Are Academic Programs Adequate For The Software Profession?

Alexis Koster, San Diego State University, USA

ABSTRACT

According to the Bureau of Labor Statistics, close to 1.8 million people, or 77% of all computer professionals, were working in the design, development, deployment, maintenance, and management of software in 2006. The ACM model curriculum for the BS in computer science proposes that about 42% of the core body of knowledge be dedicated to software engineering, including programming. An examination of the curriculum of a typical computer science department shows that, excluding programming courses, no courses specific to software engineering are required for the BS, although several are available as elective courses. Academics typically resist the demands of the industry, in part because some of them are for specific software tools, design methods, or programming languages whose use does not last. Under market pressure, more required software engineering courses may slowly be included in the curriculum. The usual solution is for businesses to offer their software professionals needed courses in software engineering.

Keywords: Software engineering, software professionals, programming languages, Java, computer science curriculum, MIS curriculum

THE JOB MARKET FOR SOFTWARE PROFESSIONALS

The Bureau of Labor Statistics (BLS), an agency of the Department of Labor, reports the following number of US jobs for the various software professionals in 2006 [4]:

Software Engineers	857,000
Computer Systems Analysts	504,000
Computer Programmers	435,000

It also reports 542,000 jobs for various computer scientists, allocated among networks and communication analysts, database administrators, researchers, and other computer specialists. The functions of various computer specialists may require in part work on software, whereas work on software is the essential part of the functions of the first three categories.

For software engineers, the BLS expects a 38% increase in the number of jobs through the year 2016. For computer systems analysts, a 28% growth is predicted the same period. For computer programmers, the BLS predicts a 4% decline, due to foreign outsourcing as well as to a number of technological factors. For the computer scientists, there will be a 37% growth.

One may question the BLS distinction between software engineers, computer systems analysts, and computer programmers. Quoting the BLS definition, ”computer software engineers begin by analyzing user’s needs, and then design, test, and develop software to meet those needs.” They may also write programs, “but this is usually the responsibility of computer programmers.” This shows that there is often an overlap between some of the functions of software engineers and of computer programmers

Similarly, the BLS indicates, among various functions of systems analysts, that they “consult … users to define the goals of the systems. They prepare specifications … for computer programmers to follow.” This

description could also be applied to software engineers. A distinction between software engineers and systems analysts appears to be that systems analysts may be involved in the full analysis and design of computer information systems (hardware, software, overall architecture) whereas the software engineers are involved only in software.

Finally, one can assume that the BLS statistics are based on data the Bureau collects from businesses and organizations. Since there is no standard definition of the three categories of jobs, businesses and organizations provide jobs data on the basis of their own understanding of those terms and on the basis of the job titles they use. In the remainder of this paper, the phrase "software professional" is used to refer to those three categories.

In most of those work categories, a bachelor's degree is required. But an associate degree is sometimes acceptable, particularly for computer programmers. The BLS indicates that more than 80% of the software engineers and 68% of the computer programmers hold a bachelor degree or a graduate degree. The majors for most of those degrees are computer science and management information systems. Other fields are also found, such as mathematics. Systems analysts working in specialized areas of business sometimes hold a degree related to those specializations such as finance, but management information systems appears to be the major much in demand for business.

THE ACADEMIC VIEWPOINT

The main two fields of education for software professionals are computer science (CS) and management information systems (MIS). Not surprisingly, employers often complain that job applicants and new hires are not well prepared to work for them. Similarly, people working in the software field report on topics that would have helped them in their current job [13].

There are two aspects to those problems. On one hand, the software field has seen a large number of various programming languages, tools, and methodologies that were popular for a short period of time, and then disappeared, sometimes as fast as they appeared. On the other hand, if we look at the topics that computer science and management information systems programs teach, they often offer some very specialized topics that most software professionals are unlikely to ever need, and could be replaced by more useful topics.

Programming languages provide a striking example of both types of problems: languages used in industry and not offered in many curricula; and languages offered in curricula, but not used in industry. Cobol falls in the former category. Although it was one of the languages most often used in business applications for almost 40 years, it was rarely taught in computer science departments (but it used to be taught in MIS departments). Dijkstra, one of the most respected computer scientists of his time, wrote of Cobol that "its teaching should … be regarded as a criminal offense." [6]

On the other hand, computer science departments, for their first programming languages, have taught languages that are not used in industry: for example from Algol [3]in the sixties to the Turing programming language [9] more recently. The proponents of such languages for the CS curriculum justify their choice on the argument that these languages are well designed and, as such, are good vehicles for the teaching of sound programming. However, as indicated by Robert Floyd, "programming languages typically encourage the use of some paradigms and discourage others" [8]. Therefore, it is better for students to learn programming languages that they will use at work. Java satisfies both sides: it has become the main language for developing web-based applications as well as systems software; it is also a well-designed object-oriented language adopted by most CS and MIS departments as their introductory programming language.

