The Computing Disciplines

Teres George and Vineeth Paleri
Department of Computer Science and Engineering, National Institute of Technology Calicut

From supercomputers to our intimate smartphones, computing technology has had an immense influence on human life as never any other modern-day technology has. An aspiring student will be perplexed by multiple branches and a great variety of options to choose from. This article is a broad survey of a convenient classification of disciplines under the computing domain for aspiring students and educators in the field.

Though this article derives its basic concept from *Computing Curricula 2005*, the report by the professional bodies, including the *Association for Computing Machinery (*https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2005-march06final.pdf*),* the views expressed by the authors in this article are their own.

Need for Classification

Unlike many other disciplines, computing disciplines encompass a broad area cutting across science, engineering, and application in every conceivable field of knowledge. *A discipline emerges when a corpus of knowledge can be studied in relative independence and depth*. The computing disciplines are so vast that it is impossible for a person to master all the knowledge falling under them. Hence, it is essential to classify the related knowledge and skills into different disciplines to enable an individual to be an expert in the chosen field.

The Domain of the Computing Disciplines

Figure 1 gives a graphical representation of *the domain for the computing disciplines*. The vertical axis looks at the domain in terms of *software* and *hardware* aspects. The software is further divided into *application software* and *system software*. System software acts as an interface between the hardware and the application software, enabling convenient and efficient use of the computing resources. System software includes essential supports such as operating systems, compilers, and database management systems. Application software is the software for application in a specific area.

The horizontal axis divides the domain into *Theory, Design & Development,* and *Maintenance*. The *theory* part refers to the theoretical aspects of computers and computing, which comprises creating new knowledge and identifying new methods for development. The *design and development* aspect refers to developing software or hardware using the established theory and methods. The *maintenance* part refers to keeping the computer system, i.e. both hardware and software, functional and updated.

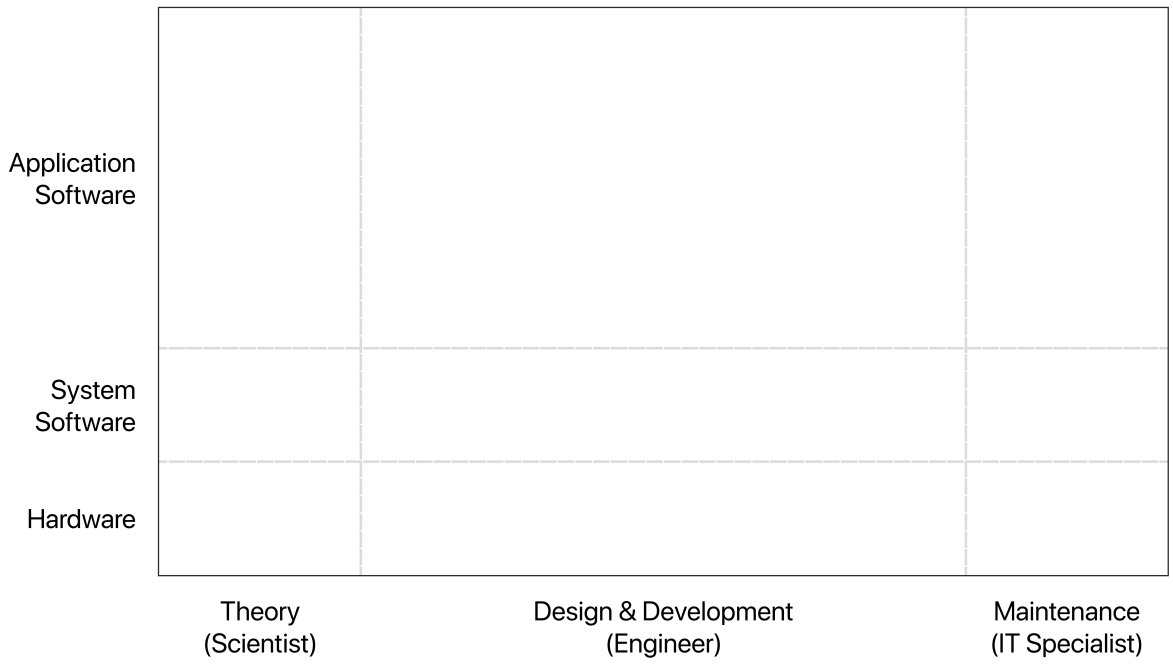

Figure 1: The domain of computing disciplines

The professional names *scientist* and *engineer* correspond to *theory* and *design* respectively on the horizontal axis. It is difficult to draw a hard line between the roles of a scientist and an engineer. However, *separation of concerns* regarding their roles helps them for effective focus and better output. A rule of thumb to differentiate the roles of a scientist and an engineer is: *a scientist builds to learn, whereas an engineer learns to build.*

Let us now see the graphical representation of the knowledge and skills that characterise each discipline. Overlaps between disciplines suggest that a professional will do good to understand a little bit from other disciplines while focussing on their domain.

