Requiring a Systems Analysis and Design Course for a Computer Science Major

David V Beard[1], **Kevin R Parker**[1,2], **Thomas A Ottaway**[1,2], **William F Davey**[3], **and Corey D Schou**[1,2]
[1]Computer Science, Idaho State University, Pocatello Idaho, USA
[2]Computer Information Systems, Idaho State University, Pocatello Idaho, USA
[3]School of Business information Technology and Logistics, RMIT University, Melbourne, Australia

Abstract - *A systems analysis and design course can be used successfully to replace the traditional introduction to software engineering course in a computer science curriculum. The systems analysis and design course can provide an improved design experience as students and faculty stay focused on learning analysis and design, rather than programming. Such a course also can enhance computer science students' knowledge of project management, cost analysis, and business process flow analysis. Missing essential computer science knowledge outcomes must, however, be included in other required CS courses. Such a substitution all but requires that the instructor has significant software development and computer science backgrounds.*

Keywords: computer science curriculum, software engineering, systems analysis and design.

1 Introduction

For the last several years, Idaho State University's computer science (CS) department has required their students to take a systems analysis and design course from the computer information systems department (CIS) in place of the traditional introduction to software engineering course [e.g., 1]. While some of the course objectives typically covered in a software engineering course have had to be covered in other required computer science courses, use of the systems analysis and design course has allowed our computer science department to provide a general purpose major that meets ACM and other curricula standards while providing CS students with a broader analysis and design experience.

There are a number of circumstances that have allowed Idaho State University's (ISU) computer science department to successfully replace a software engineering (SE) course with systems analysis and design (SA&D) that might not be present at other universities. First, ISU's CS curriculum is a general purpose major focused on the development of large complex computer systems; many of their majors take jobs building complex business applications, so much of the material covered in SA&D courses, that is not part of the ACM 2013 Tier 1 or tier2 [2] required objectives, is still highly useful to CS students. In fact, a number of ISU's CS students choose to complete an MBA after finishing an undergraduate major in computer science and working for several years in industry. Second, ISU's CIS major is fairly technically oriented with a strong emphasis on IT technical skills, security considerations, systems analysis and design, data base design including entity-relationship diagrams [3] and some development of business applications. Third, ISU's CIS faculty has extensive software applications development and/or computer science backgrounds.

Perhaps most importantly, ISU's CS and CIS departments work well together, have considerable mutual respect, and thus have been able to cooperate in the details of curricula and course outcomes. Further, each has developed clear written definitions of what CS and CIS are for Idaho State University, and they have made these distinctions clear to faculty, students, administration, and prospective employers. Defining and maintaining these distinctions is academically and politically essential.

The purpose of this paper is to discuss how a systems analysis and design course can be used successfully to replace the traditional introduction to software engineering course in a computer science curriculum. We first detail what CS and CIS majors are at Idaho State University, and discuss some of the similarities and differences between an introduction to software engineering course, and a systems analysis and design course. We also describe what knowledge outcomes need to be incorporated elsewhere in the CS curriculum to ensure complete coverage of software engineering. Finally, we summarize our initial experiences and feedback in including SA&D in the CS curriculum.

2 Background

2.1 Computer Information Systems

Computer information systems (and its variants management information systems and information systems) evolved over the years from the referent disciplines of behavioral science, management theory, and computer science. Although the roots of information systems (IS) are firmly anchored in computer science, the instruction of IS courses must transcend the technological aspect, and encompass the organizational and behavioral issues that are integral in the business world. Every course must not only instruct the student in the application of the pertinent software package, but must also emphasize the importance of such tools in a corporate environment. Information systems can be traced back to the systems analysts who worked in the typical data processing departments of the late 1950s. Systems analysts elucidated the needs of a business situation, and developed algorithmic solutions – typically represented by flow charts – as well as managed the software development process. Systems analysts were aided by "programmers" who translated those flow charts into languages such as COBOL, keypunched the code, and resolved any syntax issues. CIS curricula included typically two courses in COBOL focused on file management and reports, as well as on the skills needed to analyze business situations and manage the process of developing solutions.

