Supporting CC2001's Endorsement of a Three-Course Introductory Sequence

Edward G. Okie, K. Todd Stevens, and J. D. Chase
Department of Information Technology
Radford University
Radford, VA 24142

and

John Lewis
Department of Computing Sciences
Villanova University
Villanova, PA 19085

Abstract

As technological needs evolve, so must IT curricula. Often, such change is not simply a choice of substituting one topic for another. In order to establish a solid foundation for our students, this paper argues the need for a third course in the introductory computer science sequence. More specifically, it describes two specific departments that have already implemented a new CS 1-2-3 sequences.

Keywords: CC2001, CS1, CS2, introductory sequence, object-oriented, curriculum development

1. INTRODUCTION

As is outlined in CC2001 (ACM 2001), the typical CS1/CS2 introductory sequence has been a cornerstone of many computing curricula. At most institutions, CS1 lays the groundwork of basic problem solving and programming skills, and CS2 follows with an emphasis on data structures. However, in recent years, this approach has begun to fall short of its goal of giving students a strong foundation for their computing education and career.

Technological advancements – or, to be more specific, academia's embrace of certain technological advancements – have simply given us more topics to cover. Some topics driving this change are object-orientation and graphical user interfaces, and most technologists would agree that these are now fundamental topics. However, he problem is that these topics do not simply replace the things that we used to teach in an introductory sequence; they augment them. Bear in mind, it is just as difficult for students today to understand *if statements* and *while loops* as it was for us to learn 10, 20, or 30 years ago and we cannot decrease the time allocated to them. Now, the foundational material has expanded, and we simply need more course-time to adequately cover all of the material.

For example, object-orientation adds many new topics that we did not teach in a procedural setting, including classes, objects, inheritance, and polymorphism. Yet still prevalent are core topics such as data types, control structures, parameter passing, arrays, dynamic and linked structures, recursion, sorting, searching, and all of the classic data structures. Functional decomposition has not gone away; it is now just subordinated to object decomposition. Our students must still know how conditionals work, even though in some situations conditionals can yield to the elegance of polymorphism.

Graphical user interfaces, and the entire world of event-driven programming that underlies it, are now fundamental. Graphical components, listeners, events,

and user interface design are all additional topics that previously were ignored or did not even exist.

Other topics that some educators want to give more attention to at a fundamental level include exception handling, client-server processing, basic database concepts, and basic graphics. Most of us also want more time for establishing and building problem solving skills.

As we embrace these topics, there is less time to cover everything. Some topics are getting squeezed to the point students do not come away with a true understanding of the issues involved. At some schools the first few weeks of CS2 are now used to cover material that CS1 did not have time to get to. This leaves less time for traditional CS2 topics.

As is endorsed by CC2001 (ACM 2001), we believe the natural and proper solution to this problem is to add a third course to the introductory sequence. Furthermore, instead of tacking it onto the end and dumping all of the new topics into a third course, the entire set of introductory topics should be reorganized into a comprehensive, three-course sequence.

The bulk of this paper describes the philosophies and curriculum from two schools that independently recognized the two-course problem and have already embraced a three-course approach. We begin by describing the schools and their computing curricula in general.

2. RADFORD UNIVERSITY

Radford University is a medium size, comprehensive university, located in the mountains of southwest Virginia. In August 2001 the university brought together two well-established computing programs, computer science and information systems, to form the new College of Information Science and Technology. Computer Science, a CSAB accredited BS program that has some 200 majors, was formerly housed in the College of Arts and Sciences. Information Systems, which has about 300 majors, was from the College of Business and Economics. Added to these existing programs are new concentrations in software engineering, networks, database, and business systems development. Overall, this combination of new and existing concentrations and programs forms an IT curriculum that is similar to one described by Denning (Denning 2001).

