RESEARCH ARTICLE OPEN ACCESS

Challenges of Computer Science

Dimpal Kumar*, Aman Singh Rawat**, Satyam Jha***

*(School of Computer Science and Engineering, GalgotiasUniveristy, GreaterNoida, UttarPradesh, India
Email:kdimpal28@gmail.com)

**(School of Computer Science and Engineering, GalgotiasUniveristy, GreaterNoida, UttarPradesh, India
Email:amanrawat161204@gmail.com)

***(School of Computer Science and Engineering, GalgotiasUniveristy, GreaterNoida, UttarPradesh, India
Email:satyamjha652001@gmail.com)

--***********************--

Abstract:

The issuesis that computer science will face in the twenty-first century are discussed in this paper. To predict the long-term future and identify the pertinent issues for a science that is only 60 years old is pretty dangerous. However, it is crucial to recognize what has been accomplished thus far and what issues are anticipated or must be resolved in the near future. In light of this, we discuss issues with computer architecture, parallel and distributed computing, programming paradigms, networking computational complexity models, hardware and database challenges, and operating difficulties we face in our day today life.

Keywords: ***challenges, computer science, programming paradigms,21st century***

--**************************--

I.INTRODUCTION

Computer science encompasses programming languages, algorithms, and structural elements. Using these foundational elements as a base, we develop new technologies and applications. It teaches you how to create scalable, modular projects that follow a clear logic and are neat, clear, and reusable. The kind of thinking required to be precise, analytical, and concise is improved by philosophy. Similar to solving engineering problems, you divide complex issues into manageable chunks and develop methodical solutions, typically at a high level of abstraction.It has been 50 years since computer science and engineering entered the modern era. Our field has made significant contributions to society during this vibrant and dynamic time. There is an increasing sense of limited possibilities as the field has advanced. This is comprehensible. There is less uncharted terrain in the areas of the field that are more developed. The easy pickings have already been made. It’s more challenging for fresh ideas to have significant impact on conventional wisdom or have a significant impact on business practices. On the other hand, because of the mass market's multiplicative power and the pervasiveness of information technology in contemporary culture, advancements in mature fields can have a significant impact. a technical elite no longer has exclusive access to information technology. Even if advancements in mature fields may be tougher to come by in the current era, their overall effects on society can be profound. Our field is in a golden period right now, and the potential for study is unparalleled in many significant respects. Our field's established areas have grown older, yet its boundaries are still being pushed. We are barely scratching the surface of what computer science and engineering can do in many domains, which has apps in all aspects of modern existence New chances for innovation be continually being created by the constant change in the underlying electronics technology. Every day, computational barriers are being broken down, bringing previously unsolvable issues within reach of workable technical solutions. combination of tech and a deeper understanding of underlying issues is removing the barriers that have previously prevented practical solutions from being implemented, as evidenced by the imminent widespread use of advanced computer vision and real-time natural speech recognition systems. An emerging trend in new period is focus on system projects meant for illustrate novel research concept and enable their evaluation in practical experimental settings. These studies increasingly demand coordinated efforts from huge teams rather than just the limited efforts of individual scholars. Finding new methods for structuring academic research in this period is one of the obstacles we must overcome to enable the creation for larger research teams can handle more complex systems. Although there have numerous attempts in this direction over the years, the results have been inconsistent. If academic research is to continue making significant contributions to society and play an important part in the development of advanced systems, we must improve our performance. The remainder of this paper focuses on two research issues in systems that, although providing enormous practical benefits to society, pose considerable research challenges, pose significant research challenges, call for fresh approaches, creative thinking, and significant R&D efforts to be overcome.

In the field of computer science, a problem is said to as unsolved when two or more experts in the field cannot agree on its solution. The "science" that supports the field of computers is theoretical computer science, or the scientific basis for computing (TCS). Nowadays, theory is an important component of practically everything we do in the world of computing. For almost all models, it acts as a beacon. is the scientific foundation for computation (TCS). Nowadays, theory is an essential component of almost everything we do in the field of computing. We'll discuss a few problems with the field of computers in this article. Although no solutions are provided here, I think that once the fundamental concept is understood, these challenges might be easily addressed.

II.THE DRIVING CONCEPT THEORY

The following fundamental idea can help you better understand Theory.

A. *Discover the Theoretical*

By leaving out extraneous details, we are able to comprehend the content clearly. This can then be looked at and altered to suit the requirements.