However, this initial CIS model quickly blurred. By the mid-1970s, computing departments were hiring "programmer analysts" who did both systems analysis and programming. Relational databases [4] became widespread by the mid-1980s [5] and by the late-1990s, web-based database applications had replaced most COBOL business systems requiring a far higher level of technical expertise including web applications, database management systems [6], networks, computer operations, and computer security. Two programming courses were no longer sufficient to do much development. Some CIS departments increased their technical course requirements while others dropped programming all together.

ISU's computer information systems major has somewhat more of a development emphasis than many IS programs, with two required programming/development classes as well as some programming required in a database class. The CIS major is part of a business school, so CIS students also take accounting, marketing, statistics, project management, and business law.

2.2 Computer Science

Computer science is primarily concerned with the development of complex software and to some extent hardware. Computer science evolved out of the systems programmers' role in the data processing departments of the late 1950s and in operating systems development by computer manufacturers. Computer science systems programmers wrote and optimized asynchronous operating systems, tuned disk drivers, developed data structures, wrote custom algorithms, etc., often in assembler and FORTRAN. Computer scientists also had the math skills to handle more scientific-oriented situations such as operations research algorithms and early simulation modeling with a focus on efficiency, effectiveness, and fault freeness.

Computer science curricula educate students in abstract reasoning about software programs and, above all, develop their ability to implement large software systems. ACM 1978 recommended curriculum [7] was based on this understanding and included CS-I, CS-II, computer organization, data structures, operating systems, and compilers. CS majors typically completed math courses at least as far as differential and integral calculus, discrete math, and linear algebra. The ACM 2008 update [8] and ACM 2013 [2] curricula guidelines now urge the inclusion of information assurance and cyber security knowledge outcomes.

Idaho State University's computer science (CS) department has embraced this traditional systems programmer notion of CS by developing a general purpose curriculum focused on implementing large, complex, high-performance, secure, asynchronous systems that require complex algorithms and intricate data structures including network, operating system, compiler, graphics and simulation packages. The curriculum incorporates 30 credit hours of math and science including differential and integral calculus, linear algebra, discrete math, and statistics. However, while mathematical formalism is essential, it is a means to an end, not an end in itself. The major is well above the ACM/IEEE 2013 curriculum minimum (Tier 1 + 90% Tier 2) [1] while being less than 1/2 of the total credit hours for the undergraduate bachelor of science degree. This allows students to add a second major in math or a business minor and eventually, an MBA in a 5th year.

The computer science department's focus on large systems development is due – to some extent – to its faculty members who have extensive industrial experience that includes writing hundreds of thousands

of lines of code while developing tools for automotive engineers, radiologists, pilots, etc.

2.3 Fundamental distinction between CIS and CS

While computer science is concerned with the optimal design and implementation of software, computer information systems concentrates on the most efficient use of the finished package. For example, a database course in a computer science curriculum includes the internal organization of the database management system examining such theoretical issues as the most efficient algorithms for locking mechanisms and deadlock detection. A course by the same name in CIS generally examines the importance of organizing the data so that, from a user perspective, database access will be most efficient.

This fundamental distinction is reflected in the expected skills of the computer science and computer information systems students: The CS student has to exhibit a considerable gift for abstract reasoning, but is not expected to interact successfully with as large a variety of application domains. CIS students have a strong understanding of the management, marketing, and accounting facets of an organization, allowing them to work more effectively in an organizational setting. In addition, interpersonal skills and communication skills are more strongly emphasized in the CIS major. CIS students are also better versed in assessing the needs of the end-users, and are therefore more capable of designing software systems that satisfy those needs.