As part of the design of this new IT curriculum, we designed a new core of introductory courses that is to be taken by all incoming students in the college. At the heart of this core is an expanded three-course introductory sequence that replaces the traditional CS1 / CS2 sequence. The new sequence will be taken by all students entering the program except students in information systems, who will only take the first two courses in the sequence. The topics in the courses have been chosen and sequenced so that each group of students receives the background that they need.

3. VILLANOVA UNIVERSITY

Located near Philadelphia, Pennsylvania, Villanova is a comprehensive university with approximately 7000 undergraduates. The Department of Computing Sciences is part of the College of Arts and Sciences, and offers a B.S. in Computer Science, a B.S in Information Science, and a M.S. in Computer Science.

Villanova was one of the first schools to use Java as their introductory language in CS1 / CS2, adopting it in 1997. In Fall 2002, the Department of Computing Sciences added a third course to the introductory sequence, requiring it of all incoming students in both the Computer Science and Information Science programs.

4. CURRICULA

The faculty at both Radford and Villanova independently arrived at the conclusion that a third introductory course was needed. The design of these new courses was the result of debate and discussion among their individual departments, as well as ultimately comparing notes between the two schools.

After providing an overview of the three course sequence at both schools, we compare and contrast the approaches.

Radford curriculum issues

While working to implement Radford's new College of Information Science and Technology, we found several problems that had the common solution of revising the introductory sequence by expanding CS1 and CS2 into a three course sequence and developing a common core of courses that would be taken by all students in the college.

One such problem was the very similar upper division courses found in the CS and IS programs in areas such as networks and database. To eliminate this duplication we replaced each set of duplicated courses with a single sequence of courses for that particular area. While each sequence was designed to serve students who were pursuing that particular track in the college, there was also a need to make the first course in each track accessible to other students in the college. To make this possible, a common introductory core of courses was developed that would prepare students for any of the existing or proposed programs or tracks.

In developing this core we were faced with a dilemma: CS1 was not an adequate background for these courses, however our traditional CS2 covered several topics in more depth than was required for students in information systems. To solve this problem we spread the topics from CS1 and CS2 across a three-course sequence and only require students in information systems to take the first two courses in the new sequence. Students in the traditional BS in CS program and in tracks such as software engineering and networking take all three courses in the new sequence.

Another problem that led to the development of a three course introductory sequence was a desire to place object technology earlier in the curricula. Specifically, we adopted Java for the introductory courses. In the old computer science program, students did not encounter objects until a Software Engineering course in their fourth semester, and information systems students do not see them at all. In addition to pedagogical reasons, making object orientation the initial paradigm provides students with two immediate benefits. It enhances internship opportunities for all students and, for information systems majors, it gives them exposure to an important paradigm that they had not seen previously.

Radford CS1: The new CS1 course at Radford is taught in a three hour lecture, two hour closed-lab format. The material in the course is divided into 6 areas, which are shown below along with their underlying topics:

- Programming Fundamentals: variables, assignment, expressions, conditionals, loops, keyboard I/O, file I/O (records), arrays of primitive types, arrays of objects, class libraries
- Objects and Classes: data and methods, instance vs. class data & methods, inheritance, encapsulation, visibility modifiers, instantiation, reference types, primitive types, overriding, overloading, simple polymorphism
- Language Topics: type conversion, string manipulation, Java Virtual Machine, Java Applets, simple graphics
- Software Engineering: problem solving, software analysis and design, testing and debugging, documentation and program structure, abstraction, basic data structures
- Algorithms: searching, simple sort, recursion
- Recurring Themes: ethics, analysis of algorithms

Radford CS2: As noted above, CS2 is the last core course that is taken by all students in the college. Therefore, it must adequately prepare IS students for courses such as database I, software engineering I, and networks I. To accommodate this change, topics were carefully selected to give information systems students the background they need. As another means of making CS2 more accessible, it was converted from a four hour lecture format to a three hour lecture, two hour lab format. This, however, meant that the coverage of some topics would have to be altered to make up for the difference in lecture time. The areas and topics for CS2 include the following:

