Table 1. The assignment should also reinforce the definition and use of user-defined functions.

Programming the word game called Pig Latin requires the student to use the above-mentioned skills, making it a suitable theme for the assignment.

3. TESTING

It is important for students to know how to test the functional correctness of their programs. To support this, Basu et al. [3] provide test suites and require their students to prove their understanding of the purpose of a test case before allowing them to apply it. Students with a better understanding of testing are better equipped to accept the results from an automarker [9]. For this reason, we teach our students techniques for designing test cases. We apply these techniques when designing our test suites for assessing a program.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool isalpha(char)</td>
<td>Returns true if the argument is a letter of the alphabet and false otherwise.</td>
</tr>
<tr>
<td>bool isupper(char)</td>
<td>Returns true if the argument is an uppercase letter and false otherwise.</td>
</tr>
<tr>
<td>bool islower(char)</td>
<td>Returns true if the argument is a lowercase letter and false otherwise.</td>
</tr>
<tr>
<td>char toupper(char)</td>
<td>Returns the uppercase equivalent of its argument.</td>
</tr>
<tr>
<td>char tolower(char)</td>
<td>Returns the lowercase equivalent of its argument.</td>
</tr>
<tr>
<td>int strlen(char[])</td>
<td>Returns the number of characters in its argument.</td>
</tr>
<tr>
<td>void strcat(char[], char[])</td>
<td>Appends the second argument to the first argument.</td>
</tr>
</tbody>
</table>

**Assignment**

Pig Latin is a constructed language game. English words are altered according to a set of rules to conceal their meaning. The reference to Latin is a deliberate misnomer, as it is used only for its English connotations as a foreign-sounding language. The rules for changing standard English into Pig Latin are as follows:

- For words that begin with a consonant, the initial consonant or consonant cluster is moved to the end of the word, and “ay” is added, as in the following examples:
  - “happy” → “appyhay”
  - “duck” → “uckday”
  - “glove” → “oveglay”

- For words that begin with a vowel, “way” is added at the end of the word, as in the following examples:
  - “egg” → “eggway”
  - “inbox” → “inboxway”
  - “eight” → “eightway”

The following example is a test run of the program:

Enter English word: happy
The Pig Latin for happy is appyhay

**Specifications**

- Write a function called isVowel. It should have one argument of type char. It should return true if the argument is a vowel and false otherwise.
- Write a function with the following prototype
  
  ```c
  void toPigLatin(const char[], char[])
  ```

  The first parameter is the input. After calling the function, the second parameter should contain the Pig Latin of the given word.

- Write a program that uses the above functions. It should prompt the user for a word and convert the given word into Pig Latin. It should accept a word in mixed case and output it in Pig Latin in lower case only.

  - The program should produce an error message if the user input is not a character string which contains only alphabetical characters.

  We usually include a sample test run of the required program when specifying an assignment. This sample uses limited test cases. We expect the students to design and apply their own test suites before submitting their assignments. If their own test suite does not cater for certain cases which have been included in the test suite used in our memorandum, the feedback when failing such cases should guide the students to identify the missing cases in their own test suites.

  The test cases used in the assessment have to be well thought out so that the assignment can fairly measure the competencies it is intended to measure. The quality of the assessment relies heavily on the effectiveness of the test suites applied when testing the students’ programs.
Table 2 describes a test suite for our example assignment. There is no need to have test cases with words starting with more vowels, as the program is not required to behave differently in such cases.

4. FEEDBACK
Assessment is integral to the learning process. When applied to assessing the students’ progress it is called summative assessment. When assessment is applied to support learning, it is called formative assessment. Formative assessment requires a feedback cycle [3, 14]. Feedback helps students to discover errors in their programs and to learn from their mistakes. When students are allowed multiple submissions, as can be arranged when using an automarker, they may revise their submissions to reinforce this learning [11].

The accuracy and usefulness of feedback rely on the accuracy of the test cases as well as on the way the person who assesses or sets the memorandum for an automarker formulates the feedback. The theories regarding the error messages produced by software provide useful ideas that should be used when formulating the feedback.

<table>
<thead>
<tr>
<th>Test case</th>
<th>Basic</th>
<th>Quality</th>
<th>Extreme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluate the input prompt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prompt has expected keyword</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Prompt ends with colon and space</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluate the bool isVowel(char) function</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>argument is a vowel in lowercase</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>argument is a vowel in uppercase</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>argument is a consonant in lowercase</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>argument is a consonant in uppercase</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>argument is not an alphabetical character</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>argument is the null character</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluate translation to Pig Latin</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>input is a word starting with a vowel</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>input is a word starting with a single consonant</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>input is a word starting with a consonant cluster of two letters</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>input is a word starting with a consonant cluster of three letters</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>input is a word starting with a consonant cluster of seven letters</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluate conversion to lower case</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>input is a word in lowercase</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>input is a word in uppercase</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>input is a word in lower case with the first letter in uppercase</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>input is a word in lower case with some letters in uppercase</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluate the error message</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>input is a word containing non-alphabetical characters</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>input is an empty string</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>20</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Percentage of total</td>
<td>57.1%</td>
<td>28.6%</td>
<td>14.3%</td>
</tr>
</tbody>
</table>