3 Software engineering

The ACM 2013 curriculum [2] describes software engineering as follows:

> Software engineering is the discipline concerned with the application of theory, knowledge, and practice to effectively and efficiently build reliable software systems that satisfy the requirements of customers and users. This discipline is applicable to small, medium, and large-scale systems. It encompasses all phases of the lifecycle of a software system, including requirements elicitation, analysis and specification; design; construction; verification and validation; deployment; and operation and maintenance. Whether small or large, following a traditional disciplined development process, an agile approach, or some other method, software engineering is concerned with the best way to build good software systems. Software engineering uses engineering methods, processes, techniques, and measurements. It benefits from the use of tools for managing software development, for analyzing and modeling software artifacts, for assessing and controlling quality, and for ensuring a disciplined, controlled approach to software evolution and reuse.

Tomayko [9] identifies three periods in the history of software engineering education: the era of single free-standing courses (prior to 1978), the early graduate programs (1978-88), and the rapid spread of graduate programs influenced by the Software Engineering Institute's efforts (since 1988). By the 1970s a number of computer scientists had realized that software complexity was not just a matter of algorithms and data structures, but that software could become overwhelmingly complex due to its sheer size; even straightforward code becomes overwhelmingly complex when it grows to perhaps 300,000 lines of code and 6000 subroutines [e.g. 10]. No software engineering course was mentioned in the ACM 1968 [11] or 1978 curricula, though ACM 1978 recommended CS curriculum consider an optional "Software design and development" course. In 1978, Spencer and Grout [12] saw the need for including a SA&D course in a CS curriculum under the name "software engineering: large systems design". Constantine's [13] classic "Structured design" stressed the need for systems analysis and design to include a software implementation component so students could see the results of their design, and correct problems.

Table 1 –ACM 2013 tier-1 and tier-2 course contact hours per outcome [2]

Software Engineering Outcome Category:	**Tier1**	**Tier 2**
Software Process	2	1
Project Management		2
Tools & Environments		2
Requirements Engineering	1	3
Software Design	3	5
Construction		2
Verification & Validation		3
Software Evolution		1
Reliability		1

Even by the late 1980s the importance of SA&D and SE had not been completely embraced. In 1988, Poole and Callihan [14] argued for the inclusion of a SA&D course in the CS curriculum suggesting that many graduates have "a merely superficial acquaintance with

…SA&D" and that "many CS majors graduate under the misapprehension that SA&D is a career path that has little or nothing to do with computer science".

The ACM/IEEE 2001 CS curriculum detailed a series of software engineering courses such as software engineering, software process improvement, and advanced software development [1]. The ACM/IEEE 2013 curriculum guideline no longer stipulates particular courses but now provides a list of tier-1 and tier-2 software engineering "knowledge outcomes," most of which should be covered somewhere in a computer science curriculum (Table 1).

3.1 Is the initial software engineering course better without programming?

Both Constantine [13] and the ACM/IEEE 2013 curriculum [2] urged that software engineering for computer scientists be taught in conjunction with systems development. However, over the years, the authors have discovered a number of problems with this traditional software engineering pedagogical approach. First, in our experience, CS design courses that involve writing code, quickly devolve into only a small amount of internal design and analysis, with the bulk of the course becoming a conventional coding class. We suspect that computer scientists like writing code but perhaps are not as enamored with what might be perceived as the tedium of analysis and design.

Second, feedback from employers indicate that many of ISU's CS students did not really understand testing procedures, maintenance, version control, task analysis, data flow diagrams, project management tools such as PERT (program evaluation and review technique) and other techniques for developing large software systems. CS students were well equipped for entry level programming jobs but had difficulty moving into team leader and higher level management positions.

4 Systems analysis and design

Systems Analysis has been defined as follows:

> …the specification of what the system needs to do to meet the requirements of end users. In … systems-design … such specifications are converted to a hierarchy of charts that define the data required and the processes to be carried out on the data so that they can be expressed as instructions of a computer program. Many information systems are implemented with generic software, rather than with such custom-built programs" [15].

Pool and Callihan [14] argue that software engineering is a subset of systems analysis and design. They state that: "SA&D concerns itself with methodologies to manage the development and maintenance of computer-based systems in general, including hardware, software, people, plant, and the interfaces that link all the components together into a functionally harmonious whole. Software engineering focuses on software development within the already given context of the larger system [14]. One definition of software engineering includes the terms "analysis" and "design" [15].