Note that the area marked for each discipline indicates an ideal scenario. For example, the time and cost involved in maintenance is a significant part of software today, which should reduce considerably as the software engineering discipline matures.

Computer Engineering

This discipline addresses the theory, design, and development of computers; i.e. the body of knowledge in electronics and communications characterises the discipline (Figure 2). A computer engineer fundamentally focuses on the hardware, i.e. the theory and development of computers and associated devices. The computer engineer may specify the interface between the hardware and software of a device so that a software engineer can develop software for that

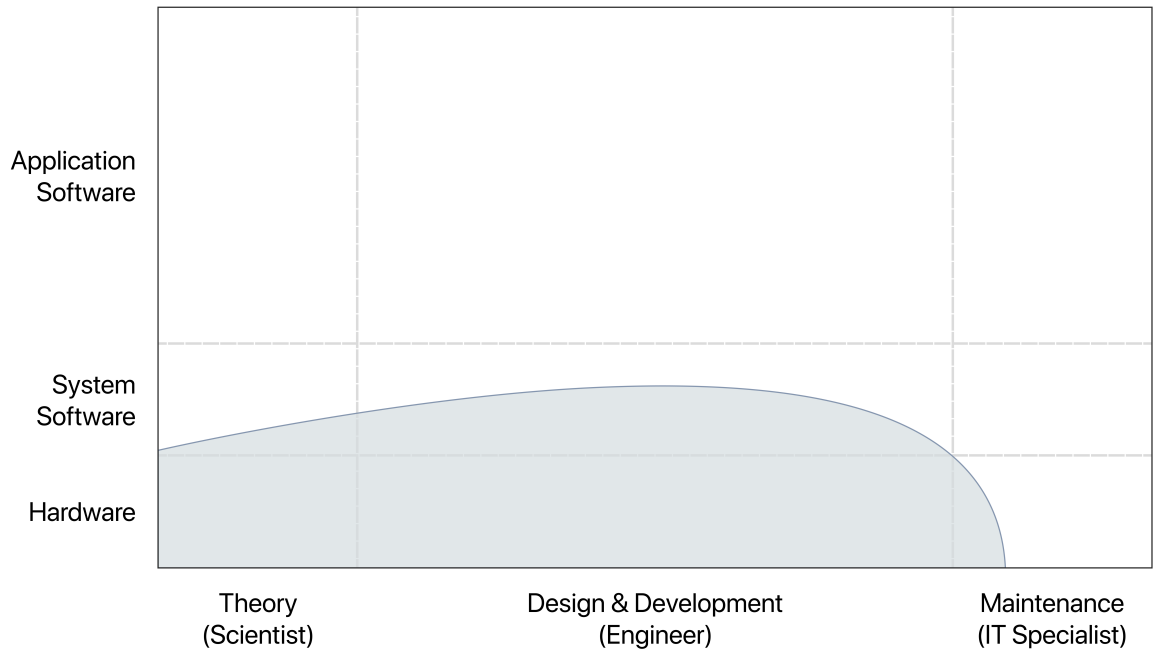

Figure 2: The domain of Computer Engineering

device independently. It is also possible that the computer engineers develop the software themselves, especially when the devices are for limited specialised applications.

Computing Science

This discipline is more widely known, even in academic circles, by the name *Computer Science,* which is a misnomer. Since the discipline of Computer Engineering addresses the hardware of a computer system, it would be more appropriate to name the discipline *Computing Science* rather than *Computer Science.* The Computing Scientist has to develop the theory, i.e. the principles and methods, on all aspects of computing, mainly the reliability and efficiency of software (Figure 3). One may classify the significant roles of a computing scientist as follows based on different aspects of computing:

Reliability: Creating the necessary theory and methods to develop reliable software, i.e. software that meets the requirements specified by the user.

Efficiency: Finding efficient solutions to problems. They should find the solutions to a problem, assess the efficiency of each solution, and decide the best solution among them.

