

# Kisses, Nakedness and Love Or What makes a good clinical trials programmer ?

Josephine Duvoid, Independent Consultant

Up until a few years ago I found myself wondering a lot of the time whether I was a good or a bad programmer. Since I hadn't been fired, and had a couple of complimentary remarks on certain pieces of code that I had done, I gradually came to the conclusion that I must be at least OK.

But that's one of the big problems in our jobs as clinical trials programmers. In the beginning, you go on a SAS course then someone hands you a DRAM or SAP, a CRF and off you go. There is no right of passage, no curriculum to follow, no apprenticeship scheme, no teachers. For years you finish report after report, someone says the results pass and onwards you continue. But you are always left with this gaping hole, this unanswered question, this *raison d'être*...yes well, I wrote the programs, it gave the answers, .....But am I a good programmer, could I do it differently, am I missing anything?

The lack of guidance in the pharma industry on how to program with clinical trials data I have often thought is scary. Absolutely anyone can go on a SAS course and write a data step, but trials programming, it seems to me, requires more skills than just SAS syntax knowledge, and it seems to be up to luck or chance to learn them. If you are lucky, brave and have the aptitude, background, and come across the right people at the right time, or happen to qc the right stuff at the right time, then you'll turn out OK, if you are unlucky, over cautious and don't share with anyone, you'll be using the same code year after year and possibly stagnating, and may give up. So I don't think a good programmer is necessarily created in direct proportion to the length of time that they have been programming. I don't think you can generalise and say well he's been programming for 5 years and therefore must be good. It depends what they have been asked to do over the last 5 years, their own approach to learning and who they may have taken ideas from.

A while ago I was working at a company that began a mentor scheme and one of the junior programmers asked me if I would mentor her as a programmer. My first feeling was absolute fear, I thought.... 'Oh my god, she actually thinks I know what I am doing!' My second feeling was jealousy...So many many times over my past career I would have loved to have a mentor of my own. I felt I had been working for years in a vacuum. I wanted someone to tell me if I was doing something right or introduce me to new methods, someone to encourage me and tell me if I was improving, whether I was doing this whole programming thing right... But that never happened, like most of you who have been in the job a while now, I have worked out most stuff on my own by trial and error and the job has been pretty lonely.

Over the years I have also come across various cultural and behavioural barriers to learning and asking about being a programmer generally in IT, you may recognise some of them. There's the 'don't ask because you might look stupid' culture. This is really because we assume that we should know everything and we judge various areas as easy. Then there is the 'don't ask me because I don't want to give you my valuable experience' culture. This was pretty rampant in my first IT job where I asked a question of the 'guru' systems programmer in the mainframe department and I was told quite frankly to RTFM . I was fresh out of university, very pure and innocent....I didn't understand what he meant. When I found out it meant Read the F\*\*\*\*\* Manual. The F of course is the English word for making babies. I was shocked and that put me off asking anyone anything for a long while. And I thought oh well everything must be in the manual. But now I've been programming a few years....I know that everything is not in the manual.

So moving back to being a mentor I thought long and hard about what I was going to say and do with Yvette my mentoree. How was I going to help her become the Good programmer that she wanted to be? And what exactly is a Good programmer anyhow? Now from my experience in IT and programming, over quite a few years now, I have decided that what defines a 'good' from a 'bad' programmer is not necessarily their knowledge of the technicalities of SAS although that is not to say that a good technical grasp of programming is not important, because it is. But what I think is more significant is that good programmers have a certain approach and attitude to their work and their code seems to shine through this. I've condensed my description of this attitude towards Clinical programming into three basic areas. These are

my back to basics rules that I follow personally and refer to especially when I'm in a rut. And going back to Yvette, if she was in a doubting period about her own skills then these are the back to basics I would give to her.

My first principle is Kiss: Keep it simple sweeties!

I think it's true to say that the more you know about something, the simpler it gets. Everything always looks complicated when you start, but after a few repetitions it gets easier and easier. So the good news is that this job will get easier and simpler the more you practice. But what about today. ? Now we aren't programming feedback control mechanisms for rocket engines here! Most of the day to day grunt is fairly straightforward although I admit it may not seem like it when you first read a DRAM. I'm not trying to be arrogant and say, this is easy and if you don't find it easy then you're stupid. What I'm trying to say is if you keep your head, stay open and really look at the problems in front of you most of your programs and tasks can be broken down into simple parts. The best way to keep things simple is to take a deep breath, in fact several breaths and PLAN.

