



Core Values :

- Knowledge when shared, grows
- Ones' experience is a lesson for others
- Giving back to the society
- Let us learn together
- Building a knowledge based society

Editorial

Greetings,

Couple of months back, Lakshmi and myself had a small talk over the coffee table sipping hot steaming cup of coffee while the rain gods were pounding outside. Each of us began talking about the fresh batch of grads who had joined the organization. Few of them were extremely talented, few were still coming to terms with the corporate world. We just had some fun over the adventures we had when we had landed on our first day @ work.

Looking back, that conversation seems to have tipped the thoughts about doing something about the state of works with the college grads. The extremely talented buch of lads passing out of our education system YoY provide a great opportunity to interact and learn. The opportunities available are worth pursuing with passion.

Think of the case of Raj, a guy with high ambitions and great attitude studying in a reputed college but in a remote corner of Tamil Nadu. The dream of joining a big corporate house could be overwhelming. A step by step guide to plan his transition to the corporate world could do him a world of good.

After having entered the corporate world 4-5 years ago, we surely think there is a foothold for us. And as many foot-board bus travelers can vouch, it is expected of the guy having a foothold on the footboard to give a helping hand to the one trying to board the bus, with a SMILE.

It is a dream come true to go online with Opengyan.

-- signed

Becoming a Better Programmer - Part 1

Welcome to the first and introductory part of series of articles on becoming a better programmer. The idea of series of articles is to bring out good practices of software development. Though software engineering is a very complex area, we are going to focus on the things that are very important for software development of any sort - may it be a lab exercise or an open source project. The readers are suggested to reflect upon the exercises given at the end of each article for better understanding.

I was reading and reflecting on the short article written by Badhri on Programming languages. Though it was short, it gave a general overview about the programming language. Any programming language is a tool to achieve your imagination. For example, if you are writing a simple program (similar to the ones that is given to you in lab), you do not need to think much. But in reality it isn't the case. To be a programmer, knowing a programming language is secondary. Knowing the syntax of the programming language won't make you a good programmer. If knowing programming language is sufficient, then compiler would do the job. Software development is much more than knowing a programming language or few.

Any software is a solution to a given problem. And programming language is just a tool to solve the problem. The solution to the problem is more important than the tool (the programming language). In order to develop a better software one needs to follow certain basic principles in a specific order. The rest of the article discusses the basic principles of software development in a much simplified way.

First one needs to understand what needs to be done. Because you cannot go on writing program simply for everything. You need to understand that you are writing a program for someone to use it. Remember that you will not always write program to improve your knowledge. So understanding the requirements - what needs to be done plays a crucial part. Understanding the requirements is the first step and if you do not understand the requirements, remember you do not have the second step.

As a second step to writing a program, you need to know how to solve the given problem without employing much of your knowledge on programming languages. For example, if you are writing a game, you need to understand and know how the game is played, how many players are involved, what are their roles etc. In this phase, you will least bother about whether you will use linked list or array. But you will solve the problem at an abstract level. This is called high level design. After high level design, you will decompose the high level design into modules and design each modules. For example, if you are writing a program to validate Palindrome, you need to have input module, processing module and output module. This ability to decompose the problems in sub-problems and solving them with sufficient data structures and algorithms is called low level design. While you are solving the problem, you should also keep in mind about the future problems that may crop up. Your design has to be flexible enough to accommodate future problems (which is quite challenging).

The third skill that is required is to convert the design into product and this is where the programming language comes in. Without the above two steps, the third skill is

not worth and in fact not needed. Once you have understood the problem and solved it at a conceptual level, then implementation becomes pretty easier. You do not need to beat around the bush. During the implementation phase, you will implement the entire module by modules and finally integrate it. You also need to ensure that your software will work by testing your software at multiple levels. Apart from implementing the design, the coding is also about producing maintainable code. When you develop software, you are writing the software which is expected to live more than your lifetime without going to hospital for treatments or surgeries. The program (aka code) can be written with a lot of grace and the code that is well written will live beyond ages (example Unix). The developer needs to understand about other developers who are working with him/her and the people who will be working in future. So programming is not merely following syntax but much more than that. As this point of time, we have discussed about three important qualities of a developer.

