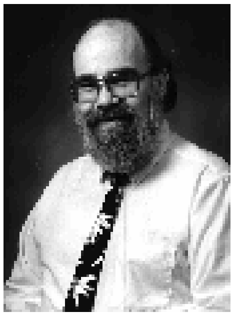
Crossroads
The ACM Student Magazine



Should computer scientists worry about ethics? Don Gotterbarn says, "Yes!"

by ***Saveen Reddy***

Editor's Note: ***This article originally appeared in Crossroads in 1994.***

Professor Gotterbarn is currently in his fifth year at East Tennessee State University where he helped develop their Master of Software Engineering program. He previously taught at The Wichita State University, Allegheny College, and the University of Southern California. He also worked as a computer consultant at the Philadelphia Navy Yard and with the Saudi Arabian Navy. He enjoys painting in watercolor, singing in barber shop, and hiking. Professor Gotterbarn is a member of the ACM and the IEEE Computer Society. He is co-chair of a joint ACM/IEEE task force on establishing standards of practice for computer professionals.

In computer science why should we take any particular pains to consider social issues?

There are a number of arguments.... One, is that as you gain special knowledge and special talent in society, along with that power comes responsibility to use it for the good of that society.... Many people in computing have not even raised the issue of social responsibility. They have bought into the view that "Computer science is a theoretical discipline with no moral consequences." Some argue that ethics is irrelevant to theoretical disciplines. You don't find an ethics of mathematics course. You might find an ethics of statistics course. I'm not talking about whether statisticians are moral or not. I'm talking about the highly likely possibility that there can be ethical applications of statistics. A theoretician might say "What do you mean ethical applications? You either do statistics or you don't." When you talk about computer science and software engineering, you find the same mistaken view. We computer scientists kept CS separate from actual applications or engineering and we were more interested in theory and efficiency than in applications. That's what we teach our students. When you take a data structures course ... We just ask "Can you figure out the computational complexity?..."

The problem is that we don't emphasize that what we build will be used by people.... I want students to realize what they do has consequences. That is something we don't touch.... Why be moral? Why build the best product you can? Our products drive cars, control life preserving and life taking devices. As software developers we have a moral obligation to put quality, which is more than mere efficiency, into our products.

When I talk about this type of responsibility to industry, people sometimes say, "Well it will cost us more to do the required testing." I answer, "Yes, it will cost you more in the short run to build a better product, but in the long run you and your customers will be better off." To see this, all we have to do is point to the automobile industry in the United States, and some people will say that if you buy automobiles from some foreign country ... you will get a better automobile. If you ask them "Is it a cheaper automobile?" the answer is "No." It costs more. What people want is that quality product. That's not an ethical justification for "Why be moral?" It's a very pragmatic justification for "Why be moral?" and I think lots of companies have realized the applicability of this justification.

I'm confused about your use of "ethical" and then "moral."

I'm not being a good philosopher here and maintaining the distinction between ethics and morals. When you talk about ethics you tend to talk about a public system of human interactions recognized as appropriate for some group. So, people can talk about professional ethics as a generally agreed upon set of behaviors for a particular group. Physicians have a public system of medical ethics, and because they face many confusing and tough questions they publish opinions of other physicians about them … every year in the AMA Code of Ethics, Current Opinions. It gives you examples of how to apply the code of ethics to tough cases. It is public and demonstrable. Engineers have a code of ethics. Folks don't realize that the ACM has a wonderful code of ethics accompanied by guidelines for professional practice. The guidelines are there to help you understand the code and use it in decision making. So, one of the principles of the code says. "Do no harm," and that is a nice generic type of term. What does it mean? Go to the guidelines. It says "Anything which will cause people to have their businesses be destroyed, lower their stature as human beings... anything that could cause physical pain and anguish," and there are examples.

If someone works in the military their goal is exactly to kill and destroy. How do we reconcile this with "do no harm?"

Maybe the goal is to get others to submit. I try to get my children to submit, and I try not to do it with a two-by-four.... You have to... balance the other principles against "do no harm." Probably, 75% of my consulting was done with the US government.... I was also connected with the nuclear submarine division.... There was no problem in justifying that. I as an individual happen to believe in a strong defense. I don't believe you have to convince others by actually using the whip. If you're carrying one that's enough. That is one way to balance things. We get into ethical problems where sets of these principles begin to conflict. For example, the code of ethics says you ought to abide by the law and fulfill your promises and contracts.... Dave Parnas made a commitment to help develop the Strategic Defense Initiative. He looked at it like a good computing professional.... How the blazes do you test a system like that which is designed to protect the US from massive attack? Well, the answer is wait until you are attacked, but that is to not test it. That was unsatisfactory as a computer scientist. In his role as a computing professional he raised that issue.... That it got raised changed the whole tenor of the project.