- Programming Fundamentals: multi-dimensional arrays, recursion
- Java Topics: interfaces and abstract classes, applets, applications, inner classes
- Graphical User Interface: events, listeners, components
- Data Structures: stacks, queues, vectors, lists, binary tree concepts, binary search tree concepts
- Recursive Sorting and Searching Concepts: quick sort, merge sort, binary search (arrays and trees)
- OO Topics: more work with objects, references, classes, methods, instance vs. class data & methods, inheritance, and polymorphism; new material on persistence, serialization, marker interfaces
- Software Engineering: problem solving, software analysis and design, testing and debugging, documentation and program structure, UML, encapsulation, abstraction, data structures
- Language Topics: linked structures, recursion, exceptions
- Recurring Theme: analysis of algorithms

Radford CS3: A CS3 course has been created to accommodate the change in topics from CS2 and to address another problem created by the addition of new IT concentrations. The new concentrations only require one semester of calculus. Therefore, the students in these programs would not take a traditional junior-level data structures and analysis of algorithms course because they would not have adequate background in calculus. To remedy this deficiency, it was important to move the most pertinent of the topics from this course to CS3. Unlike the first two courses in the sequence, CS3 will be taught as a three hour lecture course. The areas and topics for CS3 include the following:

- OO Topics: emphasize design issues

- Data Structures: binary trees, binary search trees, heaps, priority queues, hash tables
- Sorting and Searching: quick sort, heap sort, merge sort, binary search (arrays and trees)
- Language Topics: recursion, threads and basic synchronization
- Analysis and Design: patterns, UML, processes, use cases
- Recurring Themes: analysis of algorithms, GUI, software engineering

Radford's Evolving Curriculum: In Table 1 below, Radford's old curriculum is shown on the left and the new curriculum on the right. In both columns, old CS1 topics are in plain text, CS2 topics are shown in *italics*, and the Object Oriented (OO) topics are shown in **bold**. Underlined items in the old curriculum became obsolete with the new curriculum, and underlined items in the new curriculum are topics that are new to the curriculum because of the OO approach in Java. The new curriculum in Table 1 shows the "Objects-first" approach from CC2001 (ACM 2001), with OO topics spread throughout the 3-course sequence.

Table 1:
Comparison of Radford's Old and New Curriculums

Topics in Old Curriculum	Topics in New Curriculum
CS1: Basics: sequence, selection & repetition Basic I/O Problem solving methodologies Top-down design and pseudo-code Structure Charts Data Abstraction Records Arrays Ethics Intro to Sorting	CS1: Basics: sequence, selection, repetition Basic I/O **Primitive types & Objects** **Classes & Objects** **OO Data & methods** **Libraries, overloading, visibility modifiers** String manipulation Software engineering basics: problem solving, documentation & program structure, testing & debugging, *encapsulation, abstraction & data structures* Arrays Ethics Basic searching & sorting
CS2: *Data abstraction & information hiding* *Encapsulation, abstraction & data structures* *Pointer data types* *Data structures: stacks, queues, lists, binary trees, heaps, hashing* *Recursion* *Linked structures* *Sorting & searching* *Multi-dimensional arrays* *Exceptions*	CS2: **Interfaces & abstract classes** **Applets, Inner classes, Events, Listeners, Components, basic UML** *Data Structures: stacks, queues, vectors, lists, binary tree basics* **Additional OO Topics, e.g. persistence** *Additional software engineering basics* *Recursion* *Linked structures* *Sorting & searching* *Multi-dimensional arrays*
Fourth semester OO software engineering: **Software engineering lifecycles & methods** **OO analysis** **OO design** **OO programming languages** **Classes, Objects** **Inheritance, Polymorphism** **GUIs** **Interfaces & abstract classes** **Applets, Inner classes, Events, Listeners,** **Design patterns** **UML**	CS3: *Binary trees* *Heaps* *Hashing* *Complex Sorts* *Recursion* Threads & synchronization **Design patterns** **UML**

Villanova curriculum issues
Villanova had been using an object-oriented approach for CS1 and CS2 for several years. The shift to a three course sequence primarily allowed each topic to be covered in more depth and with more practice.