B. *Discover a Link:*

Make a connection between the current issue and a comparable historical one. The commonalities between them can be discovered once a link has been formed. The principles from the previous model can then be used, enabling a deeper understanding. These two essential concepts can be used to fully comprehend the theory underlying any model.

III. NETWORKING DIFFICULTIES BRINGING COMPUTING AND COMMUNICATION TOGETHER

One critical challenge for our field is to take a significant step forward in networking and communications. While the Internet has been a huge success, the limits of its basic protocols have become a major hindrance to advancement. Over the last ten years, there has been a highly effective attack on the underlying challenge of improving Internet router performance. Enhancing communication network capabilities throughout the course of the next 10 years will need the use of both embedded computing components and reconfigurable hardware. The most challenging task at hand is to conceptualise an effective large-scale programmable network model that will allow a global infrastructure of network-embedded computational resources to be made available to application developers with innovative ideas for new services aimed at assisting end users. While recent active networking projects have sparked interest in this field, The most popular paradigm in the field of active networking (packets treated as programmes) does not seem to hold a lot of promise. To allow the underlying network infrastructure to offer suitable low-level services to application developers, more defined ways are needed, appropriately allocate resources (on a network-wide scale), and provide the isolation and protection required to ensure consistent, high-quality service to end users. The design of routers with embedded computational capability presents considerable hurdles. The performance tradeoffs for processors in this context differ significantly from those for regular processors designed primarily for desktop computer situations. This environment's computational workload is well-suited to systems designed for task-level parallelism, with many processors on a chip and various hardware contexts per processor being recommended. In this perspective, the potential function of reconfigurable hardware is extremely exciting. In recent years, the capabilities of reconfigurable hardware have grown tremendously, and a significant reduction in the performance gap with fixed hardware devices. If we can build efficient models for integrating reconfigurable hardware into flexible and general-purpose computing engines, as well as novel approaches to speed up the setup process, there is a huge opportunity here (place & route computations for large FPGAs today can take hours; this must be reduced to seconds).

A.*Automated Communication Protocols*

Networking and software engineering both have communication problems. There are, however, some more practical solutions to the problem because networking communication protocols are so basic. One such choice is the use of I/O Automata. Manual labour is used mostly in the specification, creation, and verification of communication protocols. The main challenge here is automating the creation and verification of communication protocols rather than doing it by hand.

B. *Routing and scheduling techniques:*

For node failure recovery and congestion control, there are two the routing protocols which internet uses are known as MPLS and OSPF.But when more administration is required for both protocols, the amount of communication between routers and computation increases. In order to address scalability difficulties, the network is currently divided into sub-area routes. Therefore, improving real-time adaptive routing techniques is the current issue at hand. Because of the significant "thrashing" instances that took place when real-time adaptive routing protocols were first introduced in the early ARPANET, real-time adaptive routing methods have drawn a lot of criticism from network operators and experts.

IV. HARDWARE CHALLENGES

Over the previous decade, significant breakthroughs have been made in computer-aided design tools and hardware design processes in general. Hardware description languages enable digital hardware designers to work at much higher levels of abstraction than previously possible, and associated CAD tools automate important elements of the design process. However, in significant aspects, hardware design has made little progress in the last thirty years. While CAD technologies aid in the automation of the design of smaller hardware components, they offer little or no help in the assembling of these components into bigger, more complex systems. The burden of managing component interfaces and coordinating the concurrent operations of the dozens of components found in complicated processors, Designers are solely responsible. Since there are no automated tools to support this, designers are left with no choice but to use manual methods. These involve painstakingly documenting the data formats and timing requirements of every interface and creating complex timing plans to coordinate the numerous activities occurring within the system. Here, there are significant intellectual obstacles. When it comes to envisioning and managing parallelism, automated technologies are of limited intellectual assistance to the human designer, even in the software arena. The hardware industry presents fresh difficulties for solving this issue in addition to the possibility of new approaches. Although it is frequently employed in hardware, software systems rarely use timing-driven coordination.This opens the door to automated solutions that use timing-driven coordination approaches while relieving the designer of the time-consuming and error-prone process of creating extensive timing plans to achieve the desired impact. Because parallelism is so widespread in the hardware realm, better techniques for managing parallelism can have a tremendous impact on hardware design. Parallelism is not a minor subset of the hardware design domain; it is its essence. A successful attack on this topic can have immediate practical benefits while also potentially deepening our fundamental understanding of parallel computation and helping us to use parallelism more effectively in other circumstances.