There's a trend to have projects developed by geographically distributed groups. They might not even be in the same country. How does this affect us if they may not even realize the extent of the project on which they work?

That there are difficult cases we can come up with does not mean that in general we can ignore our ethical responsibilities. … When you are working on a project that has certain kinds of ramifications, there are standards that we apply that have significant consequences.... When I was working for the submarine division I knew that it was the nuclear submarine division. I knew what sorts of things they did. I could have then done a number of things. One was to continue to work for them and do any kind of job I decided to do.... I could have decided I didn't really like nuclear warfare and stuck a virus in their programs which would move reactor rods at the wrong points. I could have left the company in protest. Those are sets of options, but in all of them I am thinking in terms of my responsibility for my actions and their long range ethical impact. That is different from thinking about software development merely like a crossword puzzle to solve, where I do the solution throw it away and go on to the next one.... The guy who writes the program for a toggle switch doesn't know whether it is for a bomb site or for a car. Admittedly that is a problem. You can't answer higher level ethical problems there. You have probably heard the excuse for shoddy work from fellow students when they are having problems coding "All programs have bugs." That's unacceptable. It gets used as the generic excuse to say, "I'm not responsible. Take my shoddy work." And if you recognize your responsibility there's a difference in just the way you'll do that work.

Even if everything is done correctly, we really cannot guarantee any action one hundred percent.

You just said, "There's bugs in every program." I say, "Piffle." ...When I talk to groups I ... find there is actually very little discussion about testing in programming classes. This is an absurdity. The issue is not just to solve the problem. It is to solve the problem reliably, maintainably, and so on. We have to teach testing. You say you can't guarantee a product. I can. In fact, you have contracts where the government certification is to a standard called 4-9s standards of testing which is 99.99% reliability -- still not 100%. But, its better than 40%. "You can't certify 100%" gets interpreted incorrectly. This is the game called the black and white fallacy. It says there are only two values in the world: one for true and zero for false. So, if you say the program is reliable you must mean "one." Since you can't say "one" then it must be "zero." We can do better than that. I can give you the techniques to get as close to "one" as possible.

What are those techniques?

Various kinds of testing techniques. ... I'm a software engineer. So, I believe reliability and quality starts long before coding. I'm going to spend 40% of my budget getting the requirements right. Then, I am going to use different types of requirements reviews, design reviews, things of that sort. ... One, I use these things to get the product that you want. Two, there's a responsibility I also feel I have: If the product that you think you want is not one that I can make reliably, then I should tell you. If I cannot make it reliable and it cannot not be made reliable then I should warn you of the dangers in pursuing this.

What problem should one have if a program one creates goes horribly wrong? There are risks and everyone knows that. We accept the risks of vaccinations for instance. Even that does not guarantee patient survival.

There are those kinds of limits. You've followed out your moral responsibility... I'm not trying to lay a guilt trip on the world. ...You have the responsibility to produce this product using the skills that you have. If you take on a job you ought to have the whole set of skills. I wrote up a case in the SIGCAS newsletter several issues ago about a person who was contracting with the military. The person had designed a system ... and the military representative thought they knew how to design better than the consultant and they wanted it redesigned. It was a defense system very dependent on interactive communication in a very hostile environment. It was a very difficult process. The military person said, "Design it this way." The consultant knew that it would have been much harder to test it that way. This was an absolutely life critical system. The consultant was caught in a difficult situation. The military representative said, "Do it my way or the project will be canceled." The consultant knew that if it was canceled the military individual would move on and disappear and his replacement would not be able to pick up the project again because it would have been a canned project. That is a very tough ethical situation.... The consultant had all sorts of conflicting issues, and I do not have easy answers. He had all sorts of interesting obligations. If he refused to do it the way the military representative wanted, that would violate the contract. What would happen to his employer/customer as a result? The customer was being warned but then was being unreasonable. If this system was not in place many people would then die. If even a marginally defective system was present, then many people would also die. Now the consultant has other responsibilities, "You have the contract," "do no harm," etc. Thinking about these issues is something you have to develop. All these things are part of your responsibility.