It took me a few years before I worked out that I ALWAYS had enough time to plan my programs. Before then when I was given a programming task first of all I would panic, my stomach would sink somewhere below my knees, I would go and get a cup of coffee, then sit down and immediately type in DATA.....

I was under at least two illusions. The first of these is that good programmers always look like they are doing something..ie typing into a computer. My second illusion was that I wasn't fast enough or good enough to finish my task on time without just getting on with it and typing stuff in, any stuff. What I've learnt since then is..... Every program especially the longer ones benefit from digesting, pattern making and planning.

In order to begin your program, you must take simple step 1. Read the instructions carefully (sounds like a flat pack assembly) and if you don't have instructions demand them in writing. Sounds simple but read the protocol, print off relevant bits of the annotated CRF, read the DRAM and then read the DRAM again. Sometimes I'd like to start a Campaign for Real English in DRAMS (CRED for short), some sections I'm sure I've read time after time and still thought what on earth do they want? Sorry just one small dig at the statos there. Maybe this is why the Clinicians don't read the DRAMS. If I'm not sure about something I consult the writer of the document....hey these people may have degrees in higher maths but most of them couldn't write a best seller, so if something isn't clear, they may just have not had the writing skills to make it clear, so you'll just have to go and ask.

To take the next step in simplification I tend to draw pictures to help me envisage what I'm trying to program. I may do small bits of tester code to test out syntax method and browse data. I do this to see if there are any patterns emerging that will simplify the program. I don't know about you but a lot of my programming is done in pictures inside my head. We all need to visualise the data in some way to picture how we are going to select, mould and manipulate our way to the required result.

That brings me onto thought. Sometimes you may see me daydreaming, well I'm actually working! Some of my best ideas have come when I've just sat doing nothing but thinking about what I need to do, rolling it over and over in my mind. I think this is called intuition. Annoyingly this often happens in my drive home in the car, in non payable time, I can roll a program problem around in my head for ages and then almost like magic, the simplest solution will appear to me. After these activities I have a basic plan no matter how small or large the job....

Now if you think about it we as programmers are translators, the data comes in from CRFs then we translate it into reports. The Translation is done at various levels; it's rather like writing a book. There are chapters, paragraphs and sentences. Well structured programs are like this, broken down into chapters, chapters broken down into paragraphs, and then the single sentences. Your plan would reflect this. In fact only a couple of months ago I saw some code written by an extremely experienced programmer where he had turned off the source from his logs and instead had put statements to the log saying Chapter 1 reading in raw datasets, Chapter 2 calculating age or whatever. It was simple and made everything so easy to follow.

And then this is about as simple as it gets.... It may surprise you but you can only do one thing at a time. You can only write one line of code at a time..... So don't panic at the enormity of the task you may have been given, concentrate on implementing your plan one chapter at a time, one paragraph at a time and then one line at a time and you will get there.

My best programs and the ones I actually enjoyed are those that I took time out to think about and plan.

I have always finished in time when I took time out to plan. I always enjoyed programming more when I planned.

Most times if you don't take deep breaths and plan you end up with spaghetti and what I call 'complicatedness'. Now a programming task may be complex, but it needn't be complicated and there is a difference. Complex means that there are many parts to the problem. Complicated means you cannot distinguish one from the other. Now when I first started programming and I came head on with a complicated, convoluted program I thought 'wow' this person really knows what they're doing! Well, if it looks this complicated it must be good. But years later after many unravellings and changes to other peoples spaghetti, you finally come to the conclusion that you could have done a re-write in the time it took you to unravel, I have learnt that complicated ness costs time and money to unravel it's also deeply unsatisfying work. Mostly I think the programmer got confused about what they were doing, they didn't have a plan. Now when I see spaghetti, I go off for a cup of coffee and think about whether I can justify a re-write.

So my first rule for good programming is to try and keep it simple, read the instructions, draw pictures, break the job down into small steps, think, test out bits of code on their own, have a plan, then assemble the parts individually and I believe simplicity will scream out to you every time and make the whole job easier. The end result will shine through in your work, no matter which particular parts of the SAS language you cared to use.

Now onto the second aspect of Good programmers....Nakedness

People don't usually like the idea of nakedness. They believe that if they run around naked someone will laugh at the size of their bottom! I once, by mistake ended up in the Roman Irish baths in Baden Baden Germany. It's a system of hot and cold baths, dry rooms wet rooms, whole body scrubs by huge German ladies and general pampering. It's all fuelled from the natural hot spring water in the area. My friend and I entered the Roman Irish baths with our swimming costumes in hand, only when there was no turning back we realised that actually everyone was naked and we would be the only ones wearing swimming costumes. After our initial shyness it was actually a very liberating experience to spend an hour publicly naked. When everyone takes their clothes off they are removing symbols (i.e. clothes) about who they state they are. They have nothing to hide. You find that some people have smaller bottoms than you and some people have bigger bottoms than you. But when everyone takes their clothes off you know somehow you relax and realise the shape of your bottom doesn't really matter!

