If you're a developer and you're about to ask another developer a technical question (on a forum, via email, on a chat channel, or in person), *you'd* better be ready to answer the question "What have you tried?"

This of course isn't specific to software developers, but that's my field and it's thus the area in which I'm most familiar with the issue which motivated me to write this. I'm (sadly) quite sure that it applies to your own industry too, whatever that might be.

The thing is, there's a disease in the software development world; a sort of sickness. It's an unusual sickness in that it's often not something you acquire once you actually join the industry (like greying hair, caffeine addiction and an ulcer), but rather it's something that new recruits *already have* when they arrive.

Now, a quick clarification before I continue: when I say "new recruits", I don't just mean graduates and other young people. There are those who will say that this sickness is a product of modern western education systems, and that things were perhaps better back in the day. Maybe that's true and maybe it's not, but I'm not qualified to say and that isn't the position I'm putting forward here anyway. The illness I'm talking about seems to apply to both young and old alike.

The illness, of course, is a flawed approach to solving problems. Here's an example, which is an actual quote from a web forum:

> *1) Can we establish http connection in application.*
>
> *if so, i need that code.*
>
> *I checked NSURLconnection. I cannot intergrate that code.*
>
> *2) I want to display a image from the website*
>
> *Can anybody please provide me the code?*

If anybody having sample program please give me.

So where's the problem? It's not in the quality of English (it's reasonably evident that English may not be this person's first language, and that doesn't matter as long as the intent is clear - which it is). It's not in the punctuation and grammar, because those things again aren't particularly important in this context as long as they don't become barriers to understanding what's being asked.

The problem is that this person's problem-solving technique is to *ask for the solution*. Not to seek advice on how to approach the task, or ask for the names of likely classes to look into, or a link to an example - but to just ask for the code, fully formed and ready to go. This is *not* problem solving, and software engineering is *entirely about* problem solving.

The interesting thing to note here is that the above example isn't actually as bad as it could be; there's one tiny glimmer of light to be found in the assertion that this person "checked NSURLConnection". That inspires some small amount of confidence, since NSURLConnection is indeed a suitable class to learn about when wanting to make HTTP connections in Cocoa. However, it seems that "checking" it was pretty much the sum of our friend's effort - they "cannot integrate that code", and have thus given up.

This is an issue we all see constantly (and I don't mean having trouble making HTTP connections). There's an entire class of so-called developers whose first and final tactic when given a problem to solve is to simply ask for the completed solution elsewhere, commonly on web forums or other suitable help channels. Their goal is the same as ours - to have code which solves the problem, which can then be presumably delivered to the client. This goal is reasonable and quite normal.

What isn't normal is the unwillingness (I hesitate to say *inability* - because after all, very few things are truly, fundamentally “hard” if you apply sufficient thought and effort) to achieve that goal by a process of self-education, honest attempts and the classic iterative process of refinement and improvement until something acceptable is created. This process in turn equips you better to handle the next challenge, and sooner or later you find that:

- there are entire sets of familiar problems to which you already know the answer and can approach with confidence; and:
- you’re quite capable of approaching unfamiliar problems by generalising your current knowledge and conducting some simple focused research.

This isn’t some trick of software engineering; this is *the entire process of learning how to do anything at all*.

It’s not a secret handed out at institutions of higher education, it’s just how things work: you begin with a lack of understanding about a topic, and a need to solve a problem in that topic area. The honest, sustainable means of doing so is to *improve your understanding*. This is achieved by:

1. Formulating a *question* which, when correctly answered, will improve your understanding in some way; then:
2. *Attempting to answer it.*

Note the second step above. To argue that grabbing the completed solution in some way satisfies this process is laziness and intellectual dishonesty, and probably renders you unworthy of being helped. For after all, why should someone else do your work for you?

I’ve had a lot of personal experience with people displaying this troubling unwillingness to learn or research or try. I’ve released a lot of code over the years, and I’m very visible in the open source community for the platforms I work with. Since open source contributors seem to be seen as freelance teachers, that means I get a *lot* of email asking for help with one thing or another. And I provide that help whenever I can.

