Voices on the Core of Computing

Stephen T. Frezza
Software Engineering
Gannon University
Erie, PA USA
frezza001@gannon.edu

Alison Clear
School of Computing
Eastern Institute of Technology
Auckland, New Zealand
AClear@eit.ac.nz

Abhijat M. Vichare
Persistent Computing Institute
Persistent Systems Ltd
Pune, INDIA
abhijatv@acm.org

***Abstract*— What is Computing? This simple question has been surprisingly difficult to answer, resulting in confusion about what constitutes the core of Computing. This is particularly relevant when we consider curricula for various subdisciplines of computing. Examining various issues and voices, this paper lays out an informed case for defining the landscape of computing. This work proposes a Disciplinary Boundary Model, in contrast to a socially-constructed approach for defining computing sub-disciplines. This model can be used to describe the computing landscape by examining the underlying "invariant" activity of human inquiry, and suggest that it be used for defining the landscape for various computing curricula.**

Keywords—Philosophy of Computing; Philosophy of Engineering; Computing Curricula; Computer Science; Software Engineering; Information Systems; CS+X

I. Questioning the Core

Computing is a relatively new family of disciplines, with significant roots in mathematical analysis, engineering methodologies and scientific empiricism. Recent advances in worldwide curricula have expanded various of these disciplines, such as Information Technology, Information Systems, Computer Science, Computer Engineering and Software Engineering [1]. New frontiers include the significant development in areas of Cyber Security and Data Science. While these efforts have near universal agreement to being within the frontiers of computing education, what lies at the core of computing, and how this core supports future expansions in computing education is less clear.

Computing has been evolving along multiple paths; new developments in computing have seeded new communities, which in turn have also evolved their own paths. New algorithmic approaches and discoveries are helping to drive advances across a range of fields from medicine to psychology [2]. Each of these breakthroughs has led to new collaborations and an increased demand for deeper knowledge of computing in engineering, the sciences and humanities, challenging conventional disciplinary boundaries. This expansion has led to a wide range of jobs in virtually all sectors of society and expanding the demand for computing skills to an unprecedented extent. This affects our understanding of the landscape of computing. As more academic disciplines incorporate computing into their research and educational missions [1], the boundary of what is the core of computing, its boundaries, and what is not becomes more difficult to discern.

This work aims to lay out a scholarly-informed case for defining the landscape of computing, where there is consensus, and to set up a framework for describing the boundaries of these domains for informing computing education. The goal of the work is to be inclusive of the new and emerging areas of computing, but also enable clearer identification of areas shared across the boundary of computing to other domains, and when these research and educational topics are beyond that shared boundary. For example, there are fundamental disagreements about what *is* computing [3] - not just the international language, but even the nature of what is included and why/how. Similarly there are significant contextualization issues, that is where similar terminology is used for similar, and different aspects of computing sub- and related disciplines that vary internationally [4]. This work addresses how these issues of agreement among various communities fit, and how they affect our understanding of computing, defining its core, and examining computing disciplines in the future.

A. Starting Points

The history of modern computing is not "a" history, but many intertwined histories, each with different motivations, needs, and aims, and rooted in different intellectual traditions. [3]. In its earliest forms, computing was seen as a quest to formalize human thinking into a form that could be reduced to calculation and/or be mechanized. E.g., software-plus-computer system has been called '*Denkzeug'* (think-equipment), in contrast to the '*Werkzeug'* (work-equipment, tool) of the material world [5]. While these early attempts at 'mechanizing thought' provided the mathematical and philosophical underpinnings of what we now call computing [6], they have also left a number of issues in trying to understand what lay at the core of a discipline that has had, and continues to have significant impact on our world, society and culture [7,1,2].

These initial explorations into mechanizing thinking led to the formulation of Boolean Logic and Graph Theory, and present explorations into artificial neurons and computational intelligence impact not just scientific and engineering laboratories, but most corners of the global marketplace. Responses to Alan Turing's 1950 question posed "Can machines think?" have expanded to the point where these "digital advances bring us daily benefits, but they also raise a host of complex questions and broad concerns about how technology will affect society." [2]

While this march has been exciting, one consequence has been that each computing community has evolved its own terminology along its path. For example, while one community calls a certain operation as "XNOR," the other has been calling

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

it "equality" [8]. Generally, these are issues both of contextualization and social construction. Such developments are natural as long as we focus on a specific domain but results in confusion when we examine the computing field as a whole.

II. Issues in Defining a Discipline

At the heart of defining the core of a discipline like computing are issues of *ontology* – the study of being. In this case, it is examining what *is* computing, and with this comes the necessary exploration of collective truth commitments, e.g., our collective understanding of computing. This necessarily includes the social, historical, and intellectual processes by which persons and communities create and share their understanding of what something is. This minimally entails two perspectives: a philosophical understanding of something, and a socially-constructed, historical view of something. This essay attempts to synthesize components of both.

The postmodern understanding of ontology entails developing structures of meaning [9]. These formal structures called *ontologies* include tacit and culturally-normed definitions that both vary across time and space, and represent the 'what' that constitutes something (e.g., a domain of discourse) [10]. Here, the 'what' of an ontology is a hierarchy of related concepts with arbitrarily complex relations between concepts. Attempts to use this approach in computing have been attempted. The most notable example is the *Computing Ontology Project*, which worked to establish a common language for discourse across the breadth and depth of computing related domains [11] and contributed substantively to the CC2005 curricula development efforts [12].

