About the author:

Biya Ni • Strauss special Rupp (Bjarne Stroustrup) Yes C ++ Designer and original implementer. But also " C ++ programming language"," C ++ Language design and evolution "," Programming Principles and Practice "and other works of many authors. Dr. Strauss is a special Rupp College of Engineering Chair Professor of Computer Science, Texas A & M University. He is a member of the US National Academy of Engineering, AT & T member, IEEE Members and ACM member. His research interests include distributed systems, prototyping, programming, software development tools and programming languages. He actively participated in the ANSI / ISO C ++ Standardization work.

Translation: Gongxiao Cong, Shibao Guang (Huazhong University of Science and Technology undergraduate, Dian Team members)

What Should We Teach New Software Developers? Why?

We should teach future software developer what? what is the reason?

---In order to better meet the needs of the industrial sector, we have to change the computer science education fundamentally.

In the R & D system, it should be at the center of computer science knowledge. Otherwise, we usually only rely on research and development during the rule of thumb, finally made out of system scalability poor, low reliability, high cost. Therefore, we need to change computer science education, to improve the software development level.

problem lies in

Between science education and industry demand for computer, often a large gap. Imagine the following scene: the

famous professor of computer very proud to say: " We teach not programmed, but computer science. "

Enterprise Manager: " They simply idiots in programming. "

In many cases, and not just right on the surface they are right. Academic work is to develop not only ordinary programmers; industry also requires more

than " Every aspect of high-level thinker " with " the scientist " . Another professor of computer science, said: " I never write code. "

Another company manager, said: " We do not hire graduates computer. Department of Physics graduates to teach programming, computer science graduates than teaching simple physics more. "

Each side, but too idealistic, and there are misunderstandings. Professors can not teach those who do not own practice (in many cases is never practiced) and nothing to understand; and for business managers, software quality requirements only in exceptionally low, so that even the Department of Physics Students (and some Department of computer Science students are not trained) can handle such a case, if he is right. Of course, some physics students is very hard work, beyond the control of both physics and computer science, they are not included - Such people have dual skills, in my opinion is the best talent. Professor of computer science (talk about a student): " He found a job in the enterprise. "

Another professor of computer science: " Too bad, he was then a promising student. "

This disconnect has brought a lot of problems, but also to solve a lot of trouble.

Industry computer science graduates want to write software (at least early in their career like this). This software is usually part of a long line of code libraries, and will be used in systems that require high reliability up, such as embedded systems and distributed systems. However, there are many graduates in addition to amateur programming hobby, there is no truly, received education or training in software development. What is more, many students just programming as homework trick, little attention has been on aspects of the system test, system maintenance, documentation and code readability. In addition, students can not be linked in a lesson learned knowledge and another course. As a result, we often see students get high marks in algorithms, data structures and software engineering exam, but they completely ignored data structures, algorithms and software architecture in the experimental class operating system, only to copy off the shelf method, the experimental results mess. For many people, " program " It has become tinker with the code, calling library functions in conjunction with others, these two acts. The principle behind it, they know almost nothing about. " system maintenance " with " Code quality " The concept they understand very little, or even ignore. In the industrial sector, a lot of people complain about not find can understand " system " with " Software builds " It graduates. This complaint reflects a real problem.

My computer did not crash it

We often complain about software quality. But in the past decade, a lot of software has been greatly improved. Unfortunately, this improvement is costly. This is reflected in the price of energy spent on human and computer resources consumption. We never learned how reliable

Part of the overall formation of a relatively reliable method is to continually add runtime checking, and build a large number of test code. Sometimes even the code structure itself is changed, but not be improved much better. Excessive levels of software, complex dependencies, so that no matter how capable a person is difficult to understand the whole system, it's not a good sign: Perhaps in the future, we can not understand their own software system, can not test a key part of the system.

Of course, there are some system architect, to build them under pressure against the continued expansion of the scale, it is difficult to understand the system. When the plane did not crash, but also by telephone, e-mail also arrive on time, we have to thank them. They are commendable, their efforts to make the formation of a mature and reliable software rules, tools and techniques. Unfortunately, they are only a minority. In most cases, the expansion of software building dominates the way people's minds and thinking.