ISU's systems analysis and design course – taken by both CS and CIS students – is somewhat unique as all of the CIS students have had at least one development course and the course is taught by a CIS professor with a PhD in information systems, a MS in computer science, and extensive industrial experience building applications and kernel level code. The course outcomes listed in Table 2 have been developed in conjunction with CS faculty.

Table 2: ISU Systems Analysis and Design Outcomes

On completion of the course, systems analysis and design students will be able to do the following:
Explain the fundamental concepts of systems analysis and design.
List and explain the important development methodologies for complex systems.
Demonstrate an awareness of the complexities of requirements determination
Demonstrate the analytical skills required to examine a situation, to understand thoroughly the factors involved, to recognize any problems, and to derive potential solutions
Analyze system requirements and specify system processes and data flows, express requirements in use cases, design user interfaces, and develop a systems proposal
Employ appropriate systems design tools such as structure charts, process specifications, UML, use cases, and dialog flow designers to design a system and its user interface
List and explain the fundamental concepts behind the implementation, testing, conversion, and maintenance of a system.

4.1 Software engineering v. systems analysis and design

As can be seen from the above descriptions of software engineering and systems analysis and design, there is considerable overlap between these courses. Both deal with analyzing requirements, budget, user needs, external and internal design, project management, testing, deployment, training, and maintenance.

However, there are three key aspects stipulated in the ACM/IEEE 2013 software engineering knowledge area that are not covered in ISU's SA&D course; these aspects have to be covered elsewhere in the CS department's curriculum in order for computer science to be able to use the SA&D course instead of a CS SE course.

First, CS students focus on developing systems software rather than simply application software, so they need to understand how to design systems with parallel and real-time aspects; the design approaches that work with typical business applications may not work well with other types of software. Second, CS students need exposure to considerably more object-oriented design than can be taught in a SA&D course with CIS students. Third, computer scientists feel that software project management is best learned as part of a team building a fairly large software package. This is because students need to be able to see the correspondence between their internal designs, resulting software, and project outcomes before than can learn to improve their design skills. The ACM/IEEE 2013 curriculum states: "students learn best at the application level … by participating in a project". While the systems analysis and design course involves a semester team project made up of a series of deliverables, it emphasizes the analysis and design of a system and stops short of implementation.

5 Systems analysis and design in CS

Based on the preceding analysis, it was decided that instead of a conventional computer science introductory software engineering course, ISU's CS curriculum would require the CIS systems analysis and design course. CS students are expected to have first completed CS-I, CS-II, computer organization, and advanced English class to insure they have sufficient writing ability. The CS-I course, requires structure charts [13] for many of the assignments. CS students also take a third programming class that focuses on advanced object oriented design and development and includes a final project that requires extending an existing system of about 25 classes with new features and an additional 15 classes or so. Finally, there is a capstone course titled "advanced software engineering and senior project" where they are required to analyze, design, develop, and test an extensive system as part of a team project.

The SA&D course involves a team analysis and design project with a series of scheduled deliverables. Students choose their own teams, with a peer evaluation at midterm that functions as an early warning system – and a second peer evaluation that is factored into the final individual grade. Peer evaluations involve each team member submitting a peer evaluation form assessing the contributions of each team member with regard to the percentage contributed by each team member toward the successful completion of all phases of the project.

While ISU's systems analysis and design course does not cover all the software engineering knowledge outcomes in the ACM/IEEE 2013 curriculum document, we feel it falls within the broader, historic tradition of much of what is covered in software engineering courses, and combined with other required CS courses, provides a solid software engineering background as well as a more extensive SA&D experience.

6 Experience and feedback

We now have had CS students taking our SA&D course for several years. Feedback from CS students is positive. One group ended up with a mix of CIS and CS students that reported working well together. The fact that the course focused on analysis and design techniques unknown to either CIS or CS students and did not involve any coding put both groups of students on a somewhat even footing. One student commented, "We didn't segregate jobs based on major either, because we were all learning the same material for the first time."