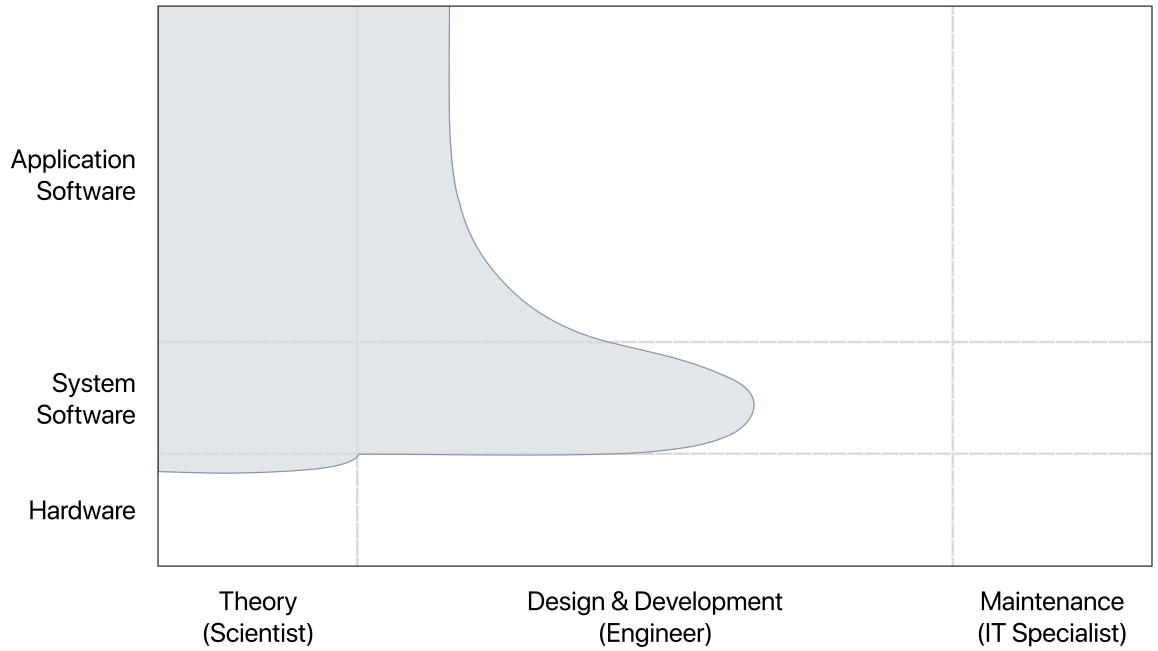

Figure 3: The domain of Computing Science

New ways of computing: Striving for innovation to make computing more user-friendly. From isolated machines to vast worldwide network computing underwent remarkable evolution. Distributed computing opened up a new area of study on Computer Networks & Security.

The roles of the computing scientist decide the core knowledge that characterises the discipline. The reliability aspect demands knowledge including, *Logic & Proof, Principles of Programming, Programming Language Concepts,* and *Type Systems*. The efficiency aspect of the solution to a problem requires knowledge in the *Theory of Computation, Data Structures,* and *Algorithms & Complexity.* Search for new ways of computing, both for the user's convenience and the efficient use of the computing resources, led to conceptualisation, realisation, and development of *Operating Systems, Compilers, Data Base Management Systems, Computer Architecture*, and *Computer Networks*.

Software Engineering

Software Engineering is the discipline for developing software which are reliable and efficient, is affordable to build and maintain, and satisfies all the requirements that customers have specified for them (Figure 4). Software engineers usually work in the development of large software.