V. DATABASE CHALLENGES

A. *Data storage*

Memory storage expands along with the database size. Thus, the concept of memory hierarchy is introduced. When aiming to lower the cost of running database queries, We need to better predict the costs of retrieving data and storing intermediate results in cache, RAM, disc, tape, and so on.

Here, the challenge is to devise fresh methods for running queries on massive amounts of data while utilising secondary-storage cost models and, if feasible, developing distributed and parallel processing.

B. *Optimization of Query:*

The response to a question is a query plan. This intricate technique was built using fundamental database operations like projection, join, etc. Based on the SQL language, a query can have any number of query plans. The process of determining the most effective strategy is known as query optimization. Dealing with complicated processes and query optimization when sub-queries come from several sources is a difficulty.

VI.OPERATING DIFFICULTIES

A. *File System Security:*

Access into private files can be obtained from a collection of programmes. Thus, security remains in jeopardy. For instance, the mail sent by the owner is accessible to both him and his adversary. An adversary may use a mail software to access emails sent from the owner. In this instance, both have the same access pattern.As we utilise and rely on "mobile code" more frequently, this issue will worsen. Modeling the safe execution of untrusted code is a challenge. Browsers now make an effort to do this, but more accurate modelling and analysis are required given the regular disclosures of security flaws in their protocols.

VII.DATA SECURITY

The data security dangers posed by technical advancements, as well as how consumers, firms, and other organisations use that technology, will be important concerns in the coming year. Data concerns infiltrate nearly every evolving technology. The risk of data breaches has dramatically grown as a result of the massive volumes of corporate and personal data being shared and kept online, and firms now need to prepare swift responses to comply with a patchwork of state and federal data breach legislation. While such regulations continue to raise the bar for data security measures, contracting parties want these standards to be held more accountable as well. Some of these concerns could be addressed by improved encryption and biometrics.

A. *Data Access Enhancement:*

The Response Time is the key bottleneck for operating system functions.The data layout must be planned based on the most common access mode in order to increase response time effectiveness. Given a fixed data architecture, the issues are developing automatic strategies for "optimal" dynamic data rearrangement in response to access patterns, as well as fast online reordering algorithms for access requests.

VIII.USING INSTRUCTION LEVEL PARALLISM

A. *Challenge in computer architecture.*

Recently developed CPUs are employing an increasing amount of hardware to boost speed in a number of methodsincludes parallelism at the instruction level. Creating a processor is the actual challenge here which makes efficient use of computer resources and an algorithm that makes it clear what the programmer is in charge of.

IX.SOFTWARE ENGINEERING CHALLENGES

A.*Software Description:*

Requirements check whether the software system adequately satisfies user needs. They are, however, frequently underspecified since they are stated in informal conversational language that is overly wide and full of repetitious and inconsistent standards. While the MSC (Message Sequence Chart) offers a precise formal mechanism for defining intended behaviour, it falls short of having the expressive power needed to fully characterise what we might want from a system. Despite LSCs (Live Sequence Charts)have just been introduced as a method for improving expressivity, it is not yet obvious whether they are exactly what is required. Additionally, there is no automated transition from an informal to a formal specification at this time. The difficulty lies in developing methods for formalising software systems that accurately describe expected functionality and behaviour.

B. ***Automatic Prerequisites Transformation to System Design:***
While system architecture specifies how they operate and interact, the requirements discuss the overall behaviour of the system. Typically, they are depicted by means of a State-Chart or another automata-theoretic model. The distance between the first and second is considerable. The task at hand is to provide tools and algorithms for translating specifications into system designs, as well as limitations on the kinds of specifications and system designs that are permitted. We can save time and effort in this way.

C. ***Comprehend Complex Systems:***
Typically, interrelated portions of a system architecture consisting of software systems are designed. The software usually malfunctions or only operates partially when some crucial design is ignored. Creating a theory that can comprehend the incomplete programmes is the difficult part of this situation. The testing and evaluation of designs that might be partially built should be based on this principle.

D. ***Conversion of System Design into Code:***
A programming language is needed to convert a system design into code; this process is known as high-level compilation. This entails using a proprietary algorithm that hasn't been thoroughly researched. The task at hand is to provide an automatic compilation method that automatically converts the design into code while providing a detailed description of the procedure. Theoreticians still have a lot of options in this field.

E. ***System Query Language for Software:***
Considering the complexity of the entities, modern software systems are challenging to comprehend. In order to comprehend the properties of the system, the present software system can be considered as a database. The challenge is to develop a query language, such as SQL, that makes questions about the programme. additionally, a system that could create an algorithm for a specific situation should be developed. The inquiries posed by the Query language created for this can be answered using this algorithm.

