1-1-2002

A hybrid approach to the core curriculum

I. Piper
University of Wollongong, ian@uow.edu.au

P. Castle
University of Wollongong, risque@uow.edu.au

A. Fuller
Faculty of Informatics, University of Wollongong, annef@uow.edu.au

G. Awyzio
University of Wollongong, gene@uow.edu.au

A hybrid approach to the core curriculum

Abstract

In this paper we review the IEEE/ACM CC2001 model. We then describe our proposed core CS curriculum comprising four strands: programming languages, algorithms, discrete mathematics and systems. These sequences are to be taught over the first two years of the Bachelor of Computer Science Degree and need to be taken in parallel.

Keywords

computer science education, educational courses, mathematics, programming languages

Disciplines

Physical Sciences and Mathematics

Publication Details

This paper originally appeared as: Piper, I, Castle, P, Fuller A and Awyzio, G, A hybrid approach to the core curriculum, 32nd Annual Frontiers in Education, 2002, vol 2, 13. Copyright IEEE 2002.

A HYBRID APPROACH TO THE CORE CURRICULUM

Ian Piper[1], Peter Castle[2], Anne Fuller [3] and Gene Awyzio[4]

Abstract *— In this paper we review the IEEE/ACM CC2001 model. We then describe our proposed core CS curriculum comprising four (4) strands: Programming Languages, Algorithms, Discrete Mathematics and Systems. These sequences are to be taught over the first two years of the Bachelor of Computer Science Degree and need to be taken in parallel.*

Index Terms — computer science, core curriculum, programming.

INTRODUCTION

Despite its many shortcomings, the most common model in Computing Curricula emphasizes the teaching of programming first, frequently with the inclusion of algorithms and data structures within a single subject sequence. The IEEE/ACM CC2001 Report [1] proposes six (6) curricular models. These are known as imperative-first, objects-first, functional-first, breadth-first, algorithms-first and hardware-first. The first three models recognize that, for a variety of reasons, the teaching of programming will continue to dominate the introductory stage of the teaching of Computer Science while the latter three are suggested alternatives aimed at overcoming many of the identified deficiencies of each of the programming first approaches.

Following a recent core curriculum review, the School of IT and Computer Science at the University of Wollongong identified that one of the main problems with the current structure of our Computer Science program is finding an appropriate balance between problem solving and analytical skills and the more mechanical language acquisition skills. After examining the CC2001 models it was decided that the School would be best served if we adopted a hybrid approach.

THE PROBLEM

Examination of first year outcomes in Computer Science revealed that, while students had reasonable facility with coding in a programming language (C++), they had little or no skill in the process of problem analysis and algorithm development despite the teaching of these skills in the introductory CS sequence; due in no small part to the time constraints caused by the syntactic complexity of a 'real' language as opposed to a 'teaching' language such as Modula-2.

A second consideration in the redesign of the core curriculum was the expansion of the expected knowledge base of Computer Science graduates, including necessary mathematics concepts.

THE SOLUTION

Given that students entering second year seemed to possess acceptable coding skills, we were reluctant to diverge too far from the existing imperative-first approach but needed to better provide the other necessary skills, knowledge and attitudes. This led to an approach involving a combination of the imperative-first and algorithms-first models.

The core curriculum is divided into four (4) parallel strands:

- Programming Languages — a three-subject sequence concentrating initially on pure language acquisition (using an algorithm-to-program approach) and, later, on available coding support tools;
- Algorithms — a two-subject sequence beginning with informal problem solving skills and progressing to a more formal development of algorithms and data structures;
- Discrete Mathematics;
- Systems — a three-subject sequence starting with an introduction to IT issues, then coverage of the interfaces between a program and its environment (HCI, operating systems, databases, networks) and, finally, the team-based development of medium scale software.

To ensure consistency and completeness of coverage across and within these strands, a detailed syllabus is being developed for each subject and a coordinator role has been established with responsibility for oversight of the entire core curriculum. This coordinator also will monitor progress and report on the success of the implementation of these changes.

REFERENCES

[1] The Joint Task Force on Computing Curricula, "Computing Curricula 2001 Computer Science", December 2001

[1] Ian Piper, University of Wollongong, School of IT and Computer Science, NSW, Australia, ian@uow.edu.au
[2] Peter Castle, University of Wollongong, peter_castle@uow.edu.au
[3] Anne Fuller, University of Wollongong, annef@uow.edu.au
[4] Gene Awyzio, University of Wollongong, gene@uow.edi.au

0-7803-7444-4/02/$17.00 © 2002 IEEE