Mystical Art of Computer Programming

Dr. Debasish Jana
Dynamic Digital Technology, Kolkata
Visiting Faculty, Jadavpur University and BIT Mesra Kolkata Campus
Fellow, IE(I) and IETE, Senior Life Member, CSI, Senior Member, IEEE and ACM
djana@alumni.uwaterloo.ca

“If people do not believe that mathematics is simple, it is only because they do not realize how complicated life is.” ~ John Louis von Neumann

Abstract
In sixty plus years of computing, with technology advancement and growing business needs, the support of legacy systems as well as new customized applications are in the rise. While rapid application development is today’s need, at the same time, we cannot ignore the increased complexity and depth of knowledge that is required to cope up with sophisticated tools and technology. Programming was hand crafted before and even today, but modern tools help to generate many codebase without knowing fully what goes inside and what the risks in adopting them are. In today’s perspective we explore the mystical computer programming which was considered to be an art..

1. Introduction
In sixty plus years of computing as industry progressed with rapid advancements in technology, more and more sophisticated applications of computing began to be realized. Today there is a vast potential as well as many computing challenges which need to be jointly addressed by academia and industry [3]. As technology sophistication is increasing day by day, more and more challenges are being faced by the researchers and practitioners to cope up with the core foundation knowledge in computing as well as the application builders who are developing and delivering applications to meet today’s user needs. On one hand a handful group of people are concentrating on core research, innovation and also leveraging technology in developing tools for rapid application developments, on the other hand, there are a plethora of programmers, analysts, business domain experts who are shaping and customizing the tools to meet the client needs. While we talk about availability, scalability, reliability of applications with increased security, the core foundation knowledge on technology front sometimes remains in dark. While rapid application development is today’s need, at the same time, we cannot ignore the increased complexity and depth of knowledge that is required to cope up with sophisticated tools and technology. In today’s ever demanding computing world, this article explores the dearth of depth in core computing, programming in particular and its mystical art and applicability.

2. Myths and mystics of Computer Programming
With more than sixty years of computing era, many programming languages have evolved in so many years. The idea of stored program was first introduced in late forties by a famous computer scientist and mathematician John von Neumann. Since then, many different programming languages have got born, became popular and some got extinct too. To express the technique of solving a particular problem through computer, we need to express the instructions to be performed by the machine through some programming language. Programming language acts as the media of communication to the machine. When students learn programming through some programming language, many a times, they focus on the
intricacies of the syntax and the detailed characteristics of a particular programming
language. But such minutiae of programming constructs may omit the significant
characteristic of the underlying paradigm if not understood from a different context often
leaves students to figure out the essential character of programming through an ad hoc
process of trial and error. There are so many, literally hundreds of different languages we
know of and some of them are quite similar. The actual syntax may vary, but the essential
concept may be quite similar. Paradigms thus stand in the crux of the learning programming
languages.

Donald Ervin Knuth, the father of analysis of algorithms and the stalwart pione
Computer Science authored the legendary seminal multi-volume book titled ‘The Art of
Computer Programming’. In early days of programming, when fundamental technology was
dominant from first principles and not many tools were available, programming was an act of
dexterous crafting. Computer Science used to be a challenging subject by discipline and
computer programming involved beautiful and interesting patterns, blended with fundamental
data structures and robust algorithms and then neatly stitched together to solve a particular
domain problem. In a way, computer scientists used to be thinkers and philosophers to put
things together in an adroit way.

With advent of sophisticated technology, day by day technology has attained a colossal
progression. However, the need of more and more production, and urge of saving time in new
developments to pace with and adapt with the technological changes has made rise of many
tools which make the life of programmers easy. With state-of-the-art tools, technology has
taken a back seat and remained confined to the experts who nurture technology from behind
the curtain and make these sophisticated handy tools that help in rapid application
development. A new breed of programmers uses the new tools and with portions of generated
codes, frameworks, libraries they are able to develop applications rapidly and relative ease. In
the process, the craftsmanship skill has taken the shape of engineering and the art is
somewhat lost in the whole process of production.

A laborer works with their hands and does not put much of mind into the work. Thus, the
work becomes more mechanical than a thought process. With tool driven approach, many of
today’s programming task has become mechanical and mundane. However, a craftsman
works both with their hands and their mind. An artist blends his heart as well with the work
with their hands and mind. Computer Science involves obviously not just programming but
many different areas and subjects blended together to be orchestrated in a uniform act which
is enjoyable, challenging and thought provoking. The way dancers enjoy choreography,
improving programs for better efficiency without losing the proficiency is still an art. Knuth
once said “We should continually be striving to transform every art into a science: in the
process, we advance the art”.

