The Journal of Computing in Small Colleges

Proceedings of the First Annual Central Plains Small College Computing Conference

March 31 - April 1, 1995, Southwest Baptist University, Bolivar, Missouri

Proceedings of the Sixth Annual South Central Small College Computing Conference

April 7-8, 1995, Louisiana State University in Shreveport, Shreveport, Louisiana

Edited by

John G. Meinke
University of Maryland, European Division

Volume 10, Number 5 May 1995

SHOULD COMPUTER SCIENCE MAJORS KNOW HOW TO WRITE AND SPEAK?

Marcin Paprzycki
Department of Mathematics and Computer Science
University of Texas of the Permian Basin
Odessa, TX 79762
paprzycki_m@gusher.pb.utexas.edu

Janusz Zalewski
Department of Computer Science
Embry-Riddle Aeronautical University
Daytona Beach, FL 32114
zalewski@db.erau.edu

Abstract: This paper deals with the problem that computer science majors, upon completion of their education are typically missing two important skills: the knowledge how to write technical reports and the ability to make professional presentations. This observation is based on the authors' in-class experience with teaching computer science. An approach to alleviate this problem is suggested and discussed.

1. INTRODUCTION

What is the worst nightmare of a typical student who is working on a degree in Computer Science? It can be surprising to some but it is not the necessity of taking courses in mathematics (e.g. at UTPB the whole Calculus sequence is required) or writing extremely complicated programs in C or Ada. There are two more horrifying experiences: to write a technical report and to speak in front of the class. Even the best students with straight A's from all courses have big problems with writing a quality technical report and are frightened when asked to make a presentation. The latter problem does not disappear even when the student took the Speech course (which is a part of the General Education Core Curriculum at UTPB).

The above mentioned facts are a big problem for two major reasons. First, any student who decides to enter graduate school will be (sooner or later) forced to write a technical report (if only the dissertation) and make presentations (if only in front of the examination committee). For such a student to learn how to do it right at that time may be a little bit too late. Second, most of the students who will enter the work force will be working in a corporate environment where communication skills, in particular, the quality of prepared reports and presentations may have an important bearing on their salary, promotion or even on their future with a given company. In this paper we would like, first, to summarize the most typical problems that we observed when students were asked to write reports and make presentations and, second, to propose how we should approach this problem without relying on technical writing and speech courses. And finally, we would like to share the particular proposals of topics that can be used for the in-class requirements.

2. BASIC PROBLEMS ENCOUNTERED

2.1 Writing Reports

There is a number of problems that were observed when students were required to prepare a technical report. First, let us point out, that over and above all the problems described in this section a large number of our students has major problems with writing in English. Grammar and spelling leave a lot to be desired in most cases (which is especially troublesome in the latter case as students pretend that they do not know what a spell-checker is). However, we will skip these problems assuming that they can be dealt with on an individual basis. The problems we would like to address vary somewhat between the two types of reports that we asked students to write: literature summary reports and technical reports based on an experimental study.

Appendices 1 and 2 contain two examples of literature summary type reports that were actually assigned in two courses we taught (Real-Time Systems and Programming Languages). They are written slightly differently, but they require students to, first, go to the library to collect material, second, read the literature found and, third, summarize what they have found in writing. The basic problems encountered were:

(a) performing extensive literature search in the library -- usually students can find pertinent books, but a search through the journals is a major problem; they also have some problems with retrieving information using technology-based tools like microfilms or CD's;

(b) reading the collected material -- the major problem here is how to read everything in a short time that they have given themselves (usually near the deadline);

(c) selecting material for the presentation -- the major problem is how to squeeze a lot of material into a relatively short paper they will be writing; more generally the problem is with finding the thesis of the paper that they will be pursuing and collecting the materials to support this thesis;

(d) writing a structured paper with a definite thesis -- very often what is written is a collection of sentences and thoughts that are somehow pertinent to the subject.