Let us suppose a product fails catastrophically and many people die horribly. Who takes responsibility in the chain of command and what are the charges?

We have to fill in the details to answer your question. At this moment, different courts in the land would give 18 different decisions.... As computing has grown, you have gotten varieties of different publicly asserted opinions. Some are well founded. Some are just to get press. Judges are well schooled in their discipline but are not computer scientists. We have very few public statements. If as an engineer, you built a walkway in the Hyatt hotel in Kansas City and used a certain type of coupling, I can tell you that's wrong, because the book says not to use that kind of coupling. If you are going to guarantee for me 4-9s safety and reliability and you didn't use boundary analysis; if you just tested one variable dead center in the range and did just one test, then you are guilty of malpractice and negligence.... There was one case where this guy stuck a trap in a program. He was let off. The judge said he's like a used car salesman. He's not a professional. He can't be held to any professional standards. There are two cases in California of computer programmer malpractice where they hung them out to dry for doing that kind of thing. Depending on what kind of damage you cause, you can be found guilty under tort law. I would think in a situation where people ignored obviously well known software engineering principles, you could be found guilty under tort law -- provided that you did something knowingly, not that you are twit and don't know how to test.

Now to your question about a catastrophic failure. Suppose 20 people die and they are roasted to death or whatever. Chernobyl. Suppose you are the guy who wrote the controller for Chernobyl. And, you did it knowing that this was a life critical system.... Knowing that the speed of the rods was too fast and that would set off an uncontrolled reaction, and you did it anyway because you wanted to play golf on Thursday, then I think even in the current state of affairs you at the programmer level would be found guilty of malpractice. Who is the person above you that is supposed to be knowledgeable? We don't have one.

If in building an engineering product you went to an engineering school and you built the walkway in the Hyatt with the washer in the wrong position, you would be expected to know that, but the professional engineer, the certified licensed professional who has to sign-off on the fact that you did that, would probably be the target of that lawsuit. A virtue of being a licensed professional is that when someone in management says sign-off on it you can say, "I can't because it doesn't meet the safety standards, and because it doesn't, Boss, you don't want me to sign off because we'd all be liable." In computing we at this moment don't have that counter pressure. When someone says, "Do it," you can't say, "Someone will sue us because it violates standards."

Freshmen CS students don't even have a clear idea what their major is about. How can we instill in them this sense of responsibility?

What I am talking about is professional responsibility. Until you begin to understand some of what computing is about and how small things like rounding numbers makes a difference. You try to explain to a freshman class situations like this. There is a skyscraper in Chicago. These buildings sway ... and we have to put glass in them under certain parameters. When you deal with that kind of sway you have a function that keeps repeating. The difference between rounding and truncation becomes significant in cumulative calculations. An architectural program was developed and applied to a 100 story building. The glass was put in based on the parameters. Right after the building was built, the fraction of an inch difference in the glass because of truncation instead of rounding led to several sheets of glasses falling from the top floors and crashing to the streets. ... That's an example where they might think that's important. However freshman students think they will never be involved in such an event.

When I give my half hour ethics talk, to get the audience's attention I drag out all these catastrophes. Most people say, "I'm going to end up working for some bank, and this doesn't apply to me." ... Ethical concerns do not sink in at the freshman level. If you take a survey of a freshman class and they don't know much about computing, their ethical views are very disparate. With a sophomore class, all of a sudden testing begins to be important. With a junior class, they start worrying about design because someone put them on a team or they got the program back and had to modify it.... Computer science professors are in some sense unknowingly teaching ethics. They are teaching the ethics of being a good computing professional.

But how do you do it for freshman? I change the programming examples. Instead of doing some program about the three color problem, I find it interesting to have them do a control mechanism for an airplane's landing gear for instance. There is lowering the gear, locking it, turning on the light in the cabin indicating the lock. Those things must occur in that order. I don't have to preach a sermon....

Why is licensing and certification important?

Having standards is important for public reasons. People learn that a doctor is committed to apply antiseptic to a wound before they close it up. The lay person knows to follow those same standards.... The notion of what is professionally good enlightens non-professional work. The notion of what is professionally good in computing as far as testing can enlighten non-professional work -- people developing programs for their own use and their own business and so on. And no one can say, "Oh well there are no standards just do what you want." You'll have a model. So, it is good for what goes on outside of the profession. It is good for protecting the consumer. It is good for what goes on inside the profession because people cannot then easily slide into "all program's have bugs, so you can't complain that mine has some in it," because we will also have in those standards metrics that can tell you when you are producing too many mistakes. ...

