

Software and Higher Education

John C. Knight Nancy G. Leveson

With software playing an undeniably critical role in our lives, one would expect that the best engineering techniques, such as rigorous specification and systematic inspections, would be applied routinely in its development. But in our experience, the opposite is often the case. Many large and important software development projects are conducted with poor choices of engineering techniques and technologies—resulting in increased risks, costs, and delivery delays.

The reason that this situation has arisen is that those making technical decisions often do not have the necessary engineering training needed to make good decisions. The blame for this limitation lies mainly on deficiencies in the computer science and computer engineering degree programs to which many people turn for their professional education. Here are some deficiencies that we have noted:

- There is too little emphasis in these degree programs on the principles of software development; the emphasis at present is on popular detail and not principles. Important topics such as specification and testing techniques tend to be taught rarely and superficially if at all. Graduates know the syntax of the language du jour and details of the associated libraries, but not how to work with other engineers to specify, design, or test a system. Coverage of all the major elements of software development (including requirements, specification, design, and verification) must be included in degree programs, in such a way that graduates understand what choices are available and how to use them.
- Education about the role of computers in systems is frequently limited to web applications and simple “isolated computer” assignments. It is essential that students get training in working with other engineering disciplines. This should take the form of examining significant example systems (both good and bad), as well as multi-disciplinary projects. Students might also be encouraged to take courses in other disciplines so as to see examples of how computer systems are used.
- Many topics are taught from too narrow a perspective. Student exposure to software design, for example, is often limited to a shallow treatment of object-oriented design; graduates are often unaware of the myriad of details of object-oriented design or that there are other approaches such as event-driven systems, table-structured designs, and functional decomposition. A carefully selected set of techniques in each area needs to be presented to the student, and then the student needs to use at least one or two of the techniques to gain insight. It is also important for students to be exposed to appropriate comparative analysis of the techniques, and to have opportunities to discuss and evaluate the material in various engineering contexts.
- There are important areas of the discipline, such as real-time and embedded systems, that are seldom taught and almost never in depth. There is a perception that such topics are somehow specialized and rarely required. In practice, many computer systems are embedded, and many

do operate in real time. Degree programs must be broad enough to ensure that graduates understand the spectrum of systems being built and the techniques that they require. This could be achieved relatively easily if topics such as real-time systems were introduced as part of a more established topic such as design by showing how real-time issues impact design decisions.

- Finally, graduates have a professional responsibility to know their limitations as the developers of crucial artifacts upon which our future depends, but the details of this responsibility are not taught extensively or as a priority. As well as being taught the right principles in degree programs, graduates must also be taught what they do not know, what their professional limits are, and when to seek help from others.

Faculty involved in these degree programs should ascertain where their graduates go, and, if many go into software development, they should ask whether these graduates are receiving the right education from the courses the faculty teach. If they are not, then either the degree programs need to be improved or it needs to be made clear that the degree programs are not designed to prepare people for a career in software development. A better alternative might be for more institutions to do what a few have done already, develop degrees in software engineering.