The fundamental difficulty with utilizing an ontology is that the understanding is always tacit; even while the work of collecting and developing agreement to the ontology, so too can the 'what' in question, e.g., the 'breadth and depth of computing' change. This limits the 'truth' or value of the ontology to the quality of the development and maintenance of the agreement within a community [13]. This implies that ontology efforts are less effective when the knowledge base for changes rapidly (as it has in Computing), and it implies that the ontology itself is continually subject to the variations in people's different contextualization of the concepts.

The contextual issues of time and language allowed the *Computing Ontology Project* to define computing at that time and for that segment of the community involved, but not the core of computing – *why* a particular topic is, or may be at the core, or at boundaries of computing.

A. Contextualization

The development and maintenance of a computing ontology is the role of the Computing profession and its representatives. However the terms used still suffer from local meanings within the community, and uneven changes in these meanings. The IEEE-CS, AIS and ACM defined five computing disciplines: computer engineering, computer science, information systems, information technology, and software engineering. These terms and many others are used as the names of educational programmes. Across the world, the same name may be used for quite different programmes and different names for similar programmes. This makes it difficult for potential students, employers and educators to determine the nature of a particular programme and how it compares to others [4]. However, in each of the identifiable social contexts, the meaning of the terms is shared, and represents a social construction of meaning.

B. Social Construction

Computing is global endeavour, with a mix of topics and degree/programme names. Even the term 'computing' as an overall term is selected for the title of this paper, as it appears to be the least value-laden term and understood most universally in the greatest number of contexts [4]. Examining these differences formulates a taxonomy, but when rooted in so many different national and social contexts, each phrase means different things in different places. Each usage, with its meaning, is valid, because it is meaningful to the people who use it in the context they use it in.

The process by which individuals produce candidate claims for knowledge, and these candidates are endorsed by the community using agreed upon standards [14]. This aspect of knowledge generation is critical to understanding both the development of, and differences in meanings about a discipline and its constituent parts. This includes a historical context; the meaning of terms – words go in and out of use; meanings expand and contract. Computing, and more particularly our shared understanding of computing is not only a socially constructed meaning, but also a collection of core meanings that have been developed over time in three intertwined traditions.

III. Three Traditions of Computing

The 1989 *Task Force on the Core of Computer Science* headed by Peter J. Denning characterized the discipline of computing as a combination of three separate but tightly intertwined aspects: theory, abstraction (modeling), and design [15]. Those aspects rely on three different intellectual traditions (the task force called them paradigms): the mathematical (or analytical, theoretical, or formalist) tradition, the scientific (or empirical) tradition, and the engineering (or technological) tradition [16].

A naïve conclusion would be that the core of computing necessarily includes mathematical analysis, scientific empiricism and engineering methodologies [7]. However this is problematic: the mathematical, scientific and engineering traditions each have different aims and methods, entailing different epistemological, ontological, and methodological assumptions. For instance, the logico-mathematical tradition and the engineering tradition entail different ontological views about their subjects of study, the empirical tradition and the theoretical tradition employ different methodologies, and the engineering tradition and the empirical tradition have different views about the epistemological status of their research results [16]. For example, debates about the value of qualitative methods vs. quantitative methods and where these fit into computing remain. The following sections briefly outline the three traditions:

A. Mathematics and Computing

At the turn of the 1970s C.A.R. Hoare argued that computers are mathematical machines, computer programs are mathematical expressions, programming languages are mathematical theories, and programming is a mathematical activity [16]. It has been argued that mathematics is the quintessential knowledge and skill for a computing professional, and that the discipline of computing is nothing but a paradigm change in mathematics. Denning et al. (1989) wrote "Theory is the bedrock of the mathematical sciences: applied mathematicians share the notion that science advances only on a foundation of sound mathematics." [15]

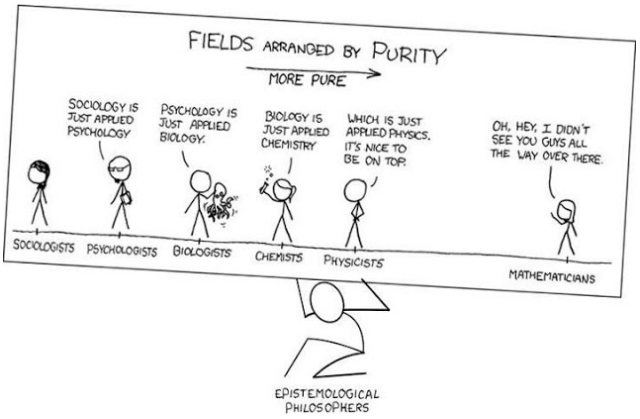

Figure 1. Problems in Epistemology: Fields arranged by purity [17]

This notion of computing as just applied mathematics is more than just an epistemological problem (see Figure 1). The difficulty is that computing is not mathematics, but rooted in it. While mathematics and mathematical abstraction and problem solving provide foundations for computing, this viewpoint ignores the automation of abstractions, e.g., that computing also develops its own science, both in method and object and is not strictly mathematical.