What licensing does is encourage education. And I am an educator.... I approach licensing using the model of the paramedic. There are several models of licensing. One model says: "Go to school, take a test, and you can pay your fees and still be licensed." ... In the paramedic model, paramedics are trained and tested and three years later they will be tested again and they will be tested on the new material.... There are too many places where computing affects peoples lives. Having a licensing mechanism will help ensure that there is a group of people that have this knowledge. When you have applications that require this kind of certainty, you can require that it is signed-off, approved, or audited by a licensed computing professional. Licensing will educate the public, help technology grow, and you have a better likelihood of making a safe product....

Licensing seems like an very good thing. Despite the fact that ACM is 49 years old, it hasn't been implemented yet. What explains the delay?

... I don't think its a question of the ACM ignoring anything. We are a brand-new discipline.... The first thing we did in computing was print checks, write reports, accounting, and actuarial tables. For all of those things if you messed up, you could come back at night, think yourself a hero, and fix your mistakes. Modern applications now don't allow us to simply reprint reports. You can't re-run a pacemaker. It better work the first time.... Generally, most professional organizations are becoming more socially conscious. It is true of engineering and medicine. In computing we are beginning to realize our professional responsibility. In the early days of professions, like in engineering the professional societies were designed to further the ends of the professional society. You were not allowed to criticize other engineers. Lawyers were not allowed to criticize other lawyers. The first engineers to ever be thrown out of an engineering society for violating a code of ethics occurred in Los Angeles in early

1930's. Some engineers criticized other engineers for putting the wrong mix of concrete into the viaducts and they were going to collapse. They were whistle-blowers....

Now if you look at the code of ethics, the social consciousness is broader. It says, "You will try and further the ends of society by putting the *best* of your profession forth." The ACM code of ethics says if you see something going on which is low quality work or dangerous, it is your responsibility to do what you can to rectify the situation. The IEEE code of ethics says not only cause no harm, but also says if you witness potential for harm it is your responsibility to act positively to stop it. They do not further go and recommend whistle-blowing, but the implication is clearly there. So there is that change in the societies. We are doing things that are too life critical right now.... Both the ACM and IEEE have gone forward, learned, and are responding now. We work with this task force for establishing software engineering as a profession. When you talk to people about it there is not a question any more, they say "It's about time."

How long do think it will be?

Before licensing? What we are doing now is preparatory.... This is a committee that goes by the label of establishing software engineering as a profession. They are setting out the skills, the educational track, the ethical commitments and standards of practice for software developers. If you take any thing that is called a profession today, they have those three things. They draw a summation bar under those three things and say, "Therefore, let's do licensing." In computing, the summation bar has been mentioned but not drawn. I think material will be out within the next two years where this is listed. Then it becomes political and I have no guess as to the timing. All of the things I think are important -- licensing and why we should have licensing, will at that point have been accomplished.

So what form will licensing take. Will we have one license for all computing professionals or have sub-licenses? How will we deal with that?

I like the paramedic model for a number of reasons. One is that paramedics get licensed to do different things to people. There is a paramedic called a "first responder" who can ride in the ambulance and help carry people into the ambulance, but that can't touch the patient.... Another level can take vital statistics but that's it. At another level, you can administer medication. There's a top level where you can cut the patient if necessary. For computing, I don't know what these categories will be. People talk about "safety critical" but it gets crazy to talk about what is safety critical. The mechanism of an airbag is safety critical when you are driving. The mechanism of antilock brakes in safety critical. There is a recent case where a program that prints checks on-line proved to be safety critical, because its failure generated some riots in a welfare center. So, where to draw the lines is a problem.... What the categories are is something I don't know yet. I think you will have a generic kind of license like physicians. They can be a physician but then also be registered as a neurosurgeon or in some other area. That will go on. There will also be corporate product licensing....

Do you anticipate that every CS graduate will become licensed or only a certain subset of them?

The model will be just like the engineers. You can be various kinds of an engineer, chemical, electrical, and civil. That is through a period of education. You can be a licensed engineer by taking an examination, becoming an engineer in training, working for an engineering company for a while, and then taking your professional engineering test.... I think that's the way it is going to be for computer professionals.