Similarly there are some educators who oppose the theory and practice of industrial separation. They also deserve praise and support. In fact, I know every educational institution has a number of projects, which aims to provide practical opportunities for students; some professors have also put a lot of effort to succeed in some projects. But from a broader perspective, I do not feel so good - Practice project or internship course is a good start, but in a comprehensive approach to teach students that they are not a substitute for a balanced curriculum system. Compared to " computer science " ,More attention " Software Engineering " or "IT" Such a thing, which perhaps reflects the changes in the teaching focus. However, to solve the problem, such a change may still be a new name. I'm right " industry " with " academia " Description irony. But I believe, more experienced people will realize that this reflects the reality of which is the problem. I'm an industry researchers and managers (I AT & T Bell Labs work

twenty four Years, 7 In a department manager); I also worked in academia 6 In (Department of Computer Science taught at an engineering school). I travel often, we have a very serious discussion with staff and management technology layer from dozens of companies (mostly American companies) per year. I found a college student can not meet the industry's training needs, which are a serious threat to computer science and computer industry is concerned.

Disconnect between academia and industry

So how do we do it? Industry tend to hire people who have received training in new technologies and tools of " Developer " It, and academia biggest goal is to train more professors. We have to adjust these goals to achieve greater progress. Graduates about to enter the industry who want to better understand software development; industry have developed better mechanisms for they can absorb new ideas, new tools and new technologies. Make a good developer to enter into a business, the results of this business aims to avoid mid-level programmer has made a technical error, this is no good, because it would limit them to try anything new and better things. Then talk about the scale of the problem the program. Many industrial systems contain millions of lines of code. However, even a 1000 No program lines of code written student could graduate from computer science. Major industrial projects involve a lot of people, programs and practices in school the importance of personal effort, thus impeding the teamwork. Many organizations aware of this, I began to focus on simplifying the tools, skills, language and processes to minimize dependence on developers skills. This is a waste of talent and energy, so that each person's role will be reduced to the point can not be cut. Both rely on the industry " Practice has proved correct " Tools and techniques, but also indulge in the dream " Panacea " , " Revolutionary breakthrough " , " Killer application " ,and many more. They want low level of technology, developers can be replaced at any time, a small number of cattle to be bothered to focus on code quality " dreamer " Guided. This causes them extreme conservatism on the basis of selection tools (such as programming languages and operating systems). This will result in a large number of proprietary code and are not compatible with the infrastructure to be developed: they need something beyond the bottom, so that developers can develop applications. Although there are similarities between the basic tools, but the platform vendor lock-in developers. Reward systems need to plan for the long-term and short-term results of an enterprise. The resulting costs, as well as the failure rate of new projects, are staggering.

The face of the reality of both industry and other similar obstacles, but the academic seclusion, doing what they do best: a small group of like-minded people together, study the phenomenon can be observed in isolation, to establish a solid theoretical foundation, to question idealized construct the perfect solution. Those used to deal with a large, old-style code proprietary tools, do not fit this model. Like industry, the academic world should also have their own incentive system. All this may well improve the quality of academic research. Therefore, the results of the needs of academia and industry as the square peg in a round hole as out of step, the industry needs to pay a high price for this purpose is the development of training and infrastructure. The total was suggested: " As long as those companies can pay to develop a sufficiently high wages, the problem is solved. " This approach may be

Help, but to pay more wages for the same work will not be much effect. One possible solution is that companies hire better developers. The software development as production lines, is a developer and general level may be replaced on the assembly line, the idea is to have a fundamental problem, which will bring the waste of resources. It is the most competitive people out of the industry and let the students do not want to enter. To break this vicious circle, academia must produce more graduates with relevant skills, tools and industry must be able to use these skills, technologies and processes.