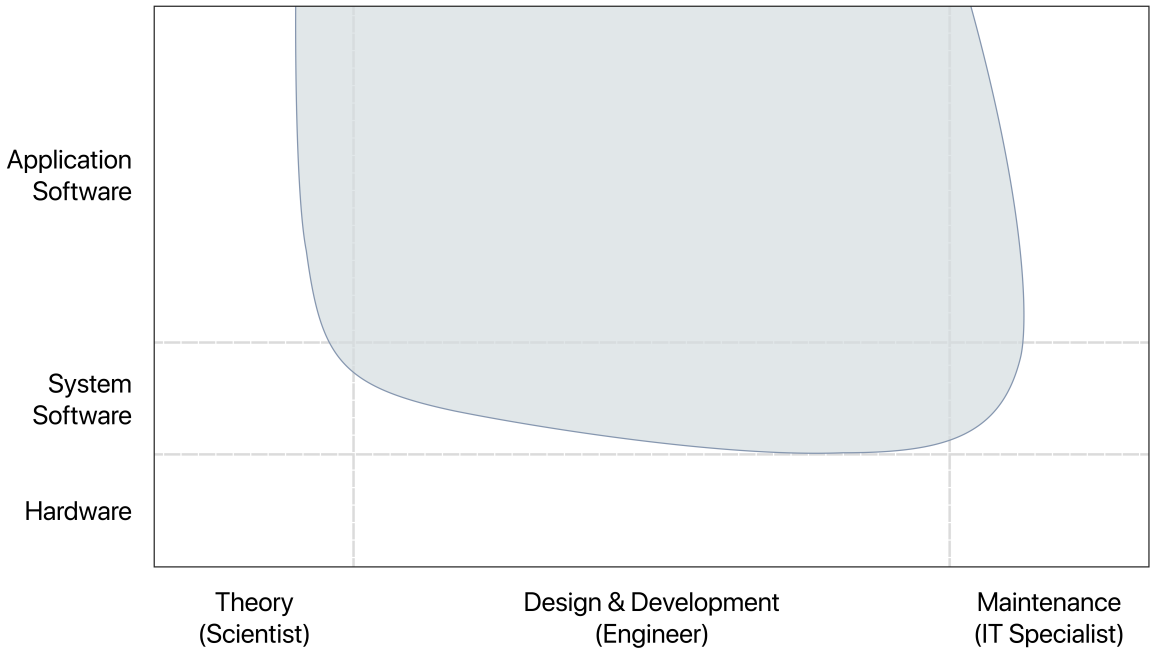

Figure 4: The domain of Software Engineering

Software Engineering is an emerging discipline, and a software engineer needs to develop software products with a guarantee as in other engineering disciplines. The discipline is yet to mature in this respect.

Information Technology

Information Technology (IT) specialists should be able to assess an organisation's hardware and software requirements, select them based on the requirements, deploy them, and finally maintain them (Figure 5). An IT specialist should be well versed in the current trends in both hardware and software. They may not develop software; instead, they usually select the particular organisation's software and deploy them.

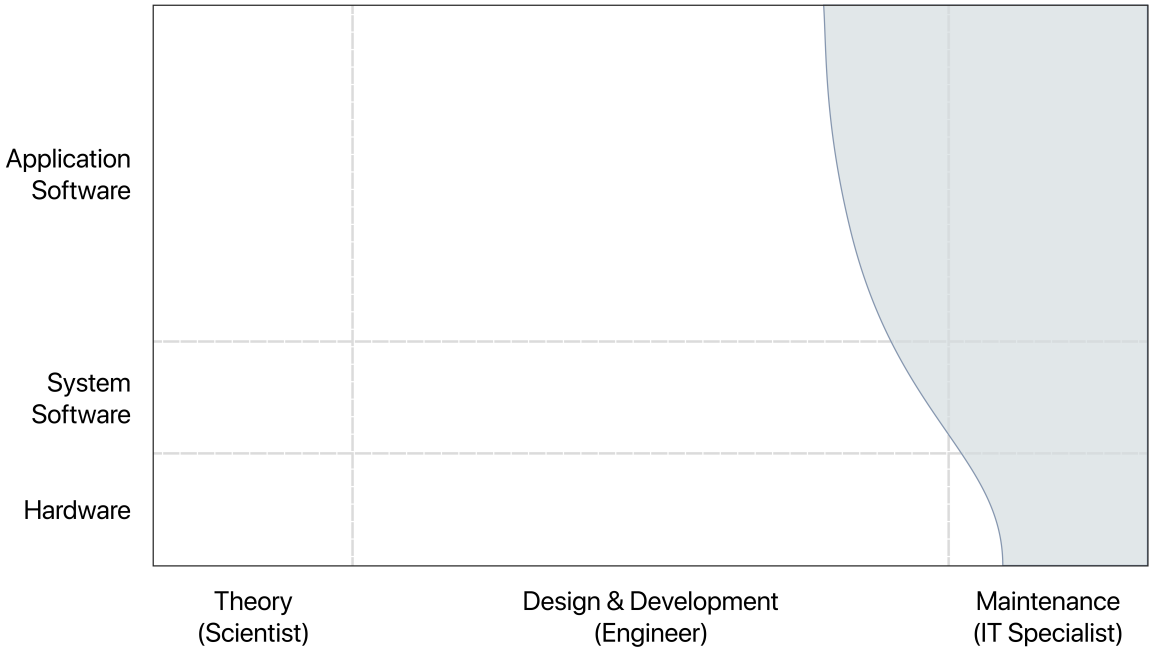

Figure 5: The domain of Information Technology

New computing areas keep evolving as we realise the benefit of applying computers to new fields. Examples of such domains include *Information Systems* and *Bioinformatics.*

Knowledge about the domain of the computing-related disciplines is an absolute necessity for educators designing the curricula and syllabi for any of the degree programmes in the area. For example, an academic who decides to offer a degree programme in Information Technology should ensure a sufficient body of knowledge, distinct from other disciplines, to justify that programme.

Educational programmes without the necessary coherent body of knowledge can mislead the scholar regarding the programme's intent, causing harm to the society in terms of the poor service delivered, viz., the software engineers of the day producing software without the expected guarantee.