Where is she going with this? What does it have to do with programming? You may well ask; Nakedness is about openness, when you are open you realise that everyone has differences but really everyone is the same. When you are open you learn that people have evolved differently from you, they have something to teach you and maybe you have something to teach them. There will always be programmers better than you and less able than you on the evolutionary ladder of programming. None of you in this room will ever know everything there is to do with Clinical programming and SAS. AND NEITHER WILL ANYONE ELSE! Don't you think that it's a liberating thought to think that actually you don't know it all! Doesn't that feel great, fearless and free? What if we all took our clothes off and talked about what we were doing in a way that no one felt intimidated? What if we shared our programming ideas more openly? What if we looked at other people's work with a constructive eye instead of a critical eye? What if we realised that as a group we may actually know it all because all of our bottoms are different? Well I think we'd learn faster, we'd enjoy the job more because we'd be talking to colleagues more instead of screens, we'd finish jobs quicker, and we'd ALL arrive at the same standards of programming. In essence by being more open, by default, we would create the curriculum, give each other the lessons, and no one would be left out.

Let's talk about evolution. A lot of people are scared to talk about their work for fear that they will look stupid or someone will finally find out their worst secret that they don't actually know what they are doing. You are all in a state of evolution and change both as a person and as a programmer. The way you write your code changes year on year as you learn new ways of approaching problems and adopt new methods. You are better than you, two years ago (I hope). And everyone has gone and is going through this process. You need to accept your state and the learning point at which you have arrived. Accept you are doing the best that you can given the experience, knowledge and understanding that you have accumulated so far. In fact your program is perfect for your current evolutionary state. It would be so boring if we were not learning and knew it all. So stay open. Have the confidence to go naked and learn from others. And don't laugh at anyone else's bottom! Try not to judge someone else's way of doing things, it's different, it's acceptable for the state of experience and learning that they are in. Remember, they don't have a curriculum either. If they are open to suggestion maybe you can help them onto a new level.

A good programmer stays open and humble to always increasing their skill levels no matter what point they have arrived.

So throw of the clothes, we're all in the same pool, Stay open to learning, open to growth and create a safe space for other's less experienced than yourself to ask questions without feeling stupid.

And finally my John Lennon part of the talk...All you need to be a good programmer is...Love

Do I love my job and am I going to tell you to? Well no! but I do love my patients. Let's pause for a moment and think about what we are actually doing. What's behind the egos and the tech talk?

Patients. ....People who are sick, afraid and in pain are taking our pills and injections, going to the clinics, filling in the questionnaires in the hope that there will be a cure or relief either for themselves or for future generations. Most of these people don't even know if they are on placebo, but they still donate parts of their lives and put up with lots of questions that they could probably do without. As programmers we are cogs and quite a key cog in the wheel, proving or disproving the efficacy of new medicines and very importantly checking their safety. (Because I know the safety area sometimes isn't regarded as sexy as 'efficacy' but it's equally important in the end),

These drugs aren't just labels they are pills, injections, liquids, those numbers aren't subjects they're people, your husband, wife, brother, mother or friend. It could be you or me one day who takes this medicine.

You know what? The very best clinical trials programmers, realise this, they take seriously the fact that they are custodians of the data that comes off the CRF. They realise that to most statisticians and clinicians the programming function is a black box and they have to trust that what comes out of the black programming box is right. It's about paying attention to detail as if you were taking the drug yourself. I lose track of the amount of times I have been asked about attention to detail in interviews. This is SO important to employers. No one wants mistakes. So how do you avoid mistakes ? (this is really the stuff for another talk but very simply) Look for detail, missing data, wrong data, unexpected data, and program defensively if you can. That means program expecting wrong things to occur. But the best way to avoid mistakes is to always always have your work qc'd. The best programmers know that they are not perfect. They know, that they can't program a summary on their own, they don't know everything and that they make mistakes. That is why they welcome QC, checks, constructive comment as part of their professionalism. If you are in hospital and the drug trolley comes round, two nurses have to check your dose, similarly I think at least two qualified programmers have to check a summary.