Similarly, one can observe that in computing there is a 'problem with problems.' very purity of mathematical expression and problem solving can be argued to be fundamentally problematic in developing the contextual and creative skills necessary for effective engineering practice expected of most computing practitioners [18]. The observation that there are 'problems with problems' can be viewed as both an epistemological problem between the math and engineering traditions of computing, but for students and practitioners it also appears as another difficult contextualization problem.

B. Science of Computing

The aim of science is to describe, organize, explain, and predict phenomena in the world. There is general agreement that these 'scientific' efforts split across two classes of phenomena: Mind-dependent and mind-independent phenomena [19]. *Mind-dependent* sciences primarily involve peoples' understandings and beliefs and aim to describe, explain and predict phenomena whose existence depends on people's states of mind. *Mind-independent* sciences primarily involve natural phenomena and study phenomena independent of people's states of mind. Computing science takes on both sides of these understandings [3].

This issue of the 'science' of computing is further complicated when exploring the 'what' of algorithms at the heart of computing: *Information*. When stepping past the formalism of algorithms is the information to which the algorithm is applied. The 'what' of algorithms quickly expands to other types of knowing: Information-as-*knowledge*, -as-*process* and -as-*thing*. And the object of the *Information-as–thing* may not have anything to do with computing. In the context of information as knowledge, there is the constant issue of knowing *that*, knowing *how*, with the additional distinction of knowing *about* [20].

However, other aspects of computing do not fit into the scientific framework well; the phenomena studied are not *naturally* occurring – rather they are applications of mathematical formalisms, or a 'formal' science, and the 'natural' environment of computing is an engineered environment which applies scientific empiricism differently [16,3]. One aspect of "how" in computing relates to algorithms, the prime formal entity that codify how to accomplish some goal. These are fundamentally mathematical and abstract [21]. But computing is about topics extend well beyond the mind-independent: building things. It is necessarily contextual: about the study of computation on real machines for real purposes, and necessarily includes the study of the mind-dependent side: People's preferences, attitudes, programming languages, values, worth, standards, processes and procedures [19,21]. But engineering is the prime discipline that is concerned with how to do things, how to build things.

C. Engineering and Computing

The engineering roots of computing are significant, but also different from other engineering sub-disciplines rooted in natural science [22,13]. Some of these roots are in the development of the technologies of computing while others are for the development of the applications of computing, most notably with the information and processes that support businesses [3,4,20]. Yet another engineering application of computing engineering is to the development of computing technologies themselves [22]. These different dimensions blur the relationship of engineering to understanding both the core and boundaries of computing as a discipline.

With significantly fewer limits to what can be created, computing differs significantly from other fields of engineering [7]. Historically, 'formalists' among engineers emphasise the mathematical nature of software and advocate mathematical methods such as theorem-proving and model-checking. 'Engineers' emphasise the constructivity of computing system development while 'humanists' emphasise the social interactions during the process of socio-technical systems development, as well as the implications of computing applications in the wider society [5]. Engineering contributes both to the further development of the 'natural' (mind-independent) environment of computing and to its mind-dependent applications: in business, science and society.

The engineering tradition of computing seeks to build and advance the methods tools used in creating those applications. But this too blurs the core and boundaries of computing:

Engineering of algorithms, tools and methods for advancing the science of computing, such as compiler technology, VLSI CAD or operating systems contribute not just applications, but also extend the science, and occasionally even the mathematics of computing.

These more recent observations on the relationship of mathematics, science and engineering to computing, provide additional insight into the core of computing, particularly in the edge cases among the traditions: Mathematical formalisms extending the *-knowledge*, *-process* and *-thing* of information could easily be properly considered computing, but apply or extend disciplines outside of computing. Computing work, like other engineering traditions extends itself;

IV. Establishing Boundaries for Computing

The history of computing established its early goals to be significant: The mechanization of human thinking. While acknowledging computing's roots in philosophy and mathematics [6,15], more recent advancements toward these goals draw from psychology and neuroscience [2]. While notable and relevant, this work still leaves significant questions about the nature of computing [3,6].

For example, computer engineering studies microprocessors and sensors. Computer science studies algorithms and computability theory. Software engineering studies things like software development processes and quality processes. Information technology studies things like network design and lifecycle analysis. Information systems studies things like organizational processes and decision support systems [12]. Many computing fields, such as human-computer interaction and artificial intelligence, span across several computing fields and also intertwine with fields other than computing [3,7,2].

The observation is that if Computing, as a discipline historically has its roots in the 'automation of human thinking', the observation is that this is fundamentally an epistemological question rather than an ontological one. This is particularly appropriate for a discipline whose foundational roots are in *denkzeug*, or the mechanization of knowing. Consequently this work lays out an epistemological approach to defining the core of computing, and distinguishing its core from its periphery at least for educational purposes.

A. Epistemological Approach to Computing

In philosophy, epistemology focuses on human knowing, most specifically on the formulation of justified true belief [14,13]. Epistemology recognizes the process of knowing, which is both an action and a process, involving both covert and overt human action and involve the application of personal and social criteria [13]. Knowing has not just a subject, *e.g.,* the person doing the knowing. It also has an object, that is, what is being known. In the general sense, this is about the knowing of all thing involving truth. But in computing, the focus is on the mechanization of the knowing and the known.