The final thing that you need to do as developer is to test your software. The primary focus of testing is to uncover hidden defects. As a developer, you are the best person to know about your code and it makes sense for others to bank on your certificate. So, it is the developers' responsibility to test the software and fix the defects.

This is pretty much about software development lifecycle and you need to evaluate how much percentage does a programming language play here. May be 40%. But please do not get me wrong. For writing a software, you need a language however for solving problems you need skills like understanding requirements and designing. Without those skills, you will not be able to excel as developer. Now would you understand why you have "Algorithm", "Flow Chart", "Hand Calculation" and "Output" section in your lab observation.

When you follow the above software development practices right from the beginning, you become a natural programmer/developer to solve any complex problems. You may want to evaluate your current way of writing programs. First solve the problem and then start writing the program.

Exercise:

Take any simple program (Palindrome, Fibonacci series) and apply all the concepts you understood from this article (Duration: 3 Hours, Complexity: Medium)

1. Clearly understand the requirement/problem. For example, what is palindrome or Fibonacci series
2. Solve the problem by decomposing into modules
3. Write the software
4. Finally test.

Transitioning from College to career

Life is just a bit different outside of college campus. Many of us never want to leave college -- and whom to blame ? :) We all love to lead a carefree campus life. If a day scholar then it is fun to have the college bus or the train ride. The whistles, catcalls, the last minute cramming for exams, friends made in the train / bus and its a world onto itself. The hostel life is even more interesting. Late night adventures, gossips, pranks and late morning sleepy hangovers. Well its a world apart when stepping into the corporate world. Apart from the fun, there needs to be a sense of commitment and self control. It is not so gloomy, on the positive side you get to earn and it is almost good riddance to exams, tests and assignments. Lets just go over few pointers to adapt oneself for the transition.

Time :

If you have been into the habit of bunking classes, visiting the class rooms rather than attending them it is going to be a tough task. If you have the habit of hitting the snooze button when the professor is taking the classes, wake up... The thoughts of vacation -- let it just vacate your minds. It has rented your minds long enough. As a new joinee you will not have much choice or flexibility in the holidays. At the end, it is a tough ask to change from an era where in you were studying for 25% of a day and leading your life for the rest of the time. Now after joining the corporate, it is going to be a 50 % office life at the very least. You dont schedule your dinner 4 weeks from now with some strangers during college days. Nor was it ever a 50 hour / week study time. But if you can schedule your tasks, then office life will be a cake walk. It is just after all 9 hours a day if planned well.

Professionalism :

College life is a passage to try out new things, lot of crazy ideas, and being irresponsible. Acting a bit unprofessional will just result in getting a poor grade. But in career, dependability and personal accountability are major factors. As a graduate you will not have all the answers, seek out the information. To succeed, you must be seen as a member of the team that can be relied on to do your job. Deadlines are critical, much more so than in college. You cannot plead your manager for deadline extension as done with professors.

Job / Career :

Many of us want to land in a job first. Career aspirations are secondary in reality. Dont fret if your first job is not a dream job. Many of the IITians join an ordinary college, take out an year and crack the JEE again. Learn where you are and if it is a mismatch you can always change. A bid in the hand is always worth two in the bush. Also, it is not necessary that you always have to land in a job related to your major. An Electical engineer becomes a programmer, a computer engineer takes up HR profession. Start planning so you can make the transition to something better down the road. Keep track of your accomplishments and develop an awareness of the transferable skills you develop in current job to close the gap between current skill level and those needed for your career.

College does not teach EVERYTHING

You might be a university gold medallist or a college guy with extra ordinary extra curricular activities, still college life does not prepare for everything. You need to start understanding about :

- Team work skills
- Cross cultural skills
- Personal finance management
- Balancing work - life
- Job hunting :)

Dont be full of yourself.