3. Programming as a passion
The general trend among today’s students is to invest resources and energy into learning
skills that will help them to benefit in the immediate job market. In four years of formal
baccalaureate degree of Computer Science or Information Technology, many students get
recruited in advance in their third year of study. This has a very negative effect on their
ongoing study and knowledge. Many of them feel that they have got the ticket of a field with
greener grass, so why waste time to studying and gathering knowledge? Consequently the
emphasis moves away from the core Computer Science discipline. In short the fundamental
knowledge of the student remains rather tenuous founded as it is on not very deep knowledge
of the first principles of Computer Science [3].

Computer Science or Information Technology study should not be treated as a ticket to
lucrative jobs but also dearth should be felt where new ideas are waiting to be explored in
dynamic and evolving field of computing practice. Industry practitioners and academia members can help to make this happen. The faculty members need to work proactively in designing courses and curriculum to help foster the spirit of enquiry and inquisitiveness among students, rather than concentrate on subjects which though are lucrative in the job market, do not provide as solid an insight into the discipline itself. Higher the quality of research, greater is the long term benefit to industry, because this very research will trickle back into industries to enable them to produce better products and make them more competitive. Also what is required is to encourage research across a broad spectrum of topics and encourage students to explore new ideas fearlessly [3]. Industry also needs to pay attention in Research and Development activities for long term sustenance. Because of increased off shoring opportunities and demand of consultants in the western market and the huge currency differences, the information technology industry has seen a phenomenal growth with sky-high salary to professionals in last decade. But offering professional services need to strengthen one with its core fundamentals as well. On one had we see more users of the sophisticated tools today that reduce the development time in some cases, whereas there is also increasing dearth of professionals who have confidence as well as grip on the fundamentals to make a tool like that. More product initiatives should be emerging with industries concentrating on long-term growth of knowledge as well. Indian knowledge industry cannot survive for long with service model alone. Research and development initiatives towards making successful products as well as market those worldwide will be bigger challenge of IT industries in India to sustain for long time.

Some people who love to program are passionate about the work they produce. Not just brain, not just the fingers typing the keyboard, but with a gentle heart, a great philosophy, a profound thought process makes programming a real piece of art. Don’t we agree that UNIX and its variant of Linux is an art by itself? The creativity and imagination of the programmer can very well lie in the layout of the source code, documentation, internal comments, error messages, exception design, and many others, very much like a novelist planning the novel or a poet penning down the poem. Knuth expressed his thoughts about the art in computer programming as that some software development could be classified as requiring the skills and perspective of an artisan -- building Graphical User Interfaces (that involve the user and integrate them into the user experience), or perhaps image processing, pattern recognition, multimedia and computer graphics, or maybe even an elegant solution to a routine problem.

4. Computer Science = Programming?

It’s a common misapprehension that learning Computer Science is all about programming. The brilliance of the core intricacies like soft computing, discrete mathematics, algorithms and data structures, operating system principles, database concepts, networking and above all programming pearls and its various gamut of ever emerging applications are all pervasive to make Computer Science as a separate discipline of study in its own right [3]. James Morris, Professor of Computer Science and Dean of Carnegie Mellon University's West Coast campus rightly pointed out that programming doesn't begin to define Computer Science” [6]. Computer Science should not be portrayed as the money minting machine, rather knowledge of Computer Science should be put forward to build tools and technologies of tomorrow. Computer science education is not meant to supply programmers to the computer industry. A computer scientist should preserve the knowledge and update regularly to cope up with the challenging nature of changing technology to bring out new innovations in research as well as development. Just to cite evidence, Tony DiGioia, who emerged as Pittsburgh's most famous hip doctor, began in computing at Carnegie Mellon University [3][6]. Knuth did not believe that it would be possible to take any person off the street and teach them to do deep research, or make them into an expert programmer. While coaching senior
graduate students, Knuth used to start with small problems. First a student would solve a problem that was solved centuries ago, but the student should solve it independently. Then the solution to a problem would come that was first solved only a decade ago. Eventually, with the same research techniques, along a result would come that would be totally new. With this philosophy in mind, the problem solving skill gets developed in an artistic way. Knuth had always thought that the psychological profiling would be mostly the ability to shift levels of abstraction, from low level to high level while architecting solution. He believed that to see something in the small and to see something in the large is a kill by itself that needs to be nurtured and developed. To explain science and act of art, Knuth said in 1995, “Science is what we understand well enough to explain to a computer. Art is everything else we do. During the past several years an important part of mathematics has been transformed from an Art to a Science.”