For this approach, we utilize three framing questions to help distinguish and explore the boundaries of computing: *how* knowing occurs, and *what* the knower knows, which we extend by also analyzing *why* the knower wants to know/act [14,23].

V. Boundaries for Computing Disciplines…

While the goals of the computing discipline are broad, and focus significantly on the process(es) of mechanizing human knowing, they also include the *object* of mechanizing knowing: the data and information being mechanically 'known' [20]. Addressing both the 'what' and 'how' aspects of the mechanization of knowing, the philosophy of computer science deals predominantly with problems and questions around the nature of computation as a process in time, the physicality or non-physicality of information, or with the question whether or not computer science belongs to the group of mathematical or natural sciences [5,10,21]. These topics all include both the *how* of mechanizing knowing and the *what* of mechanizing knowing. But to a certain degree, neither these developments in philosophy of computer science and historical developments in the communal understanding of computing significantly explore the *why* of 'mechanizing knowing'.

The thesis of this work is that it is the combination of the why, how and what of 'mechanizing knowing' that can serve as a broader, extensible model for identifying what are the boundaries of computing, and consequently its core. The assumption here is that there is a system that is supporting the what of mechanized knowing.

A. 'What' of Mechanized Knowing

The need to distinguish data, information and knowledge are common to our age of information workers [24]. In these models, *knowledge* is viewed as that which is known by a person from the information available. But knowing is both action and process, and within the bounds of computation, the 'what' is properly extend. Computational systems not only present information for knowledge by humans, but also are acted upon by these systems themselves in serving the needs for which a computing system is built.

But this need, identifies very clearly the what of computation: Data, and its recorded components, information, and the actions of the system that transform the knowing of the data and information into action. These are at the heart of information systems and information science studies. The actions, and study of the actions are at the heart of software and computer engineering studies, and normatively included in computer science studies. The means of recording and transmitting information is both its own science [20,3], but also has its historical roots in electrical engineering, physics and mathematics. Cognisant of these roots, the processes for collecting data and converting data to information are properly studied in many different aspects of computing. These techniques and methods range from the very human (e.g., business process, HCI), to the very physical (encoding, sampling and sensors), and ultimately to the goal of any such system, human knowing [12]. In the broad sense, this entails the meaning and value that the system presenting the information and/or acting upon it produces for humans interacting with (e.g., reasoning support) or affected by the system.

The object of mechanized knowing is reasonably viewed as within the core of computing from both socially-constructed and philosophical points-of-view [3,21,20]. Consequently the boundaries of the what are at its boundaries:

the means of collecting data recorded as information; the means of encoding it for transmission and its transmission; the physicality of its storage and most especially how meaning is established from the information presented and system actions. In our model, distinguishing reasoning support (information presented) from system actions plays an important role in distinguishing computing from traditional engineering [13,14]. Where the object of computing, e.g. the data or correct thinking and reasoning is not about computing *per se*, then this area of computing is necessarily on the boundary, not the core of computing.

B. 'How' of Mechanized Knowing

The mechanization of knowing has consistently been the heart of the computational endeavour [6,21,3]. But within the 'how' of computing comes various parts. At one 'edge' of this domain, are the mechanisms of computational knowing – such as psychology, neuroscience and philosophy, and the resulting mathematical and computing abstractions and theories that these have generated [21,6]. One of the difficulties in expressing the edges of computing as a discipline is understanding that this discipline is constantly expanding in its depth of knowledge and creation of new computing artefacts, methods and processes [7,11]. Like other engineering disciplines [16,23], computing has a proper science [3], whose purpose is the expansion of the tools and techniques of computing, only its historical/foundational roots are in mathematics and epistemology rather than mathematics and natural science. More recently, psychology and neuroscience have been added to these foundations. At its depths, this dimension has a boundary as well, as physical computational structures remain on the boundary as well.

C. 'Why' of Mechanized Knowing

Computing has, even prior to being recognized as a proper discipline [6], always served purposes outside of itself: a 'why.' Computing includes the automating of reasoning, e.g. the 'how', not only of the means of computing, but rather the computing artefact(s) delivered for a particular computing purpose. It is in this sense that computing is significantly like other engineering disciplines [23,22].

In this sense, computing shares aspects of other purposeful, creative technical activities: serving human need [18]. As such it also includes the creative processes both for the development of new computational methods and artefacts [13,22]. In this sense, the boundaries of computing are found in manners similar to those of other engineering disciplines: The edges of the discipline are the edges of the application domain(s) to which computing is applied, that is, its purposeful use. Consequently, distinguishing the science of computing from its application is similar to distinguishing the science of computing from its use in engineering [22,13].

Assuming computing occurs and is studied in the context of an abstract system, these three {What |Why |How} dimensions are illustrated in the Disciplinary Boundary Model (DPM) presented in Figure 2. The 'what' dimension refers to the objects of computing, that is, upon what information computing occurs, *e.g.,* its inputs, and how these are received, managed, structured internally, and map to their human/external system use.