X.OBJECT ORIENTED PROGRAMMING DIFFICULTIES

A. ***Programming is streamlined:***
Data come first and control flow second in object-oriented programming. Its capabilities are very helpful in fostering modularity and code reuse while offering encapsulation and protection techniques. Java and C++ are currently well-liked programming languages. But in an effort to make programming simpler, Language theorists are continually researching the structure and function of such languages. Creating tools to comprehend object semantics, particularly how they interact with types, is a difficulty.

B. ***Programming Language Expression:***
A collection of partial recursive functions in any programming language is the set of defined functions on integers. The majority of programmers do, however, distinguish between languages. While certain programming languages are helpful for quickly creating simple programsOthers, however, may be better suited for developing large-scale, maintainable software systems. Finding a technique to explain the distinctions between programming languages is the difficult part. Comparing the capacity to localize program design choices is one such approach. A different approach is to contrast languages based on

how well they can express operations on program modules.

XI.CENTRAL COMPUTER PROGRAMME

A.*Protocol safety:*

Simple strong formalisms can be created for protocol security. Formalism in use now can only detect protocol errors. No formalisation provides security for a protocol. The difficulty lies in creating an automated tool to check the validity of the protocol. By creating such a formalism, we may avoid the issue of implementation defects that compromise software security as well as the security risks associated with the underlying protocol. The majority of the time, the very real threat of several instances of different protocols running concurrently is not guaranteed by present security principles.

XII.CONCLUSION

In this whole report, we spoke and tellsabout the main difficulties faced by various computer science professions. Maybe I'll include the potential answers in my next paperwork. Despite being at the core of the field, These underlying difficulties provide the following generation an opportunity. Finding a solution to each of these issues not only enhances our comprehension of the subject at hand but also simplifies our work. Every company, non-governmental organisation, and governmental agency strives to create cutting-edge solutions to problems in the real world. In the present world, knowledge generation will actually become the most important tactic for success and survival, not only for an individual but also for a corporation.

REFERENCES

[1] Chu S., The Laser Cooling and Trapping of Atoms and Biomolecules, 2000 Ernest W. Guptill Memorial Lecture by 1997 Nobel Laurerate in Physics, Dalhousie University, Halifax, NS, Oct.2000.

[2] Eberbach E., Expressiveness of $-Calculus: What Matters?, in: Advances in Soft Computing, Proc. of the 9th Intern. Symp. on Intelligent Information Systems IIS'2000, Bystra, Poland, Physica-Verlag, 2000, 145-157.

[3] Eberbach E., $-Calculus Bounded Rationality: Process Algebra + Anytime Algorithms, Proc. of the Intern. Conf. on Recent Advances in Mathematical Sciences and a Symposium on Challenges in Mathematical Sciences in the New Millennium ICRAMS-2000, Indian Institute of Technology, Kharagpur, India, Dec. 20-22, 2000.

[4] Challenges in Theoretical Computer Science – Paperwork by Ed Clarke, David Harel, Anna Karlin, Gerard Holzmann, Daphne Koller, Butler Lampson, Jeff Ullman, and Uzi Vishkin.

[5] Database System Implementation, Prentice Hall, 1999,Garcia-Molina, Ullman, and Widom.

[6] Challenges at the Interface of Data Analysis, Computer Science and Optimization by Gaul.W.A; Geyer Schulz.A Human Language Technology-Challenges for Computer Science and Linguistics by Vetulani.

[7] Garzon M., Models of Massive Parallelism: Analysis of Cellular Automata and Neural Networks, An EATCS series, Springer-Verlag, 1995.

[8] Almstrum, V., Hazzan, O. and Ginat, D. (in press, December 2004). Special Issue on Import/Export Relationships to Computer Science Education Research, Computer Science Education 15(3).

[9] Dale, N. (2002). Increasing interest in CS ED research, SIGCSE Bulletin inroads 34(4), pp. 16-17.

[10] Fincher, S. and Petre, M. (2004). Computer Science Education Research, Routledge Falmer.

[11] Goldweber, M., Clark, M. and Fincher, S. (2004). The relationships between CS education research and the SIGCSE community. SIGCSE Bulletin inroads 36(1), pp. 147-148.

[12] Valentine, D. W. (2004) CS Educational Research: A metaanalysis of SIGCSE Technical Symposium proceedings. SIGCSE Bulletin inroads 35(1), pp. 255-259