What exactly is Software Engineering?

Once upon a time, in a kingdom far, far away, a king summoned three of his subjects. One was a coder, the other a computer scientist, and the third a software engineer. The king showed all three a shiny metal box with two slots in the top. "Behold", said the king. "This is my toaster. The first slice of bread is too light, the second just perfect, but the third is too dark. I want you to make my toast consistently cooked!"

The coder replied that he could easily fix it with a simple microcontroller. He said "Using a four-bit microcontroller, I would write a simple program that reads the darkness knob and quantizes its position to one of 16 shades of darkness, from snow white to coal black. The program would use that darkness level as the index to a 16-element table of initial timer values. Then it would turn on the heating elements and start the timer with the initial value selected from the table. At the end of the time delay, it would turn off the heat and pop up the toast. Come back next week, and I'll show you a working prototype."

The second advisor, the computer scientist, immediately recognized the danger of such shortsighted thinking. He said, "Toasters don't just turn bread into toast, they are also used to warm frozen waffles. What you see before you is really a breakfast food cooker. As the subjects of your kingdom become more sophisticated, they will demand more capabilities. They will need a breakfast food cooker that can also cook sausage, fry bacon, and make scrambled eggs. A toaster that only makes toast will soon be obsolete. If we don't look to the future, we will have to completely redesign the toaster in just a few years. With this in mind, we can formulate a more intelligent solution to the problem. First, create a class of breakfast foods. Specialize this class into subclasses: cereal grains, pork, and poultry by-products. The classification process should be repeated with grains divided into toast, muffins, pancakes, pastries, and waffles; pork divided into sausage, links and bacon; and poultry divided into scrambled eggs, hard-boiled eggs, poached eggs, fried eggs, and various omelet classes. The ham and cheese omelet class is worth special attention because it must inherit characteristics from the pork, dairy and poultry classes. Thus, we see that the problem cannot be properly solved without multiple inheritance. At run time, the program must create the proper object and send a message to the object that says, 'Cook yourself'. The object itself will know if it consists of toast, eggs, pastry, etc.

Reviewing the process so far, we see that the analysis phase has revealed that the primary requirement is to cook any kind of breakfast food. In the design phase, we have discovered some derived requirements. Specifically, we need an object-oriented language with multiple inheritance. I suggest C++, or maybe Java if we plan on re-hosting to newer hardware in the future. Of course, users don't want eggs to get cold while the bacon is frying, so concurrent processing is required, too. Maybe Ada 2012 is the way to go. And – of course – the unit will need an intelligent camera and a sensitive weight unit, so we can determine the composition, weight, and derived density of bread and other breakfast objects. This is going to require a lot of processing – I'm thinking a i7, maybe a 128 Gig Solid State Drive, 4 Gig RAM, and a rock-solid UNIX-based OS.

We must not forget the user interface. Users won't buy the product unless it has a user-friendly, graphical interface. We need a good OS – perhaps UNIX. Once the system boots – the system could display a pull-down menu. Users can pull down a menu and click on the foods they want to cook. Specific tailoring options can then be selected (raisin bread, cinnamon bread, etc.)"

The king asked the software engineer his opinion. The software engineer suggested that the king banish the coder and analyst from the kingdom, and he taught the king how to adjust the toaster – turning it up a bit for the first slice, down a bit for the second slice, and down even further for subsequent slices. And they all lived happily ever after. THE END.

What is software engineering? Well, I have taught "software engineering" at the college level for 29 years now. I would imagine, if you have made it to this point in this issue – you probably have an idea of what software engineering is for yourself. In my college classes, we teach such diverse topics as risk analysis, cost and time estimation, quality assurance, and configuration management. I also teach lifecycles, coding techniques and approaches (functional, object-oriented, etc.) I teach good documentation (lifecycle and code), design techniques, and how to perform testing. This is a required class; all Computer Science majors take my class.

I have the luxury of teaching a second class ("Requirements Analysis and Design") to the majority of my CS majors (not required – but over 90% decide to take the second class). In this class, we perform more in-depth analysis and design, and talk more about the non-coding part of developing software.

So, when they graduate – are my students software engineers? Nope – not even close. Because any project that I can comfortably cover in a one-semester course is way to small to give them a real taste of what the real world is like.

I heard from a lot of my former students after they have been "in the profession" for several years – and they always say "I though you were kidding with your stories. Actually, it's worse in the real world than we could have imagined.' However, the basic skill set I teach them seems to give them the tools and knowledge they need to keep them employed until they gain the experience they need. But the question remains – what are they gaining experience for?

Back to the original question: what is software engineering? It's the ability – through skills, experience and techniques – to organize chaos and create products that are as simple as they can be. That's what a software engineer tries to do – make things as simple. As simple as possible. This takes a lot of experience, luck, and intelligence.

To all my fellow practitioners - hard work, isn't it? And, to my former students – I was doing the best I could!

David A. Cook
Stephen F. Austin State University
cookda@sfasu.edu

P.S. I take no credit for the originality of the toaster story. It's been floating around the web for years.