The 'what' of computing necessarily involves a transformation of data into information into/supporting the development of human meaning. While these processes are proper and internal to the discipline, they also have significant boundaries: All data and meanings are on the edge of computing, both within the discipline but also on the edge. For an electronic system, the science and methods proper to the input/output signals external to the computing system would belong to that discipline (*e.g.,* electrical engineering), but also share a boundary (*e.g.,* digital logic), which then is managed within the context of both computing and electrical engineering. At another end of the abstraction chain would be user interfaces. The development of these would be proper, but also their impact and effectiveness on the user external to the system would be proper to that domain (e.g., psychology, psycho-motor, etc.) while also sharing a boundary (e.g., user experience). The mathematics and other abstractions proper to these transformations would also be 'shared'.

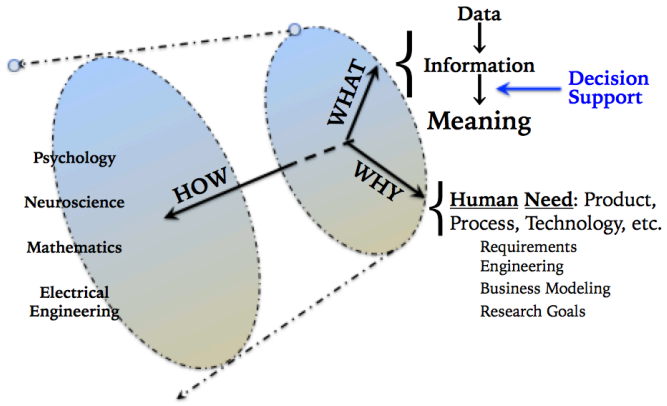

Figure 2. Disciplinary Boundary Model of Computing

Of particular interest in the 'what' of computing is the area related to the development of meaning from information, e.g., decision support and other related fields. This is a particularly interesting disciplinary boundary, as it draws back to the historical goals of the discipline, the German *denkzeug* and the mechanization of human thought. This is where the goal of the computing output is to integrate into human thinking external to the computing system itself. Consequently disciplines like "decision science" while rooted in psychology, would also necessarily share a boundary with computing, as computing aims both to automate and support human knowing.

The support for development of meaning sets up the second proposed dimension: 'why' compute. Figure 2 depicts this dimension of this disciplinary boundary, which is where the inputs and outputs of the computing work are important – non-computing disciplinary domains supported by computing. Like other engineering products, this in the form of tools, methods, abstractions and other means by which computation supports that particular disciplinary domain. For example, where computing is applied to chemistry, *e.g.,* computational chemistry, computing is being used by chemistry; the combined methods, techniques, models processes and tools developed are neither completely proper to computing nor to chemistry. In this view, the algorithms, user-experience, development of the software supporting computational chemistry is proper to computing, while its use in education

and practice is proper to chemistry, and research into the tools and the impact on education and practice proper to both. These boundary areas are more recently labelled "CS+X" [1], and present an ever-growing boundary, as computing developments in any of these areas expand the corpus of computing, but also the disciplinary domain being supported. In general, these what/why boundaries of Figure 2 are governed by the needs of the discipline or specific application.

This same argument, when applied to computing itself yields a different answer, because, the engineering aspects of computing are proper to the discipline. The results of the engineering development are proper to the computing discipline, yet distinguished from its science. Hence the development of compiler technology is both a scientific and an engineering endeavour that expands the depths of the 'how' of computing, rather than the border of a related discipline. This boundary is internal to computing – as Software Engineering vs. Computer Science are both part of Computing, with distinctions from each other yet intertwined in history and tradition [13].

VI. An Emerging Landscape

During the last decade, computing has taken a new, more empirically driven path with the maturing of machine learning, the emergence of data science, and the "big data" revolution. This is an excellent example of illustrating the boundaries computing landscape, as it draws on both the depth of the 'how' of computing, yet also crosses the 'what' and 'why' boundaries of Figure 2. It has a core in computer science, but leveraging methods beyond computing, data science should evolve in breadth to address the needs of domains outside computer science [25]. As the ability to solve computing problems has grown, the landscape has also grown. For example as the problems grow to be challenged by scale of applications, engineering approaches are needed. For hardware, the natural choice is to use the techniques from the electronics discipline.

The ability to handle challenges of scale has enabled cross fertilization with a somewhat distinct computing discipline: Information Systems. Human history presents numerous examples of a continuously development in the ability to identify, store, manipulate and generate information [20]. Computing has added, and will continue to add new technologies, tools and methods to store, manipulate and generate information useful to humans. Since information is ubiquitous almost every discipline that uses information can now be affected by computing. The consequent impact is large and has brought forth issues like security of information, its ownership, and fuzzy legal boundaries. Increased capabilities with this expansion of the field has also led to enormous data that has now challenged the processing needs across disciplines, and not limited to traditional information intensive disciplines either! Today we see Data Science (and Engineering) as an integral part of this landscape of computing. This has been particularly expanded by the artificial intelligence community that has applied its techniques to the data explosion resulting in disciplines like machine learning and deep learning. The impact has been gargantuan spanning across traditionally distinct disciplines.