Many undergraduate programs in CS and MIS adhere to some extent to the model curricula developed by the Association for Computing Machinery, the Association for Information Systems and the IEEE Computer Society [1]. A new report for the CS curriculum was published in 2008 [2]. It defines 13 broad areas of core knowledge and indicates the minimum time that should be spent on those areas. The areas that seem directly relevant to the software profession include Programming Fundamentals (16%), Net-Centric Computing (5%), Programming Languages (7%), Information Management (4%), Software Engineering (10%), for a total of 42%. As expected, this

body of core knowledge includes theoretical areas, such as Algorithms and Complexity (10%), Discrete Structures (15%), Artificial Intelligence (3.5%), as well as areas related to hardware and computer architecture (18%).

Ultimately, the BS curricula of specific CS departments are based on a combination of many factors: faculty preferences, influence of graduate curricula, traditional specializations, resources, requirements imposed by the university, and other scheduling constraints. The Computer Science Department at San Diego State University is a typical department. Its BS requires 10 specific courses in computer science (a normal course is taught 45 hours during a semester). Five of them are directly related to software design and development: two courses on programming, using Java, one on data structures, one of programming languages theory, and one on systems programming. Moreover, students are required to take 3 more electives in computer science, chosen from a fairly long list.

We note that the required courses are more about programming than software engineering. A student interested in software engineering could take the three required electives among purely software engineering courses (Software Engineering, Component-Based Software Engineering, and Software Measurement) and advanced programming courses.

Yet, several topics, which are important to software professionals, are not taught at all in this program. Those topics apply mostly to large software systems, but some of them are also relevant to small software systems. They include [5, 7, 12]: structuring of large systems; process of software building; software testing; software deployment; software maintenance; and interaction of software designers/developers with customers

For example, software testing is not mentioned in the SDSU catalog description of software engineering courses, although some instructors may discuss it. The 2008 ACM model curriculum indicates that about 10% of a course on software engineering should be dedicated to software testing. Given the critical need for software testing as well as its complexity [10, 11], this seems too little.

DISCUSSION AND CONCLUSIONS

A comparison of the undergraduate computer science and management information systems curricula offered by American colleges with the needs of the software profession show that some of those needs are not satisfied. Under the job market pressure, there have been efforts to add more required or elective software engineering courses to the BS curriculum. This is not necessarily easy because software is competing with an ever-growing number of computer science topics. Also, as discussed earlier, academics are reluctant to add to the curriculum topics, software design methods, and programming languages that may be short-lived. Until universities are able to satisfy better the needs of the software industry, software organizations will continue to enhance the skills of their workforce by providing their software professionals in-house or outside courses and seminars.

AUTHOR INFORMATION

Dr. Koster has been on the Faculty of San Diego State University's Information and Decision Systems Department since 1983. His research interests include software design, database systems and most recently -- music file-sharing, on which he has published many papers. He holds a Ph.D. from the University of North Carolina at Chapel Hill.

REFERENCES

1. ACM, AIS, and IEEE-CS. *Computing Curricula 2005- The Overview Report.*
2. ACM and IEE-CS. *Computer Science Curriculum 2008: An Interim Revision of CS 2001*.
3. Backus, John. "The syntax and semantics of the proposed international algebraic language of the Zurich, ACM-GRAMM conference." Proceedings of the ICIP Conference, Paris, June 1959
4. Bureau of Labor Statistics. *Occupational Outlook Handbook*, 2008-2009 edition. www.bls.gov/oco
5. Brown, William, Hays McCormick and Scott Thomas. *AntiPatterns and Patterns in Software Configuration Management*. Wiley, 1999.
6. Dijkstra, Edsger . "How do we tell truths that may hurt?", *SIGPLAN Notices* 17(5), May 1982

7. Heydon,. Allan, Roy Levin, Timothy Mann and Yuan Yu. *Software Configuration Management Using Vesta*. Springer, 2006.
8. Floyd, Robert. "The Paradigms of Programming." *Communications of the ACM*, 22 (8), August 1979
9. Holt, Richard and James Cordy. "The Turing Programming Language." *Communications of the ACM*, 31(12), December 1988
10. Kao, David and Ken Koster. "State Coverage: A Structural Test Adequacy Criterion for Behavior Checking." Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Dubrovnk, Croatia, September 2007
11. Koster, Ken. "A State Coverage Tool for JUnit." 30th International Conference on Software Engineering, Leipzig, Germany, May 2008
12. Miller, Roy. *Managing Software for Growth*. Addison-Wesley, 2004.
13. Plice Robert and Bruce Reinig. "Aligning the Information Systems Curriculum with the Needs of Industry and Graduates ." *The Journal of Computer Information Systems*, 48(1), 2007