Ideas on specialization

" computer science " Is a bad, there is the term misleading. Computer science is not primarily about computers, not primarily a science. Rather, it deals with the use of computers, and the way computing (algorithms and quantitative thinking) about the way of thinking and working. It combines all aspects of science, mathematics and engineering, and are usually held together by computer. For almost all computer science, it is " Pure computer science " , The actual application is out of line. A computer science professional people, and people in other areas (such as medicine or physics). When developing an application, how to distinguish between them? The answer must be " Mastery of core computer science " , Then this " core " what is it then? It contains many set up in the Department of Computer Science courses - Algorithms, data structures, computer architecture, programming specification, mathematics (mostly quantitative reasoning) and systems (e.g., operating systems and databases). In order to integrate knowledge and inspiration to solve difficult problems, each student must complete a number of projects require teamwork (you can call it junior software engineering). The need for a balance between theory and practice, this is a very basic principle. Computer science is not just theory and principles, not just tired code. with " Compute " This whole field compared to the core is significantly more computer-oriented. So no one expertise (such as graphics, networking, software architecture, human-computer interaction and security) of the computer field, can not be called a computer scientist. However, these are still not enough. It is applied and interdisciplinary, therefore, each of the graduates in computer science, should have some other areas (for example, physical, medical engineering, history, accounting, French literature) knowledge on the nature of the practice of computer science, and It can be achieved and a minor in the same level. Experienced educators will find: " It is impossible, almost no student can master those within four years. " They are right, we need to make some compromises. My advice is, first degree in computer science master's degree should be, and the entire system is in accordance with the curriculum to train master for the purpose of designing, rather than undergraduate postgraduate level plus one or two years. Determined to academic research student then after a master's degree to pursue a doctorate. Many professors will object: " I do not have time to write the program! " However, I think that those educational software development professionals ready to become students of professors, to take time to write the program, educational institutions, where they should also find ways to encourage them to write programs. The ultimate goal of computer science is to develop a better system. Let not seen a patient's doctor to teach students how surgery, or never touched a piano teacher to teach students to play the piano keyboard, you feel fly it? Computer science education must enable students to stand on a higher level than book learning, enabling them to master knowledge in the practical application of a complete system, and the ability to appreciate the aesthetic value of the code. I used " professional " Such a word, it is a word of many meanings and has implied. In areas such as medicine and engineering, which means that certification. Certification is a very difficult and emotional topic. However, our civilization depends on software. If anyone can based on personal preferences, or the company's decision to modify an important part of the code, so that it is reasonable? If it is now, that 50 During the year it is still reasonable? Those millions of people use the software, if there is no quality guarantee, reasonable it? The real question is, professional recognition through certification achieved, dependent on a lot of knowledge, tools and techniques. Engineers have a certification, we can ensure a building built up with appropriate materials and techniques. However, computer science students should have what qualities, there is not a universally accepted list, I do not know what to do. Now, I do not even know how to pick out a group of people to design a certification exam.

Industry how to do is to narrow the gap between academia? Compared to talk about academic, drawing " industry " with " Industrial demand " Much more difficult. After all, academia has a fairly standard structure and fairly standard approach to achieve its goals. The industry is more diverse: some small, profitable there are non-profit, the methods used in the construction of the system has generally also have high-end, and so on. So, I am not a little out of line to eliminate this possible way. However, I have found the problem that caused the disconnect: Many obviously very dependent on the computing business, but the low level of technology was terrible. Industry manager: " Introduction of professional knowledge and skills, vital to the survival of the enterprise. "

No " Institutional memory (Institutional Memory) " And set to tap and recruit talented person's system, no organization can succeed. Enhanced cooperation with the business community interested in the academic software, it may make both more effective. Joint research, as well as courses beyond the learning of lifelong learning stressed that may play an important role in the cooperation.

in conclusion

We must do better, otherwise we will continue to go wrong equipment, expansion, resource-intensive. Finally, some parts stop working with some unpredictable and catastrophic way (think network routing, online banking, electronic voting and grid control). We have to make academia and industry have to make a change so that the disconnect between them can be reduced. My advice is to be based on computer science education professional core part, and with some direction and goals of professional knowledge, education partial application of the software is to make students able to obtain certification, at least in part can be achieved. This may require long-term cooperation to industry and academia to achieve.

Original Address: http://cacm.acm.org/magazines/2010/1/55760-what-should-we-teach-new-software- developers-why / fulltext