The second type of report is a technical report based on experimental results. Appendices 3-6 contain examples of actual assignments from the Data Structures, Programming Algorithms and Programming Languages courses. The assignments vary in details but in each case students are asked to either use a code developed earlier or implement a certain algorithm (the very fact of implementing the method does not have bearing on the grade). Students perform a number of timing experiments varying the one or more factors that are expected to influence the program's performance. These activities are followed by the most important part of the project -- writing a report summarizing their findings. In each case, the format of the report is discussed before the assignment. Students are informed that their papers are supposed to consist of the following sections: Introduction, Methodology, Experimental Results and their Analysis, Conclusion, Bibliography. They are also informed that paper should be prepared using *Information for the Authors* from any computer science related academic journal. They are also referred to these journals and papers published in them when additional questions about the structure of the paper occur. Let us first observe that students typically have no problems with implementing algorithms and performing the experiments (as long as the sequence of the experiments is precisely specified). What they do have problems with includes:

(A) problem formulation -- quite often we tried to leave students some leeway in setting the experiments up (e.g. decide what should be the input sizes to experiment with - see Appendix 6); students find these circumstances extremely

discomforting; they prefer the problem to be fully defined so they can just go and do what they are told;

(B) representation of the experimental results – what should be used (table or graph), what should be represented in tables and/or graphs;

these are directly related to the following two (more important) problems:

(C) recognizing what the major problem to be studied is -- students know what to do but have major problems with verbalizing why they do what they do (and thus they have problems with selecting the appropriate results and their representation to describe their findings);

(D) analysis of the results -- since students have doubts what and why they study and since find it difficult to select the appropriate data and the form of its representation they have trouble with analyzing experimental results; the major problem here is to describe what that they see and why this is important.

In summary the basic problems are related to formulating the research hypothesis and, after collecting data, writing a paper supporting or rejecting this hypothesis. The missing capabilities are those of critical thinking and the putting thoughts in an organized form.

2.2 PROBLEMS WITH PRESENTATIONS

The problems with presentations are in some sense quite similar to those with the papers, but also some new problems appear. They can be divided into two categories: problems with the content and problems with the medium. The former include:

(i) What is it that is to be presented? -- What is(are) the major point(s) that the audience should get out of the presentation,

(ii) How to get the message across? -- How to select that material to be presented? What to present and what to omit? What level of detail is digestible by the audience?

The basic problems with the form of the presentation are:

(iii) How to prepare readable overheads? -- How small the print can get before it becomes unreadable? How much material should be presented on one overhead?

(iv) How to manage the presentation? -- How to manage time? How to manipulate the overhead projector: where to stand; how to point to things; where to put the used overheads?

In other words, the problems with making a professional presentation are related, first, to the substance of the presentation: selecting the message to be contained in the presentation, and then selecting the material to get this message across and, second, to the mechanics of the presentation. It should be added that at least the latter part of the problem is common not only among our students but also among faculty members at various conferences.

3. INTRODUCTION OF WRITING AND SPEAKING INTO THE CURRICULUM

The persistence with which the problems occurred in our practice forces us to conclude that we cannot fully rely on special courses offered by other departments to teach our students what we want them to know. Even if a student takes a course in Speech or Technical Writing it does not guarantee that the student will be proficient in either activity. What they will gain is, most likely, some theoretical knowledge about the subject and a minimal practical experience. What is necessary to overcome the problem is a global modification of our approach to teaching. We would like to suggest that we should introduce the writing and

speech components across the CS curriculum wherever appropriate. This is the only way we can hope to achieve the ultimate goal of students being able to write technical papers and make professional presentations. Below we describe how this goal is achieved in various courses taught at UTPB

A) Computer Science I and II -- only a minimal chance for report writing exists in these two courses. At the same time the experiments with matrix multiplication (Appendix 3), are sometimes used here. In addition, since the team-work is introduced in CS II students are asked to write short reports based on their team-work and to make team- presentations.

B) Information System Design (Software Engineering) -- this course contains the software design project (based on the team-work) as its major component, the report writing and team presentations activities appear naturally there (see also [3] for more details).

C) Digital Computer Organization (Hardware) -- a literature-summary based report (similar to the project from the Programming Languages course - Appendix 2) is required here.

D) Data Structures -- this is the first course in which students are introduced to writing experiment- based reports (Appendices 3-6 contain examples of projects that could be used). Due to the size of the class no individual presentations are typically required.

E) Programming Languages -- a literature summary project is required (Appendix 2) as well as experimental studies that compare two programming languages (similar to the project in Appendix 4); in-class presentations are used from time to time.

F) Programming Algorithms -- most of the homework consist of experimental projects; there is an in-class presentation requirement (based on a semester-long research project); the research project itself is to be prepared as a technical report. It should be noted that some of the questions on the final exam are based on student presentations -- this forces the students to be well prepared during their presentation as other students know that the primary way to be prepared for the final exam is to listen and ask questions.