You can be proud of yourself, admire your skills, but just dont take it to your head. Being an university gold medallist or having studied in a "famed" college can take you just to certain level. It cannot entitle you to be placed so high for the rest of your career. You need to prove yourself and learn new skills always. Instead of making your employer notice your "work" just try to do the job and innovate to make yourself more an integral part of the team. At the end of your college you just have skills just to fill a page. just bear that in mind, dont oversell and lose credibility. Just because you scored high ranks does not mean you join as a leader of the team. Everyone has to work their way up the ladder one rung at a time.

Take time and plan your career. It is atleast going to be a two decade long journey based on the efforts put over the past two decades. Dont start it in haste !! Plan the transition.

Scoring and soaring high

Once you are on the runway of the career flight path, you just have to take off. There is no other realistic option. Then starts the journey. You have to plan soaring high in the initial years or risk the crash land.

It was my first year at office and I was really disappointed at having to land in a company. Its all because I did not want to land up in a company, the first place. I wanted to pursue higher studies. Though I liked the pay, I liked college better:). Thankfully I got into a division where in there is real product work rather than services. So thats point number 1. [If you dont like something and forced to take it up, find the good sport in it.](#)

Secondly the point that I considered myself to have landed up in a company rather than studies, made me undergo sleepless nights. It is that I wanted to prove above all to myself that I am better than the average or above average in this place. I always felt, to earn the first salary equal to my Dads' I have to put in all the efforts he had put in all these years. And if I am not @ top 5 % then I am doing a disservice. I said : [I should never ever feel guilty of getting the payslip.](#)

To move up one needs to show that he is ready to take the next higher task. There has to be a balance between finishing off the current work at hand and not to over do oneself and burning ones' hand. I had many times burned my hands for overdoing. Sometimes subtle sometimes not so subtle. One just gets to learn. The point is [Finish your task, Step up to next task](#)

The other thing I learnt is that consciously or unconsciously your actions should not make the others look bad / stupid. You might be a genius, but that doesn't mean you have to pose like one by making the guy in next cubicle other look dumb. It has to be a principle and consciously one has to do it. The point is : [never ever make your Manager / peer look like a fool.](#)

There are two contradicting styles of handling things. Never to over do or to under do them. I had always tried to follow few things.

- 1) Say sorry in the open, It adds to the earnestness.
- 2) Wish someone personally and not as a group, It adds to sincerity in the wish.
- 3) Praise others genuinely not a fulsome praise.
- 4) If your team mate, dont hit him / her when [s]he is down.
- 5) If an adversary, dont hit him / her when [s]he is out.
- 6) Avoid blaming a person unnecessarily.
- 7) Be truthful to yourself.

The first time I came in to a project, I thought, I need to prove a point to myself. The idea of having a role model for oneself never appealed to me. I am unique and so is

everyone. No one can equal ME and never can I equal Mr.X. But that doesn't mean I have to boast myself to the hilt. There are and will always be a better person than me around. Humility is a characteristic one has to constantly remind oneself. [Thalaganam koodadhu](#). It is but natural to feel proud. Pride is needed but humbleness should always be there. In front of peers and elders, be humble. But that doesn't mean you have to be in a shell always. When questioned, just blow the opposition to smithereens. I just remember two historic instances.

One more thing to consider @ office is to follow NAM :). [Get away from office politics altogether](#). Bootlicking, back stabbing do help to get phenomenal short term gains. But just think what happens if you change team or your manager jumps the company. One will be in deep trouble then. Negotiation, flexibility and swift thinking are needed. But that does not require one to be a bully boy nor a "yes boss". Just strike a balance most of the times and it should be fine...

Well the other point is come out of college life. Hi, "office / business talk" , bye, let these be the guiding states while talking to peers of opposite sex. It is not the college life where you just become friends upon first sight. Not anymore. It starts unwanted gossips which results in unwanted diversions. Once you build the rapport, friendship builds. Just don't jump right away !! [Friendship is a long journey to be begun hastily...](#)