One key point of this paper is that the high rate at which the landscape of computing has expanded to influence other disciplines has not been supported by a similar rate of amalgamating the what-why-how between computing and other disciplines. The current approaches appear to be mainly pairwise ostensive between computing and a given discipline under consideration. Consider computer games and data science. Game theory, while proper to mathematics contributes to the core knowledge base for computer gaming. From data science, more extensive reliance on statistics adds to its core. In all of the emerging disciplines, their mind-dependent aspects Equally contribute to the core. These could come from the science proper to their domains, such as maximising reward points as a motivator for gaming, or warnings about cognitive bias for data science equally contribute to the core. Similarly techniques would expand computing from the related disciplines' engineering traditions. Computer gaming adds to the corpus of interaction design principles (from HCI). From data science might add principles of data collection. At present, these developments appear to be on the shared boundaries of computing and these related disciplines, and how critical these techniques are outside their sub-discipline could easily vary.

A. Implications for Computing Curricula…

Traditionally, most descriptions of technical curricula have used terms like "science", "technology" and "engineering". The intended meanings have been assumed to be self evident. Most descriptions have defined these terms within their domain of discourse. A natural consequence has been the proliferation of terms to describe the same idea or concept or artifact, by a different community in a different context. Given the urgency of identifying the core of computing towards designing computing curricula, we offer a framework in this section. We redefine some terms towards that goal. The key idea of the model is to focus on the human activities common to disciplines, at least the ones we believe are influenced by computing.

1) A Layered Framework Model

Our observation of the world around us results in creation of layered concepts about the technical world.

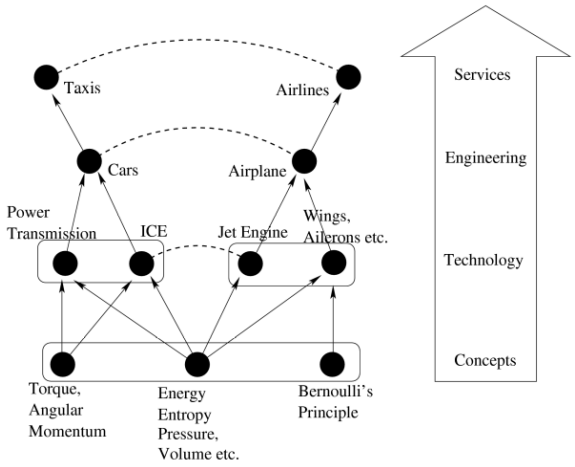

Figure 3. Conceptual 'Layered' Framework applied to Mechanical Engineering

Figure 3 illustrates the intuition of our model for mechanical engineering. In this figure, the bottom nodes allude to a continuous process of amalgamating scientific concepts into a set of coherent and consistent ideas. For example Physics has created concepts like energy, entropy, torque, angular momentum, pressure, volume etc.. When pressure and volume are manipulated we observe gases behaving in a pattern that suggests the idea of a heat engine. We suggest using the word "technology" to allude to the process of realising ideas from the cerebral world into the physical world. Thus, the idea of a heat engine can be realised in practice in a number of ways, each giving us a "technology". One technology is the internal combustion engine (ICE in Figure 3), and another is the jet engine. Similarly the knowledge of torque and angular momentum is realised in practice as the technology of mechanical power transmission system that transfers the power generated by the heat engine to the wheels of an automobile. Each technology has its own capabilities and limitations.

However, a technology by itself may not be very useful in the external world. A number of technologies need to be brought together to create useful artifacts. Given that each technology used in composing the artifact has its own capabilities and limitations, the process of bringing them together often involves balancing a variety of constraints of the constituent technologies. These technologies, developed by the discipline and within the discipline expand the corpus of the discipline.

It is also possible that new technology may be required to improve other technologies or methods proper to the discipline. The differential in an automobile is one such technology. We will refer to this process of bringing together technologies to realise a useful artifact as "engineering". Thus the power transmission system and the ICE are technologies used to realise an automobile. Similarly, improvements in jet engines and wings are the technologies that are added to, and deepened the core of mechanical engineering [26].

Engineering, like computing embraces the value of the objects it creates for society. When the engineering of a given artifact is good enough, they are pressed into service using a variety of business models. If economies of scale form one engineering constraint, then a service based on scale of operations can be built using the engineered artifacts. Taxi services use automobiles and airline services use airplanes. Economies of scale reflect in the cost with low scales usually having higher costs of individual artifacts, and high scales are usually cheaper. Contrast a custom luxury car with a mass produced car, or a private jet with an airliner.

We limit our framework to these four levels of human activity: science, technology, engineering and services. Considerations of economy of scale are explicit at the last two levels, and implicitly influence the first two. We observe that cerebral aspects like imagination, creativity and elegance exist at each level but are only challenged differently. Figure 3 captures these four in an arrow from the mental to the physical. It is crucial to note that this is simply a framework to describe human activity and does not reflect actual practice. In practice each of these levels are constantly interacting with and influencing each other. In particular, the figure should not be interpreted to suggest some sort of hierarchy. Quite the contrary, we opine that all the four levels are equally important, interdependent and mutually constructive, and that each level is also evolving alongside the others.