Similar projects can be used in other courses and we will leave it to the readers' invention to think of appropriate examples (see also [1] and [2] for more examples). The crucial point here is that there is a uniformity of requirements throughout the curriculum. From the very beginning students are told that in most of the CS courses they will be required to write research papers and/or make professional presentations. Before each project is assigned, either a detailed description of what is expected is given to the students (Appendices 1 and 2) or the basic ideas behind writing technical papers are presented in the context of the project assigned. In the latter case we discuss with the students what they are to study, what the major hypothesis they will be testing is, and what kind of results they can expect and what effect can this have on the data representation. The results of this approach are quite encouraging. Obviously, early attempts at report writing (even with the help mentioned above) and/or presentations are usually of rather low quality. This is especially true in case of experimental reports with which students have particular difficulties. To help we typically try to explain what their mistakes were and give them a chance to rewrite the paper at least once (this is especially important when they are just starting to work on the reports). A definitive improvement of the quality of students' work can be observed as they take consecutive courses.

4. CONCLUSION

It is our belief that the areas of technical writing and professional speaking are extremely important for the graduating students. It was observed that currently CS graduates are seriously impaired in these aspects of their professional preparation. We argued that these deficiencies can be overcome without introducing new courses, rather through the modification

of the existing courses to include appropriate exercises. Obviously, the proposed modification increases the demand on the teacher – for instance, there can be a substantial number of term papers to be graded at the end -- but this is the price that is worth paying for the benefit of the students. It should be made clear that what we propose here is not a panacea on the problem of students' lack of critical thinking skills. It is rather an approach that when applied regularly can lead to significant improvement.

ACKNOWLEDGMENT

We would like to thank Douglas Hale for allowing us to use his Programming Languages homework assignment and Katarzyna Paprzycka for useful comments and help with English.

REFERENCES

1. Paprzycki, M., "Matrix Multiplication -- a Very Simple Program With Some Interesting Implications," The Journal of Computing in Small Colleges, Vol 8., No. 2, 1992, 154-162
2. Paprzycki, M., Khosraviyani, F., Wagaman, M., "Binary Search on a Linked List -- a Research Project in Experimental Computer Science," The Journal of Computing in Small Colleges, Vol. 7, No. 5, 1992, 16-21
3. Zalewski, J., Paprzycki, M., "Parallel and Distributed Computing Education: a Software Engineering Approach," to appear in: Proceedings of the CSEE'95 Conference, New Orleans, March, 1995

Appendix 1. Real-Time Systems Class Project

STEP 1. This is the list of topics to be selected for individual research and report preparation (+ class presentation) in Real-Time Systems class. Please select one which is not yet assigned and send me e-mail about your choice. I'll have to approve the topic before you can start working on it.

1. Real-Time Scheduling and Performance in Futurebus+
2. Real-Time Parallel Computing
3. Real-Time Distributed Computing
4. Real-Time Systems Benchmarks
5. Real-Time Object-Oriented Methodologies
6. Timed Petri Nets
7. Formal Method Z in Real-Time Systems
8. Ada as an Object-Oriented Language
9. MIDI -- Musical Instrument Device Interface
10. Simulated Annealing in Task Scheduling
11. Sensor Data Fusion
12. Kalman Filtering
13. Hartstone Benchmark
14. Network Controller for the Water Boiler Control Problem
15. Neural Networks in Real-Time Computing
16. Topic of your choice (must be approved by instructor)

STEP 2. The student is supposed to do a literature search to find sources. Use of UTPB library should be sufficient (for example, you can use on-line search, CD-ROM abstracts service, etc.). If you find sources, please contact me to see if they are appropriate. If not, I'll give you directions and provide additional information. If they are OK, you will need to read respective article(s).

STEP 3. The final report should be structured as follows.

1. Introduction and Background
   General information on the particular technology covered by this topic.
2. Problem Description
   In general, you should include here enough information (giving background) what particular problem or problems this technology/technique/algorithm is supposed to solve (problem formulation), stating its purpose.
3. Description of the Method
   Here you should include the solution, that is, how this technology/technique/algorithm, etc., solves the problem described above. If your topic covers more such methods, describe them all and compare.
4. Example
   Give an example to illustrate the usage of the described technology. This must be a detailed example (using numbers is the best).

(5. Conclusion)
(If appropriate, the report should include a couple of concluding statements: a summary. The contents of a summary usually vary, but may include information on unsolved problems, future work, applications, extensions, comparisons, etc.)