I once QC'd a medical history listing for cancer patients. It made pretty grim reading, years and years of drugs, surgery, chemotherapy, radiotherapy and most of the patients died. And you know what the purpose of the study was? The fancy word is to test the toxicology of the drug. The bottom line was to test the maximum dose before the patient was poisoned probably killed. To me the data from this and all clinical trials is like drops of golden nectar borne of tremendous suffering. We have a responsibility to the patients and ourselves to make sure we get the translation from the CRF to the report right. So the next time you are confronted with a QC and you yawn and think this is oh so boring....think again take responsibility, equal responsibility with the main programmer to check the efficacy and safety of this drug for your friend.

Still on the theme of Love.....Love yourself.

I'd also like to say something about loving yourself and supporting your colleagues because this programming lark can be pretty stressful. Apparently one of the most stressful jobs in the world is a bus driver (this probably means more to those who are English with the congested English traffic). Let's say, an English bus driver has a fixed timetable that he cannot change but has to stick to, but he can't control the traffic flow and never really knows if he will get to each stop on time. There maybe unknown traffic jams and road works ahead ruining his schedule. It's a bit like that with programming. I hate estimating how long things are going to take me. It's not easy an easy thing to do. I can never really tell until I've got into a job and am familiar with the data. Also any number of things could obstruct the development and prolong the development time, usually issues with the data or a change in the DRAM, or items requiring further clarification. But the deadlines remain the same. So it can get pretty stressful out there when you are programming to deadlines that you didn't invent and have little or no control. Now I don't know about you, but I want to enjoy my days, I don't want to be stressed so I think we need to support each other more in various ways. Sometimes, you may need help and I want to tell you that ASKING FOR HELP IS A STRONG THING TO DO. People are just waiting to give you help, they've been wanting to speak to

someone all day! They need to GIVE too, so let them do it. If you've been given a task with a deadline don't assume that the person who created the deadline knew how much work was involved. If it's too much, talk about it; share it for the sake of the quality of the project (and the patient and you).

Regard Your QCer as your buddy, your partner, both checking that you have the same result, the same interpretations of the DRAM. Don't regard him or her as the person waiting to catch you out or show that you don't know what you are doing.

Another stress area is guilt. Don't regard not knowing anything with guilt 'Oh I should have known that' I hate when someone says to me 'You should know that!' If this happens first say to the person "Stop shoulding on me !", then say to yourself: 'Well I could have known that but I didn't. No one ever told me the curriculum, so what? Now I do know it! Great! I've learnt something.... It's time to move on!

And then when it's all too much and the deadlines won't change and there isn't any help....I was once asked in an interview whether I would compromise quality if I wasn't going to hit a deadline. I'm afraid my answer was no. I don't really want to care who is breathing down my neck, I'd have to know that the results I was publishing were right. If the patients bothered to go through the trial, I can be bothered to make sure we get to the correct result, even if it is 'late' by someone else's milestone. (I did get offered this job by the way)

Another area of stress is criticism; there is an old saying that goes:

If one person says you are a horse, you don't have to listen. If two people say you are a horse, you probably need to pay a bit more attention to what you are doing. If three people say you are a horse more than likely you have hay hanging out of your mouth and a saddle on your back.

If you come across criticism of your work assess the intensity. It may be that the person criticising has problems of their own that they are projecting. If it happens again become more aware, but try not to take things personally. That means try not to judge yourself and therefore get angry. To quote one of my favourite writers: 'If you can control the ego long enough to hear what is being said, you may just realise that people usually say to you the very things you have silently said to yourself'. If you are aware of your problem without emotion attached, then you are in a position when you can choose to do something about it. If someone criticises your work it doesn't mean you are a bad person, it maybe means there is something you could change about your work. Could not Should....there is always a choice.

In general if your program produces the desired and accurate result required and is well documented (I'll repeat that....**well documented**. I could do another talk on documentation.....) then it has suited the 'business purpose' that you were paid to perform. The bottom line or fear, I guess is losing your job because you are 'bad'. I've only seen one programmer get the sack and that was because he kept falling asleep at his desk, I suspected that he had some type of narcolepsy but the manager was not prepared to explore sympathies. Generally if they're still employing you then there's more time to learn and improve your programming.

So my summary for being a Good Programmer is this:

Keep it simple... break things down, and then build them up step by step....have a plan.

Go naked, stay open to your learning, and stay open to others' learning.... share without blushing.

And all you need is LOVE....Don't forget what you are actually doing here.....the Patients

We don't have a curriculum or apprentice scheme, we don't have courses on clinical trials programming techniques, so we have to support each other and support each other's learning so that we are all 'good' programmers together.

P.S Programmers who kiss, go naked and give lots of love, are fun to work with too.