The 'room' to add the new CS3 course comes at the cost of eliminating one (of three) free elective that the students had in their overall schedule.

Villanova CS1: CS1 at Villanova is a 4-credit course. Some instructors use all contact hours as lecture while others incorporate a lab period. Students are given fairly well defined project descriptions in CS1, with a growing emphasis on requirements analysis and design.

Instructors have the option to use a third-party I/O class to simplify keyboard input, but they must expose the students to 'real' Java I/O declarations and use by the end of the course. CS2 instructors do not use third-party support classes for I/O at all.

The topics covered include:

- Programming Fundamentals: variables, assignment, expressions, conditionals, loops, keyboard I/O, file I/O, arrays of primitive types, arrays of objects
- Object-Orientation: objects, classes, instantiation, encapsulation, data and methods, instance vs. class data & methods, overloading, overriding, visibility modifiers, references, class libraries
- Language Topics: type conversion, string manipulation, Java Virtual Machine, Java Applets, simple graphics
- Software Engineering: problem solving, testing and debugging, documentation
- Recurring Themes: ethics, basic analysis of algorithms

Villanova CS2: Like CS1, the CS2 course at Villanova is four credits including an optional lab. The initial emphasis in CS2 is on object-oriented design issues, including conceptual tools that were only mentioned in passing in CS1 such as inheritance. Data structure concepts then fit nicely into this flow because they can be shown to capitalize on these design characteristics.

One difficult issue in the redesign of the CS2 course was the decision to wait until CS3 to focus on most non-linear data structures such as trees and graphs. However, the split has taken on a natural feel of its own (linear in CS2 and non-linear in CS3) rather than the initial awkward feeling it gave us to split the coverage of data structures in general. We feel that this is a good example of where we should not feel bound by historical trends.

The topics covered in Villanova 's new CS2 course include:

- Object-Orientation: inheritance, abstract classes, overriding, interfaces, marker interfaces polymorphism, persistence, serialization, inner classes
- Programming Fundamentals: linked structures, recursion, basic sorting and searching, exceptions
- Graphical User Interfaces: events, listeners, components
- Data Structures: tables (multi-dimensional arrays), stacks, queues, vectors, lists
- Software Engineering: problem solving, requirements analysis, software design, basic UML, stronger emphasis on tool use
- Recurring Themes: analysis of algorithms

Villanova CS3: The fairly clean division between linear and non-linear data structures between CS2 and CS3 provides the opportunities to explore all of the related topics in more detail. The new CS3 course also provides a place to include topics that have otherwise been ignored in traditional CS1 / CS2 sequences.

The topics covered in Villanova 's new CS3 course include:

- Object-Orientation: emphasize design issues
- Data Structures: binary trees, binary search trees, general trees, directed and undirected graphs, hash tables
- Algorithms: advanced sorting and searching
- Database: fundamental concepts, relational model, tables, fields, keys, queries, use of JDBC
- Software Engineering: advanced design techniques, UML details, client-server architecture
- Recurring Themes: analysis of algorithms

5. COMPARING CURRICULA

Given the manner in which the curriculums are currently defined, there are many topics that are covered in the same manner in corresponding courses between the two schools.

One general difference is the location and emphasis given to particular topics. For example, Radford currently introduces both GUIs and recursion in CS1 at a fundamental level, and then reinforces them both throughout CS2 and CS3. Villanova segments these topics somewhat, deferring both until CS2.

The only substantive difference in the two curricula is the emphasis on databases at Villanova in CS3. Since CS majors at Radford are required to take a database course later in the curriculum, getting database fundamentals into the introductory sequence is not a priority. At Villanova, where the advanced database class is an elective, they can now guarantee a minimal exposure to database concepts by including the basics in the required CS3 course.