2) Illustrating the Layered Framework

The concepts-technology-engineering-services framework can be used to describe multiple levels of abstraction. Figure 4 illustrates this framework for the C programming language system, and could be easily extended for other programming language systems. At the concept level are concepts finite state machines and push down machines from computability, lattice theory and graph theory from mathematics. At the technology level, mechanisms of control flow and data flow analysis exemplify the abstract "technology" where the conceptual knowledge is reflected in practice via algorithms. Language design elements are formed out of a need to engineer the language for some intended purpose. The C compiler system chooses some rules defined via standards, some rules defined by the need to interact with other software components in the tool chain, and some rules (*e.g.,* a collection of optimizations at a given optimization level that is visible to the programmer) through field experience. The engineered C system is pressed into service through libraries that support specific domains like networking or graphics. These services are visible as C callable APIs.

Figure 4 may be used to identify the knowledge units for programming languages. A curriculum designer can then choose to focus on the conceptual and technological aspects of the language if the intent is foundational, or focus on the API level details if the design goal leans towards practice.

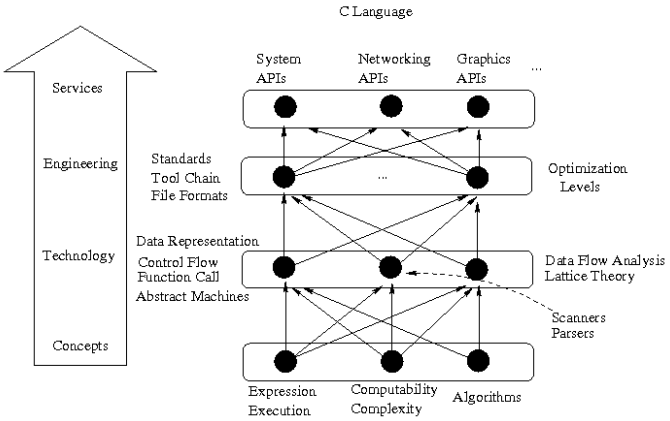

Figure 4. Conceptual Framework applied to Computing

The conceptual and technological levels are applicable across languages. Similarly the techniques of engineering a language may be discussed using C for illustration. The proposed model could also be more effective if we also assume that (a) such instances for specific components of a discipline can be created as in Figure 4, and (b) cross fertilization across disciplines typically occurs first at the services level of the interacting components of such disciplines and gradually filters down to the conceptual level. This needs more work, though.

VII. Impact on Computing Education

This work presents two models – the DPM for

distinguishing boundaries of the computing discipline (Figure 2), and a layered framework for examining the relationship among topics within the discipline (e.g., Figure 4).

The DPM suggests that computing is a discipline that will necessarily expand in both its breadth and depth, and that various related disciplines (e.g. psychology, mathematics, etc). will become more prevalent and inform computing as the boundaries expand. The intrinsic linkages implied also suggest that computing education needs to more seriously consider philosophy of computing as a more significant contributor to our understanding of computing as it expands [21,3]. Similarly, findings related to core computing learner competencies [13,27] and valuing educational research might play a more significant role in shaping curriculum development [28].

Leveraging the DPM, the core of computing makes itself visible and it's contribution apparent on its shared boundaries, especially in a CS+X context. This means for distinguishing among concepts, and patterns for distinguishing sub-disciplines are important, as errors in these distinctions could easily undermine the computing aspects of related disciplines. This could be of clear use both in the formulation of programs and curriculum development, and encourage cross-disciplinary education and joint educational research in CS+X disciplines.

The layered framework suggests a language for discussing dimensions within the core of computing, which could e applied at course and discipline levels to simplify the development of computing curricula.

The current versions of the ACM/IEEE curriculum guidelines for each discipline contains much identical content yet is taught from different perspectives often without reference to other disciplines. These two models strongly suggests that computing disciplines are intrinsically linked, and the distinctions between and among computing sub-disciplines would be subtle, and often difficult to distinguish. What is a contextual difference due to language may also be an issue of how the three traditions of mathematics, science and engineering are locally applied, or even due to sub-disciplinary culture. Consequently the impact of developing curricula guidelines for existing and emerging computing sub-disciplines takes on a more significant role and curricular integration is paramount [26].

VIII. Conclusions and Next Steps

This work has developed a scholarly-informed case for the issues in defining landscape of computing education, and used this to propose a framework for describing the disciplinary boundaries of computing. Starting with a foundation in the philosophy of computing, it establishes an epistemological approach to support differentiating consensus about the core and boundaries of computing. This framework is then applied to different aspects of computing sub- and related disciplines that vary internationally. This work will address how these issues of agreement among various communities fit, and how they affect our understanding of computing, defining its core, and examining computing disciplines in the future.

This framework approach suggests an expansion in the locus for computing curriculum development from that which currently exists: Computer Science, Information Systems, Information Technology, Computer Engineering, Software Engineering, CyberSecurity. This locus should expand to computing areas yet to be strictly defined: Data Science, Data Engineering, Computer Gaming, etc. These could and should have computing curricula developed with the expectation of newer computing disciplines to emerge.

This framework approach also suggests a new look at the mechanisms and approach for developing computing curricula. Contextualization and community issues significantly affect curriculum development, and that the intrinsic linkage among disciplines that work suggests the need to look at each computing discipline not just independent of each other, but in the context of each other.

References

[1] National Academies of Sciences, Engineering, and Medicine, "Assessing and Responding to the Growth of Computer Science Undergraduate Enrollments," Committee on the Growth of Computer Science Undergraduate Enrollments, The National Academies of Sciences, Engineering and Medicine, Washington, DC, 2018.