5. BALANCE
The criteria for the effective assessment of the identified outcomes by using the selected tasks involve weighting the importance of tasks, estimating their difficulty and balancing the marks allocated for a fair assessment of the specific outcomes in question. This is not a trivial task. A well-founded framework should be used. One of the most familiar educational frameworks of this kind is the *Taxonomy of Educational Objectives* commonly referred to as Bloom’s taxonomy [Error! Reference source not found.]. Anderson [Error! Reference source not found.] explains that such a taxonomy is an important instrument for the preparation of assessment tasks. Ideally each examination, test and task should be evaluated and balanced by using the now widely accepted revised Taxonomy for Learning, Teaching, and Assessing by Anderson et al. [Error! Reference source not found.] or the equally applicable and more modern SOLO taxonomy [Error! Reference source not found.]. Johnson and Fuller [Error! Reference source not found.] acknowledge its use as a way of ensuring that assessment strikes the right balance between the rote learning of content and high-level skills such as synthesis and evaluation, whereas Scott...
[Error! Reference source not found.] blames an imbalance in the levels of Bloom’s taxonomy in tests and assignments for the bimodal distribution of frequencies versus test scores often observed in computer science tests.

Instead of using a comprehensive framework such as the above-mentioned frameworks, we find it adequate to evaluate the difficulty level tested in each test case, using the following three levels:

**Basic** Students who have a basic understanding and application of the specific learning criteria for the task, should pass the test.

**Quality** Students who apply general quality requirements without an explicit instruction to do so, should pass the test.

**Extreme** Students who have a deeper understanding and implement robust programs capable of dealing elegantly with extreme tests should pass the test.

Even with the best intentions, such classification remains highly subjective. It is depends on many factors, such as how the task was formulated, the amount of guidance the students received before being given the task, the level of assistance students received while programming their solutions and the current competency level of the students.

As a rule of thumb, at least 50% of the marks should be allocated to basic cases and not more than 15% of the marks should be allocated to extreme cases. This can be achieved by adjusting the weight assigned when passing each of the test cases. Table Error! Reference source not found. shows how we classified the test cases for our sample assignment in the above-mentioned levels. The values indicate the marks to be allocated when passing each of the test cases to ensure that the assessment of this assignment complies with our rule of thumb.

The rule of thumb may be adjusted. It is acceptable to strike a balance over a number of assignments instead of balancing each assignment individually.

### 6. MEMORANDUM

The rubric shown in Table Error! Reference source not found. was compiled with the intention to use it to formulate a memo for Fitchfork. When doing so, the program is executed for each test case. In forthcoming research we will apply it to guide the design of a new schema for the formulation of memos for Fitchfork. Here it is used as a guide to compile the instructions for manual assessment shown in Table Error! Reference source not found.

### Table 3. Instructions for manual assessment

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompt is clear, ends with colon and space and expects user input on the same line.</td>
<td>3</td>
</tr>
<tr>
<td>bool isVowel(char) function returns the expected result for uppercase and lowercase alphabetical characters as well as for non-alphabetical characters and the null character.</td>
<td>10</td>
</tr>
<tr>
<td>void toPigLatin(const char[],char[]) function has the expected postcondition when called</td>
<td>13</td>
</tr>
<tr>
<td>with a word starting with a vowel and with words with 1, 2, 3 or more consonants.</td>
<td></td>
</tr>
<tr>
<td>The output should be given in lower case if input is lower case, upper case or mixed case.</td>
<td>7</td>
</tr>
<tr>
<td>An error message is displayed when a word containing non-alphabetical characters or empty input is given. It should be descriptive of the error and echo the input.</td>
<td>2</td>
</tr>
<tr>
<td>Total</td>
<td>35</td>
</tr>
</tbody>
</table>

7. Conclusion

We have come a long way in refining the system and the methods we use for designing programming assignments so that they can assess the competencies of our students efficiently and accurately.

As a first step in addressing the deficiencies of Fitchfork [7], a decision was taken to redesign the schema for the formulation of its memoranda. This paper is the outcome of preliminary work needed to illustrate the viability of a new schema.

This article showcases an assignment. We give details of how it was developed. We describe the grounds for our decisions during its development. We designed a rubric aimed to be translated into a memo for Fitchfork. Although the design of the assignment is intended for use in our automatic assessment environment, we deem it a good strategy that could be applied if the work is assessed manually. We show how the rubric for automatic assessment can be transformed to serve as a rubric for manual assessment.

The need to gear our assignments towards automatic assessment led to the development of strategies to create assignments which are clear and assessment instructions which are clear. We share this experience in the hope that it may inspire other educators to be more conscious of how they assess. We also hope they can reap the benefits of our experience to renew their assessment in the same way as we constantly aspire to do ourselves.

8. REFERENCES


[5] Bloom, B. A taxonomy of educational objectives: Opening remarks of BS Bloom for the meeting of examiners at