I've helped literally hundreds of people looking to get started with Cocoa, since Mac OS X was first released. I don't send boilerplate replies, either - I replied to each email individually. Everything from specific code issues (including but by no means limited to queries relating to my own code), book recommendations, right up to advice on how to get started with programming as a whole; I've done my duty in that regard, and I really do believe it *is* a duty. People who can do something to whatever extent ought to help others who wish to be able to do the same; surely that's a fundamental truth and indeed a desire for all of us.

But this is real life, and no principle comes without the consideration of certain realities. Help may be free for the most part, but that doesn't mean there isn't a cost-benefit ratio to be considered. My benefit may come from a warm fuzzy feeling rather than cold hard cash, but if you're wasting my time then you're not going to seem as worthy as someone who genuinely wants to*learn*.

Here's a secret: ***willingness and desire to learn are the true qualifications***.

Not *ability*; we all have differing innate and developed levels of ability to acquire certain skills. Some (probably most) of these can be improved with practice, and some can't - and it's wrong to pigeonhole or generalise a person's ability in an entire discipline just because of their seeming difficulty in one particular aspect of that discipline. But if you want someone to spend time and effort (especially if it's time they're giving freely), then you'd better earn it.

Earning it isn't about throwing a few units of currency at your teacher, and it's not even about successfully completing your task - it's about bloody*trying*. And trying is what so many of the type of developers I’m talking about seem bizarrely unwilling to do. So, of course, many of us ignore them. Problem solved, right? *Wrong*.

There’s a huge knock-on negative effect of the proliferation of this unwillingness to make the effort to solve problems yourself. People who are in a position to help stop frequenting the chatrooms, forums and mailing lists. “Bad signal to noise ratio”, they say, with some justification. The losers are the genuine (by which I mean well-meaning, willing-to-learn people who just happen to be new to a particular area) developers who naturally choose those places to ask their legitimate questions. These people have a reduced chance to get meaningful guidance because of the effort involved in working out who’s a lazy time-waster and who isn’t.

This is an awful thing, and it’s never been more relevant - since logically, platforms which are young and/or experiencing a surge in popularity are exposed to this phenomenon the most. There are fewer people who are genuinely experienced, the average level of experience is less, the amount of help being requested is higher, and there’s a far higher proportion of lazy, grab-the-money-and-run types mixed up in it all. This is the iPhone, Android and so on.

So, if you’re going to ask a technical question, I guess the first thing I have to say to you is: Great! You’re asking a question, and that means we have a better than even chance that you want to learn something. That’s usually awesome, and I stand ready to salute you.

But wait. Have you considered - really considered - what it is you’re about to ask? Is this the right time to ask, or can you take one more step first, which might either make your question clearer (good) or even unnecessary (probably better)?

Try taking a few minutes to run through these points:

- Have you broken the question or problem down sufficiently to really ask something concrete? In software engineering, you can pretty much divide problems into the two categories of (1) things that can be broken down further, and (2) things you already know how to do or can look up trivially.
- Is your problem the sort of standard question for which there’s *definitely*already some sample code and documentation available? There isn’t a GUI toolkit in the world that doesn’t have a section in the tutorial on how to put a window on screen. There’s no programming language that doesn’t tell you how to read the contents of a file. Skim the documentation, or do a quick search. If your problem is that simple, the answer is probably just moments away. You can find it!
- Try searching the web. This is glib advice, I know, but stay with me. If you’re having trouble getting a decent result, you need to narrow things down. Don’t search for “if statement” if you’re just interested in an if-statement in *ruby*; instead, try “ruby if statement”. What might be even better is finding a site that’s specific to the language or technology you’re working with, and searching *there*. For Cocoa, this is the CocoaBuilder list archives. Someone else has probably asked your question - or maybe a*hundred* someones.