Bibliography
Cite the literature sources used. Remember that all literature sources listed must be referred to in the text of the report.

STEP 4. After the report has been submitted, a class presentation will be scheduled. Presentations normally take half an hour each. At the time of presentation all students in class should be provided with copies of the report (by e-mail or with paper copies). The talk should basically follow the structure of the report, that is:

1. Explain the problem
2. Describe the solution to this problem (the algorithm)
3. Illustrate the solution with an example or application
4. Conclude and leave some time for a few questions and discussion

Deadlines (penalties apply for delays):

1. Topic selection:	March 23, 1994 (class time)
2. Literature search:	March 30, 1994
3. First draft submission:	April 8, 1994 (Friday)
4. Final Submission:	April 18, 1994 (10pm)
5. Class Presentation	(according to schedule)

Form of Final Submission: e-mail to zalewski_j
(if you prefer submission of paper copies, please make as many as 12 copies for the whole class)

Report size: Should be between 5-20 pages (less than 5 pages is not very informative, more than 20 may be redundant)

Grading criteria:

1. Major criterion: 0-40 pts
Report contents (how well the problem and the algorithm(s) are described)

2. Additional criteria:
   - Literature search 0-20 pts
   - Report size and organization 0-20 pts

- Class Presentation 0-20 pts

Appendix 2. Programming Languages Class Project

One of the requirements of this course is that you prepare a research paper on a topic from the general area of programming languages. The paper will determine 25% of your grade for the course. It is due, in my hands, by December 9, 1993. There are no late papers; papers received after 12:00 noon on December 9 will not be accepted.

Here are some thoughts concerning the preparation of your paper:

1) Your paper will deal with a subject germane to this course. Several suggestions have been included below. You should talk to me sometime during the first half of the semester (the date will be announced) so that we can agree on your topic.

2) You will hand in to me, on October 14, one or two page abstract indicating what your paper will be about. This should include a brief discussion of the topic you have chosen, something about how you are approaching the subject, and a bibliography of sources you have consulted at that time.

3) Your paper will be presented in a professional manner. In particular, it will be printed on a computer printer. (Dot matrix is acceptable.) Please be sure to root out all typographical, spelling, and grammatical errors!

4) You should endeavor to find a sufficient number of relevant sources of sound academic quality for your research. You absolutely must give credit where credit is due. It is considered crooked to take credit for material you have borrowed from others. This will lead to severe academic penalties. I don't care whether you use footnotes or endnotes, as long as you give the citations. In either case, you will want to include a bibliography of all sources consulted in your research for the paper.

5) The layout of the paper is more or less up to you. It is usually considered good form to divide a longer paper into sections, to number pages, to double space, and to provide a table of contents. Please leave wide left margin as a space for comments. The length of the paper is determined by many factors, but I suspect you will not be able to do an honest treatment of a topic with some substance in less than nine or ten pages of text. Please feel free to write as much as may be necessary to make your case, but don't pad. Thirty five pages of drivel is just that, drivel. I do not grade by the pound. It is possible to write a 35 page paper and have it assigned the grade of F.

6) While I value your opinions, I am concerned that you also discover and analyze the thoughts of other experts in the field. Please do not present me with a long essay which simply describes your feelings about the topic. Do some digging, find out what the experts have to say, and then digest and analyze their thoughts. Let me know what the current thought is on the subject - often there will be more than one view, as is the case of the development of Ada (there's a topic for you). Your own analysis of these ideas and the conclusions you base on them is the heart of the work.

Listed below are some suggested areas of investigation for your research paper. This list is not intended to be exhaustive, only suggestive. If you have other ideas about what you might like to do, please feel free to pursue them, subject to your discussion with me.

A) The historical development of a specific programming language
- How did it come into being?
- Who wrote it?
- Why? What was the need for the new language?
- What did it contribute?
- What implementations are possible?
- What standards have been established?
- What is its current status? (Is it still in use?)
- Are there any new standards under development?

B) Case-study type description of a particular programming language (as in one of the chapters in section two of the text -- please pick a language not covered there)
- Compiled? Interpreted?
- Typing?
- Primitive data types
- Primitive operations
- Subprograms -- parameter passing, recursion, flow of control
- Information hiding
- Dynamic/static data structures
- Control structures -- loops, WHILE, UNTIL, CASE-of
- Multiprocessing, multitasking, intertask communication

C) ANSI standards
- What do labels like FORTRAN77 and COBOL74 mean?
- What is the significance of such tags?
- How standard are languages? why?
- How are the standards set up?
- For which languages are there standards?
- Who follows the standards? how do we know?