Having the SA&D course taught by someone with a MS in computer science as well as a PhD in CIS and extensive real-world software development experience was mentioned as significant. The software engineering knowledge outcomes incorporated into a number of required CS courses as well as requiring the SA&D course seems to have provided a solid practical SE background for the students.

Our experience has shown that while many CS graduates enter the workforce as systems software developers, the bulk are hired by businesses to build

applications software. For the latter group, experiencing a systems analysis and design course has proven invaluable, providing knowledge of not only software development techniques, but also familiarity with the business context inherent in a systems analysis and design class. This makes such students better prepared to excel in the workforce. CS majors have gained exposure to concepts like business rules, varied stakeholders, and business requirements elicitation. They have learned the importance of accounting and financial data, and gained a better awareness of the importance of written and oral communication. CS students have also be exposed to students with a business background, and, if a live project is used, exposure to an actual business. CS students have learned that there can be considerable differences between business application software – which provides the bulk of employment for CS graduates – and the more traditional CS systems software.

Perhaps most importantly, they also have become aware of how critical application software can be to the success of a business, as compared to system software that may be used across a variety of businesses. Based on this, a number of CS students have decided to start taking business courses in addition to their CS majors and some graduates are in the process of completing their MBA.

While this single case is based on a university with a unique history and with a faculty with a specific skill set, the outcomes reflect a broader question: Is the strict division between CS and IS relevant to the modern graduate? It has been shown that very little of the pure CS program is lost through the substitution that has been tried. Undergraduates value their exposure to a broader curriculum and contact with both business students and faculty. At least in our alumni community the tasks of the recent graduate are not those of the traditional CS graduate but require a better knowledge of the context of the systems being developed. We are sure that this change has been beneficial to our students and we suspect that this alternative would be of benefit elsewhere.

7 References

[1] ACM/IEEE Computing 190 curricula 2001 final report. http://www.acm.org/sigcse/cc2001.

[2] ACM/IEEE-CS Joint task force on computing curricula 2013 strawman draft.

[3] Chen PPS The entity-relationship model – towards a unified view of data ACM TODS 1,1 9-36 Mar 1976.

[4] Codd, EF "A relational model of data for large shared data banks". 13, 6 377–387 1970.

[5] Bagui S, Earp R Database design using entity-relationship diagrams, Second Edition, Boca Raton, FL: Taylor & Francis 2012.

[6] Ramesh B, Pries-Heje J, Baskerville R Internet software engineering: a different class of process. Annals of Software Engineering 14 169–195 2002.

[7] Austing R H et al., Curriculum ’78: Recommendations for the undergraduate program in computer science – a report of the ACM curriculum committee on computer science, Comm. ACM 22,3 147-166 Mar 1979.

[8] ACM/IEEE-CS Joint Interim review task force. 2008. Computer Science Curriculum 195 2008: An Interim Revision of CS 2001.

[9] Tomayko JE Forging a discipline: An outline history of software engineering education, Annals of Software Engineering 6, 3-18 1998.

[10] Brooks FP. The mythical man-month Reading, MA: Addison-Wesley Publishing, 1975.

[11] Atchison WF, ACM curriculum committee on computer science. Curriculum 68: recommendations for academic programs in computer science. *Comm. ACM* 11,3, 151-197, Mar, 1968.

[12] Spencer JW, Grout JC Systems analysis and design in a computer science curriculum, SIGCSE Bulletin, 24-27 1978.

[13] Yourdon E, Constantine LL Structured design: fundamentals of a discipline of computer program and System Design Yourdon Press, NE 2ed 1978

[14] Poole BJ, Callihan HD Systems analysis and design: an orphan course about to find a home SIGCSE bulletin 20,2 54-64 Jun 1988.

[15] Encyclopedia Britannica. Encyclopedia britannica online academic ddition. Encyclopedia Britannica Inc., web, Mar 2013.

[16] Shoorman ML The teaching of software engineering SIGCSE 83, 15,1 66-71 Feb 1983.