[2] Brad Smith and Harry Shum, "Foreward," in *The Future Computed: Artificial Intelligence and its Role in Society*.: Microsoft Press, 2018, p. 77.

[3] Matti Teddre, *The Science of Computing*. New York: CRC Press / Taylor & Francis, 2016.

[4] Simon et al., "What's in a Name? International Interpretations of Computing Education Terminology," in *Proceedings of the 2015 ITiCSE on Working Group Reports*, Vilnius, Lithuania, 2015, pp. 173-186.

[5] Stefan Gruner, "Problems for a Philosophy of Software Engineering," *Minds & Machines*, p. 25, February 2011.

[6] Martin Davis, *The Universal Computer: The Road from Leibniz to Turing*. New York: A K Peters/CRC Press, 2011, ISBN 9781466505209.

[7] Alfred Z. Spector. (2017, August) Changing Nature of Computer Science and Its Impact on Undergraduate Education. [Online]. http://sites.nationalacademies.org/cs/groups/cstbsite/documents/webpage/cstb_173998.pdf

[8] D. Gries and F. Schneider, *A Logical Approach to Discrete Mathematics*. Berlin: Springer-Verlag, 1993, Footnote on page 36.

[9] Jan H. Kroeze, "Ontology Goes Postmodern in ICT," in *Proceedings of the 2010 South African Institute of Computer Scientists and Information Technologists*, Bela Bela, SA, 2010, pp. 153-159.

[10] Raymond Turner and Nicola Angius. (2017, Spring) Stanford Encyclopedia of Philosophy. [Online]. https://stanford.library.sydney.edu.au/entries/computer-science/

[11] Lillian Cassel, Robert H. Sloan, Gordon Davies, Heikki Topi, and Andrew McGetterick, "The

computing ontology project: the computing education application," in *Proceedings of the 38th SIGCSE technical symposium on Computer science* , Covington, KT, 2007, pp. 519-520.

[12] Joint Task Force for Computing Curricula 2005, "Computing Curricula 2005, The Overview Report," The Association for Computing Machinery (ACM), The Association for Information Systems (AIS), The Computer Society (IEEE-CS), Computing Curricula Series 2005.

[13] Stephen Frezza, David Nordquest, Richard Moody, and Krishnakishore Pilla, "Applying a knowledge-generation epistemological approach to computer science and software engineering education," in *Proceedings of the 120th ASEE Conference and Exposition*, Atlanta, 2013.

[14] Joseph C. Pitt, "What Engineers Know.," *Techné: Research in Philosophy and Technology,* vol. 5, no. 3, Fall 2007.

[15] P.J. Denning et al., "Computing as a Discipline," *Communications of the ACM*, vol. 32, no. 1, pp. 9-23, 1989.

[16] Matti Tedre and Erkki Sutinen, "Three traditions of computing: what educators should know," *Computer Science Education*, vol. 8, no. 3, pp. 153-170, September 2008.

[17] xkcd: A webcomic of romance, sarcasm, math, and language. [Online]. https://xkcd.com/435/

[18] Catherine Goodman and Stephen Frezza, "Finding Möjligheter: Creativity and Ill-Structured Problems," in *Proceedings of the 2017 American Society for Engineering Education (ASEE) Conference and Exposition*, Columbus, OH, 2017.

[19] Matti Tedre and John Pajunen, "An Easy Approach to Epistemology and Ontology in Computing Theses," in *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*, Koli, FInland, 2013, pp. 97-104.

[20] Michael Buckland, "What Kind of Science Can Information Science Be?," *Journal of the American Society for Information Science and Technology*, vol. 62, no. 1, pp. 1-7, 2012.

[21] William J. Rappaport, *Philosophy of Computer Science*. Buffalo, NY, USA, 2016, Unpublished - Private Notes 2004-2016, intended as a book.

[22] Matti Tedre, "Computing as Engineering," *Journal of Universal Computer Science*, vol. 8, pp. 1642-1658, April 2009.

[23] Stephen Frezza, David Nordquest, and Richard Moodey, "Knowledge-generation epistemology and the foundations of engineering," in *Proceedings of the 2013 Frontiers in Education Conference*, 2013, pp. 818-824.

[24] N Ingebrigtsen. The Differences Between Data, Information and Knowledge. [Online]. http://www.infogineering.net/data-information-knowledge.htm

[25] Francine Berman et al., "Realizing the Potential of Data Science," vol. 61, no. 4, pp. 67-72, April 2018.

[26] Michael Goldwebber et al., "Historical perspectives on the computing curriculum," *SIGCUE Outlook*, vol. 25, no. 4, pp. 94-111, October 1997.

[27] Curricula Task Group on Information Technology, "Information Technology CUrricula 2017," Association for Computing Machinery (ACM)/IEEE Computer Society (IEEE-CS), Computing CUrricula Series DOI: 10.1145/3173161, 2017.

[28] Tony Clear, "Valuing Computer Science Education Research?," in *Proceedings of the 6th Baltic Sea Conference on Computing Education Research: Koli Calling 2006*, Uppsala, Sweden, 2006, pp. 8-18.