D) The extent to which languages are machine dependent
- How are languages modified for use on certain machines?
  - PCs?
  - minis and mainframes?
  - supercomputers?
- What will be the impact of various non-von Neumann architectures?
  - vector processors?
  - multiple processors?
  - massively parallel processors?
  - neural nets?

E) Various topics in natural-language processing
- Non-procedural languages
- What is the future of using English, or French, or whatever, as a computer language?
- What are the difficulties in natural-language processing?
- Written or typed input/output
- Spoken input/output
- Translation
- Syntax, semantics
- What impact has work in the area of natural-language processing in computer science had on areas like linguistics, learning theory, speech analysis, philosophy?

F) A comparative study of two or more languages
- How are they the same?
- How are they different?
- Be sure to discuss the major topics considered in the design of a language!

G) For the venturesome, a look into the future of programming languages
- What considerations will force changes in the design of languages?

What will the changes likely be?
In fifty years, what will have become of today's languages?
High Performance FORTRAN?
What will be the place of natural languages?

Appendix 3. Matrix Multiplication Project

You have recently learned that six different implementations of matrix multiplication may lead to different performance characteristics. In this homework you will be practically checking this hypothesis.

1. Implement all six versions of matrix multiplication.
2. Establish their execution time for:
   - matrices of sizes 50, 75, . . ., 200
   - one integer type and one real type
3. Write a report analyzing the results

Notes:
a) you can use any programming language,
b) this homework may not produce the expected results on a PC,
c) analysis of the results is the most important part of the homework.

Appendix 4. Basic Arithmetical Operations Project

In this homework I want you to investigate the efficiency of implementations of addition, subtraction, multiplication, division and system function calls between each other for different data types and for two different programming languages.

More specifically -- try to establish for two of your favorite programming languages if there is any difference between:

A) Inside each language:

1) efficiency of implementations of addition, subtraction, multiplication, division and a function call (for a language defined function),
2) performing operations described in 1) for (whichever are available) short integer, long integer, single precision real, double precision real and quadruple precision real.

B) Between the two languages:

1) compare the performances of the operations studied in A) between the two languages.

Notes:
- perform your experiments by running a loop 100K, 500K, 1000K, 2000K, 3000K and 4000K times and measuring the execution time,
- use averages of 4-5 runs,
- do not allow a smart compiler to fool you,
- when comparing the two languages pay attention to the representation of data types in each (e.g. what is FORTRAN's integer comparable to in C).

The most important part of the homework is the analysis of the results (it is not what you see, but why!). The analysis of results should have a form of a technical report as discussed in class.

Appendix 5. Sorting Algorithms Project

During last couple of lectures a number of sorting algorithms were introduced. In addition, for each of these algorithms claims were made related to their performance characteristics. The aim of this homework is to make practical comparisons of the performance of the following sorting algorithms:

- selection sort,
- insertion sort,
- bubble sort,
- shell sort.

In the first part of the homework you are supposed to study the algorithm performance for the random data, sorted data and inverse sorted data.

Notes:

- this is again a computer "time consuming" homework;
- based on trial runs, establish number of elements to experiment with (for instance, the minimum time must be meaningful and the maximum time should be no longer than 30 minutes);
- each result should be an average of 3 runs;
- random data should be generated by a random number generator;

In the second part of the homework study the performance of the algorithms for partially sorted data. To do this select one of the input sizes experimented with earlier and run the experiments for the data 10, 20, ..., 100% sorted.

Summarize your findings in a technical report having the form as discussed in class.

Appendix 6. Sparse Matrix Multiplication Project

In your last homework you have implemented a link-list based sparse matrix multiplication algorithm. In this homework you will study the effect of increasing the number of non-zero elements in the matrix on the performance of the implemented algorithm. You will also try to establish for what fill-in density it is better to use standard matrix multiplication algorithm. To achieve this goal:

A) select matrix size for which the execution time for the best version of standard dense matrix multiplication takes approximately 1 to 2 minutes.

B) time the execution of the sparse matrix multiplication algorithm systematically increasing the fill-in density.

Present the results of your experiments in a technical report form discussed in class.