Overall, the two curricula are very similar and both follow the recommendations of the "Objects-first" model in CC2001 (ACM 2001).

6. DISCUSSION

The main impetus for investigating switching to a three-course introductory sequence from two courses is the volume of additional material that currently needs to be covered. With the development and advancement of the IT field, the breadth and depth of material that needs to be taught keeps increasing. We have simply reached a point where the amount of time we spend on the fundamental material has increased enough such that we need to introduce students to the material over three semesters.

To continue to provide students with a quality education, giving them enough personal attention and time is (arguably) easily addressed by using closed labs. Working with students one-on-one allows students, as well as instructors, to discover whether students are truly learning material and which material each student might need to study more. Unfortunately, closed labs typically require more contact hours from professors/instructors and/or graduate teaching instructors or peer instructors. Also, this may "take up" more credit hours in the curriculum, which may reduce electives for students. We feel that overall this reduction in electives is more than offset by the quality of the foundational knowledge that the students acquire from the three-course sequence.

The division of linear and non-linear data structures in CS2 and CS3 at Villanova appears to work well. Even though this has not yet been explicitly adopted by Radford, the courses may be organized that way in the future.

Finally, the objects-first approach that has been adopted by both schools is significant. Such an approach is increasingly important to our industrial counterparts (i.e. employers of our students). We feel that it is both a pedagogical advancement as well as a practical advancement because employers see this as immediately useful for part-time and summer employment for our students. This cannot be over-emphasized: students seem to learn easier and better and employers are pleased.

A major challenge for both programs was how best to introduce and integrate OO topics with traditional introductory sequence topics. At Radford, the following sequence for CS1 seems to be successful in addressing this challenge by providing students with an OO foundation along with the fundamentals of programming,. The other introductory courses are sequenced similarly.

To begin with, students are exposed to using objects as tools and to the concepts of the traditional sequential execution, assignment, and simple input and output. Primitive types and Strings are then introduced. Objects are introduced as providers of services, and classes are presented as templates for objects and as libraries of methods. No internal details are initially provided. Next, students learn to use the various conditional and loop statements. Up to this point, all code is written in the *public static void main* method. With this as a background, students then learn in detail how classes are constructed by defining and using simple classes. This progresses to using multiple classes that interact. Finally, arrays and vectors are covered, and inheritance is introduced and used in the last few simple laboratory assignments.

During a semester, the students write approximately twenty small programs in the laboratory setting with the assistance of the instructor and Peer Instructors (students who have recently and successfully completed the course for which they PI). The students also write four or five out-of-class assignments that are more complex, and they must do this work individually with only minimal assistance from their instructor and Peer Instructor.

7. CONCLUSION

The general consensus at both schools is a feeling of relief. With the creation of CS3 as an integral part of the introductory sequence, the time now seems available to cover topics that had been given too little attention, or none at all, in the past.

An important practical issue that comes up is the use of textbooks. Currently, both Radford and Villanova use two textbooks to cover all three classes of their new introductory sequences. The first book is used throughout CS1 and the first portion of CS2. The second book is used for the rest of CS2 and throughout CS3.

Villanova has found the need to provide some supplemental material for the extra topics covered in CS3.

Faculty support for the new CS3 course at both universities has been largely strong. The occasional devil's advocate has wanted to ensure that some topic shifts did not cause the topic to be de-emphasized. So far, the opposite appears to be true. That is, the topic shifts have allowed them to be given more attention than previously.

Though still early in the process, the feedback on this new approach has been positive at both schools. Some issues will certainly need to be ironed out as the smoke clears. But these modifications are motivated by our belief that change is best managed by those who create it.

8. REFERENCES

ACM, Computing Curricula 2001, http://www.computer.org/education/cc2001/final/index.htm (accessed July 10, 2002).

Denning, Peter J., The IT Schools Movement, Communications of the ACM, August 2001, pp. 19-22.