5. Growth Management
Today, practitioners are talking about availability, scalability, reliability, security of applications. Many of the applications are legacy applications running over mainframes or some legacy platforms over years. With advent of new technology, these applications need to be maintained with same ease and control. However, as technology advances there are increasing issues of interoperability, compatibility and adaptability among varieties of platforms, technologies and tools. To sustain the need of ever increasing business growth and dynamics, rapid application development and rapid architecting a solution have become a necessity. However, to cope up this agility in demanding business scenario, the knowledge workers are not spending enough time in understanding the dynamics of underlying architecture, technology and above all programming these systems. As a result, computing architects are suggesting solutions somehow to fit in one tool with another with an obvious load on performance, memory management, resource utilization and of course cost. Cost reduction in proposed new systems sometimes remains a saga but there are many hidden costs and factors that underlie. For example, sometimes to cope up client requirements, a group of architects sometimes even suggest to run a Java Virtual Machine (JVM) on a Microsoft Dot Net framework to meet both side of the worlds – to run on dot net framework as well as support legacy or varieties of Java applications that run on JVM environment. Some people look for running legacy windows applications on Linux by having a virtual machine kind of environment or having some tool that somehow converts window based dynamic link library to run under a complex emulated environment on a different platform. In the name of keeping business viability, what stable architecture we are proposing in terms of these varieties of tools and stitching them through somehow, time will tell.

6. Algorithmic Challenges in Programming
Imparting computing knowledge should not comprise of tool-based training or knowledge of a few new languages but rather in-depth knowledge of current technologies and methodologies. Such activities will make our students globally competitive and more adaptable to changes. Just to cite an algorithmic challenge, recursion is thought to be the definitive art of the algorithmic programming. Just to cite an example of algorithmic tickle, recursion is nothing less than a magical work of art of a poet in writing the poem. Recursion has its own limitation, being bound by the stack size limit. More depth in recursion has an impact on size of stack and may result in stack overflow. Thus, for large sized problems, an iterative approach using a loop is highly scalable and maintainable because probably this is often the only alternative. However, for some complex small logic, recursion can be written as a poetic piece of work as elegant piece of art and the program may even run faster if implemented recursive way. Stack is an example of data structure with recursion in its heart.
Many recursive programs may be iteratively converted in a non-recursive way by using loops or using stacks.

Although, now-a-days, spending time on algorithmic computing is no longer considered to be useful by a group of industry practitioners who feel that knowledge on business verticals like banking, insurance, health sector will drive the business only and programmes can be hired to put things together in the business process architecting through application integration. While application integration is surely a business need, but going in-depth to find out a suitable solution, and if required to tailor it from scratch is becoming very rare.

The current murmur is to centre on software engineering aspects and the elite jargons, SOA, cloud computing and what not. However, to have something new or having solid foundation, the algorithms cannot be ignored, and after all algorithms have been proven to be the best motivation of the mystic behind programming. The sense of achievement that is more realized when completing a complex algorithms implemented in a business scenario. It’s so evident from many years back and even today when I struggle through intricacies of programming, more in algorithms. To cite some other example, programming with graphics or window based system are very different from traditional console based programs. Again web based programs are very different from native applications running on a smart phone, for example.

Programming Windows is a bit different than the way we are familiar with the conventional character oriented application development (for example in MS-DOS or UNIX). As the programming guru Charles Petzold puts it “Windows are easy to learn but difficult to program”. Not because Windows is a multitasked environment, not because it resolves the screen in terms of pixel resolution rather than characters, but because it defines every action to and due to a window as messages. Therefore, programming Windows is programming messages with which we are not usually familiar. Messages are generated due to actions or activities related to a window, for example, when the window is resized, or repositioned, or mouse has been moved over the window area, so on and so forth. A Windows application expects to service messages and also generates messages to be serviced by the Windows Operating System (OS) or any other Windows application. Is the Windows programming difficult? Not really – like any other programming, once we appreciate the core issues of message driven architecture, once we understand that from where these messages are generated and how they are serviced in a program, we are the experts in Windows programming. Let us then explore the message driven architecture of Windows programming [7].

7. Conclusion
Although agility and structured object-oriented approach has given us cheaper and more reliable software products, do we still feel that programming is still an art? Is programming looked down upon in Indian context whereas we see plethora of expert gray-haired programmers in western world who love programming by passion and produce quality software. Are we in India holding that passion to create hand crafted programs? Is it still an art? We strive for the answer.

Reference: