Review Article

RFP Indian Journal of Medical Psychiatry
Volume 7 Number 2, July – December 2024

Impact of the Psychology of Programming

T.V. Gopal

How to cite this article:

TV Gopal, Impact of the Psychology of Programming. RFP Ind Jr of Med Psy. 2024;7(2):73–77.

Abstract

Human beings are good in ideas and abstractions. Developing Software is a process that expands the documentation with details that are close to the machine. Donald Knuth proposed "Literate Programming" in 1984 which changes the traditional approaches to the design and development of programs. Instead of having the main task as instructing a computer what needs to be done, the focus should be on explaining to human beings what the stakeholders want a computer to do. Instead of writing code containing documentation, the literate programmer writes documentation containing code. The "program" then becomes primarily a document written for humans. Writing a literate program is more intricate than writing a program in a chosen programming language.

With the computers becoming ubiquitous, the inherent essence of computing is so contagious that every human tends to code and even revel in a sequence of codes that work like a program that runs on a computer.

The interplay **"Human X Computer"** with computing as the basis does impact the mental ability as well as the agility of the human coder or programmer. This paper is an overview of certain such impacts.

Keywords: Psychology, Coder, Programmers, Program, Literate Programming, Mental Ability, Mental Agility.

INTRODUCTION

The lack of first principles of software development results in "the software crisis." The North Atlantic Treaty Organization [NATO] Software Engineering Conferences held in 1968[20] and 1969[11] suggested some initial principles given below.

- **Software artifacts are "machines"**
  - ♦ They are primarily characterized as deterministic, in adherence to the principle of cause and effect, describable using appropriate formalism
- **People are potential "machines"**
  - ♦ After initiation with adequate education / training in the understanding of the pertinent formalisms
- With a chosen set of management principles and practices
- **Systems are Cartesian**
- **Model = Machine**

Author's Affiliation: Director, Center for Applied Research in Indic Technologies & Professor, Department of Computer Science and Engineering, College of Engineering, Anna University, Chennai 600025, India.

Corresponding Author: T.V. Gopal, Director, Center for Applied Research in Indic Technologies & Professor, Department of Computer Science and Engineering, College of Engineering, Anna University, Chennai 600025, India.

E-mail: gopal@annauniv.edu

Received on: 09-08-2024 **Accepted on:** 05-10-2024

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0.

- Possible to define a formal syntax and grammar capable of Unambiguous description of the implemented machine.

Computers support software models. Software implementations are representations (models) of real-world conceptual systems. The task of development is to represent concepts with technology. Software is developed only once. The cost and effort estimates often have unreliable basis making them go woefully wrong. Software artifacts have many defects resulting in frustrations among all the stakeholders. Software development has programming as an important task. The nature of programming is best expressed as given below.

"I have often felt that programming is an art form, whose real value can only be appreciated by another versed in the same arcane art; there are lovely gems and brilliant coups hidden from human view and admiration, sometimes forever, by the very nature of the process". - Ed Nather (UUCP email address utastro! nather), The Story of Mel, Usenet news group "net.followup" on May 21, 1983.

The process is such that, it is very difficult to read the code scripted by another programmer. Maintaining an existing software is both expensive and cumbersome. The thumb rule is if more than 10% of an existing code needs to be modified, it is better to scrap the entire code and rewrite the program. It takes ten years to learn programming in a chosen language[19]. There are unlimited number of possibilities to solve a problem and there are no strict constraints or nature imposed limits except those stemming from the complexity that is difficult to determine in advance.[10]

Software engineers[3] are highly individualistic and they resist being forced into an organizational structure. The speed of technological advancements hinders both planning and organization of software projects. New and powerful components appear during the project development forcing redesign.

The benefits of tools deployed are difficult to assess in advance.[4] In summary, programming is:

1. Human Performance
2. Social Activity [Peer Support, Mentoring, Professional Societies]
3. Individual Activity
4. Tool - Based

Software began to be understood as a simultaneous crafting of Art, Science, Discipline and Psychology. The aspects of crafting the program amply illustrates the cognitive complexity that is far beyond mere coding skills. Programming highlights the human element as psychological factors on programmers' efficiency, effective creativity, and well-being.

PROGRAMMING AND THE BRIAN

Programming helps one to understand technology, visualize data and enhance problem solving skills. Programming improves the logical thinking skills by allowing the problems to be seen from a new perspective. This can be applied to any scenario both in the personal and professional life.

Algorithmic thinking[7] ideally is to work towards a generic formula for every solution of a given problem. It is eventually an expression of the solution in the form of a series of ordered steps with each step having a clear definition. Algorithm enables the programmers to visualize problems using the abstractions of data flow and the transformation of input into output. An algorithm with a good notation clearly establishes the efficiency, scalability, and maintainability of the coded solutions.

"The Algorithm's coming-of-age as the new language of science promises to be the most disruptive scientific development since quantum mechanics." - Bernard Chazelle, Eugene Higgins Professor of Computer Science, Princeton University, USA.

People tend to underestimate the difficulty of the task. Overconfidence explains most of the poor quality of software being developed. Doing it right is hard work and the adoption of shortcuts is leading to disasters. Software Testing[13] has the following four facets.

1. **Art:** There is individualism in the choice of test cases from within the space of valid inputs.
2. **Science:** The test results are independently verifiable
3. **Discipline:** The test cases must cover a wider space of the mapping from set of well-defined inputs to well defined outputs
4. **Psychology:** No programmer wants to choose test cases that fail / Report too many failure cases.

There is a popular opinion that programming warrants lot of mathematics. It does require some mathematics. The goal of programming has always been to arrive at a formula which is the solution for the given problem and lends itself to a formal proof. However, this is proving to be extremely difficult[14].

One can program in the mind without any need to relate it to the computer. Language is a wonderful thing. It can be used to express thoughts, to conceal thoughts, but more often, to replace thinking. Programming relies more on the language centers in the brain rather than mathematics. Programming language has far limited vocabulary and syntax to learn than natural language. Learning programming in a specific language is not the same a becoming bi-lingual. Other than the programming language, one needs to organize in the mind the moments on the computer screens to interact with the computer. The working memory is the brain's Random Access Memory [RAM]. To understand a line of code, a programmer requires the working set i.e lines of code before and after this line to keep in mind to trace of debug.[15] This working set is imaged in the brain and the logical reasoning within this working set is the mathematical model of the crucial abstraction. The process is replete with abstract symbols and ideas that need to be manipulated to convey higher level concepts[1].

The brain areas involved in this processing at an abstract level are the same as those associated with the verbal semantic processing.[17] Language represents ideas in the brain and a program with limited vocabulary and syntax represents logic.[9] Please see *Fig*. 1.

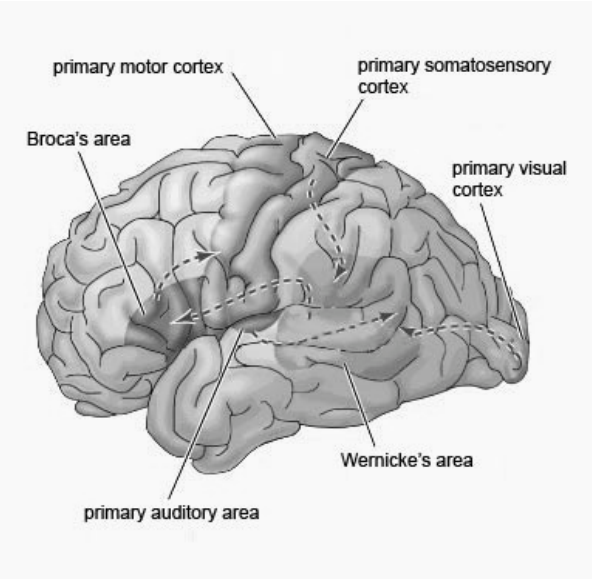


Fig. 1: Language Areas in the Human Brain

The working set needed to model the logical reasoning fits into the "Working Memory" of the human brain seen in *Fig. 2*. "Working memory"[18] is at the core of many real – world functions through a vast set of mnemonic processes and associated brain/neural networks related to the basic intellectual abilities.

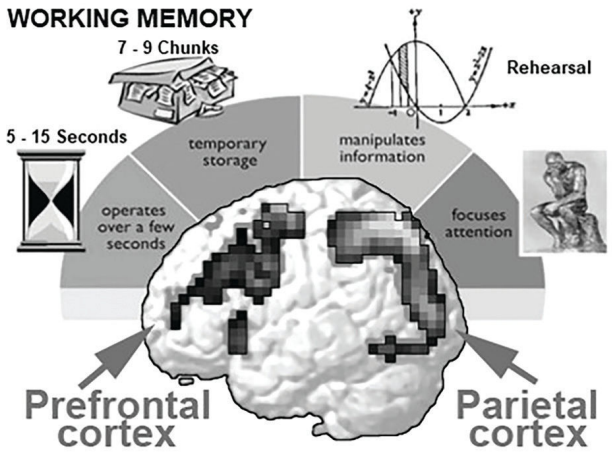


Fig. 2: Working Memory of the Human Brian

The "Working Memory" processes and interactions[2] are seen in *fig. 3*.

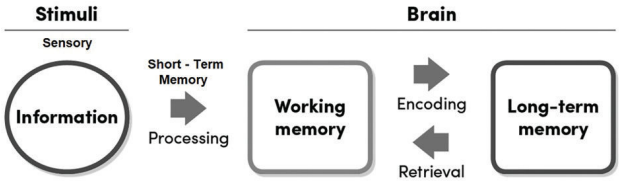


Fig. 3: Working Memory Proceses

PROGRAMMERS ON NOOTROPICS

At least in theory, the brain of a programmer is good at abstraction, language and the Working Memory[12]. Working memory is related to short-term memory, but it is active for slightly longer time and is thus involved in the manipulation of information. Concentrating on the act of programming catches on. The trouble is shaking off this spell. Programming really confirms to the philosophy of "work is its own reward".

However, the nature of programming is such that, the following traits are commonly seen in the best of brains doing computing.

- **Frustration:** Programming the solutions for complex problems often times shakes the programmer out of the comfort zone due to the bugs, and errors. The trial-and-error method for tracing the errors and debugging when there is lack of clarity of the solution may lead to seemingly inevitable frustration and feelings of inadequacy / incompleteness.
- **Burnout:** The demanding nature of programming, coupled with the stiff deadlines often lead to mental fatigue and burnout. The typical symptoms include

exhaustion, cynicism, and significantly reduced professional efficacy that clearly shows on one's quality of life and work.

Nootropic also known as 'cognitive enhancers' are drugs that some people use to hopefully improve memory, increase mental alertness and concentration as well as boost energy levels and wakefulness. There are many different nootropics. Some of them are pharmaceutical drugs that treat conditions such as sleepiness or narcolepsy. Prescribed by a qualified medical practitioner they improve attention span and focus in people with attention disorders. However, some healthy people also tend to use these drugs hoping to improve their cognitive performance and memorizing skills.

Medically, the nootropics may help mask or mitigate fatigue, procrastination or boredom. Usually, these drugs do not make people more intelligent. The effect of these drugs is experienced only when they are chemically within the body. The body needs their replenishment at regular intervals. Such a requirement may cause dependence and can have a wide range of side effects making the young programmers highly vulnerable as they progress in the profession.

Nootropics are also known as Smart Drugs, Brain Boosters, Memory Boosters, Neuroenhancers, Drive Drugs and Study Drugs.

The research is still inconclusive but the early results indicate that they may act on a variety of different systems within the human body and mind simultaneously. Some researchers report that Nootropics can cross the Blood – Brain – Barrier to increase blood flow to the brain and hence more oxygen. Some of the nootropics may raise the adrenalin levels and produce effects like drinking large amounts of caffeine. Use of dopamine tends to result in dependence. Eugeroics are a class of drugs that promote wakefulness and alertness.

BRAIN IMPLANTS

Neurosurgeons all around the world have been implanting electrodes into the human brain for decades to calm down the irregular spikes and spurts in the electrical activity that is responsible for some symptoms. The current brain/neural implants are much more sophisticated and they allow full-duplex or two-way communication between the brain and device or devices,

Machine learning and artificial intelligence are also expected to make the neural implants more sophisticated and intelligent. Integrating neural implants with other technologies, such as virtual and augmented reality promise new possibilities for treating and enhancing human cognitive function. "Brain – as – Computer" metaphor is now much cliched.

"Descartes was impressed by the hydraulic figures in the royal gardens, and developed a hydraulic theory of the action of the brain. We have since had telephone theories, electrical field theories, and now theories based on computing machines… We are more likely to find out how the brain works by studying the brain itself, and the phenomenon of behavior, than by indulging in far-fetched physical analogies." - Karl Lashley, Neuroscientist, 1951.

Psychology of Programming will be vastly improved with Brain Implants as seen in the *fig*. 4.

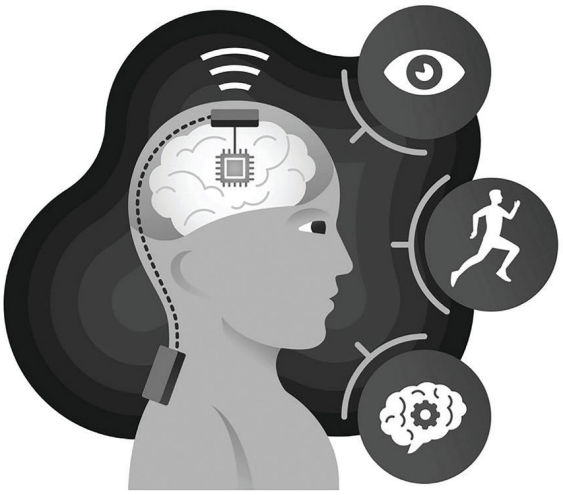


Fig. 4: Brain Implants

There are many technical challenges involving the collection, export, and the interpretation of the signals from the brain. It is a promising science in its infancy. The surgically invasive brain related methods require appropriate rules and ethics that are evolving.

Programming and Formation of Habits

In mastering any profession, the whole man is involved, body and mind, sensation and movement, thought, interest, imagination, will and similar such innumerable known and unknown aspects of our psycho-physical life. Habits emerge because the brain is constantly looking for optimizing effort[5]. The process by which the brain converts a sequence of actions into an automatic routine is known as "chunking". It is at the root of how habits form. Habits form when new behaviours become sub-conscious.

The" Min – Max" is a common sensical habit formed in many good programmers. A programmer most frequently attempts to "Minimize X and Maximize Y". For example: minimise effort and maximize effect or minimize losses and maximize gains.

Diverse changes of activity within and across circuits encode habit formation.[6] Habits are differently processed by striatal output pathways. Key questions concern action sequences, avoidance habits, and habit modulation. Habits may involve components with distinct neural signalling processes.

Habits are actions that are triggered by cues, such as a time of day, an activity, or a location. They culminate in a feel-good reward that when repeated often times will blur the distinction between the cue and the reward.

"If we can find the neural basis of programming and see which other cognitive functions share their neural basis, it provides a hint to what skills are needed to learn to code." - Yun-Fei Liu, John Hopkins University Neuroplasticity and Development Laboratory, USA.

CONCLUSIONS

In computer programming, more emphasis is placed on the mechanization rather than on the human element. A proper study of programming positions the individual in the context of the work and its social conditions. Psychology of Programming is an interdisciplinary area spanning into computer programmers' cognition[16], design and development of tools and methods for programming and related activities such as education and training. This paper is a unique attempt to bring together certain medical aspects that may enable a better understanding of the impact of programming on the brain of the programmer.

REFERENCES

1. Adam Sinicki, Hacker's Brain – The Psychology of Programming, The Bioneer, 21 November 2015.
2. Atkinson, R.C.; Shiffrin, R.M., "Chapter: Human memory: A proposed system and its control processes". In Spence, K.W.; Spence, J.T. (eds.). The psychology of learning and motivation. Vol. 2. New York: Academic Press. pp. 89–195, 1968.
3. Bloom, Benjamin (ed.) Developing Talent in Young People, Ballantine, 1985.
4. Brooks, Fred, No Silver Bullets, IEEE Computer, vol. 20, no. 4, 1987, pp. 10-19.
5. Bryan, W.L. & Harter, N. "Studies on the telegraphic language: The acquisition of a hierarchy of habits. Psychology Review, 1899, 8, pp. 345-375
6. Charles Duhigg, The Power of Habit", Random House, New York, 2012
7. Chase, William G. & Simon, Herbert A. "Perception in Chess" Cognitive Psychology, 1973, 4, pp 55-81.
8. Dmitriy Malets, The Mind Behind the Code: Exploring the Psychology of Programming, https://technorely.com, 7 February 2024.
9. Gerald M Weinberg, The Psychology of Computer Programming, Van Nostrand Reinhold, New York, USA 1971.
10. Hayes, John R., Complete Problem Solver, Lawrence Erlbaum, 1989.
11. J.N. Buxton and B. Randell (Eds.) Software Engineering Techniques: Report on a Conference Sponsored by the NATO Science Committee, Rome, Italy, 27-31 Oct. 1969. Scientific Affairs Division, NATO, Brussels (May 1970).
12. J.R. Binder, C. F. Westbury, K.A. McKiernan, E.T. Possing and D. A. Medler, Distinct Brain Systems for Processing Concrete and Abstract Concepts, Journal of Cognitive Neuroscience, Vol. 17, No. 6, pp 905–917, 2005.
13. John Von Neumann, The Computer and the Brain, Yale University Press, New Haven, USA, 1958.
14. Jorma Sajaniemi, Psychology of Programming: Looking Into Programmers' Heads, Human Technology, Vol. 4, No. 1, pp 4–8, May 2008.
15. Lahiri Choudhury, D.K, "Of codes and coda: meaning in telegraph messages, circa 1850-1920, Historical Social Research, Vol. 35, No. 1, pp. 127-139, 2010
16. Lave, Jean, Cognition in Practice: Mind, Mathematics, and Culture in Everyday Life, Cambridge University Press, 1988.
17. Lewis Crawford, The Language System, Brain Networks, 25 June, 2022.
18. Nyberg, L and Eriksson, J., Working Memory: Maintenance, Updating, and the Realization of Intentions", *Cold Spring Harbor Perspectives in Biology, Vol. 8, No. 2*, pp 1 – 16, 2016
19. Peter Norvig, "Teach Yourself Programming in Ten Years", norvig.com, 1998
20. P. Naur and B. Randell (Eds.). Software Engineering: Report on a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968. Scientific Affairs Division, NATO, Brussels (Jan. 1969).