

# The Many Faces of Science: A Look at Computer and Life Science

For sharing in the common term of "sciences," Biology and Computer Science certainly are quite different disciplines. And yet, they exist for and operate in many of the same ways, when it comes down to it. This paper explores a little bit of what the two share between them, and where they diverge, by approaching them from three main standpoints: Method of Procedure, Accumulative Nature, and Precision Importance. Seeing as how all members involved can rightly claim for themselves the title of "scientist," those who are particular to Computer Science will now be affectionately addressed as "geeks," while those partial to Biology will be just as affectionately deemed "nerds."

## Method of Procedure

Method of Procedure may be perhaps the most striking contrast between geeks and nerds. First, it is necessary to be specific about what is meant by Method of Procedure, especially since nerds in Biology (and Chemistry and Physics and Astronomy, etc.) hold to a rather specific method of scientific procedure; in many cases this is not relevant to geeks in Computer Science. For example, a nerd may find a particular question he or she wants answered based on some phenomenon or observation from nature. Take for such a case the change of apples from green to red as they ripen. The nerd would develop a hypothesis, set up and conduct an experiment, collect the data and develop an appropriate conclusion based on the findings. This system works very well because there is much still about nature and natural occurrences that is not known. For geeks, however, the procedure typically goes the opposite direction. It still all begins with a problem to be solved, but the geek is typically constructing rather than explaining. For example, a geek designing a

program to monitor the electric current put out by a particular dam in New York may run into a problem; due to space restrictions on the monitoring devices, only so much memory is available for the computer to work with. The geek must now develop a program that which complete the desired task without using a large amount of memory. This is quite a different sort of problem than that of figuring out why things happen as they do. Much of this springs from the nature of the two different disciplines themselves. Whereas Biology has developed over the years as a method of determining events that already go on, Computer Science is a relatively new human creation that exists not so much to explain as to create: effective machines, data storage, computational programs, etc. Some of the reasons for the existence of both these disciplines are the same (and will be discussed in the last section, *Progressive Goal*), but they serve rather different purposes. It is much as though Biology and nerds work from the outside in, while Computer Science and the geeks work from the inside out. That is, a nerd works for discovering more and learning more, while a geek will use a set of assumptions and work for creating something anew. This is evident in the way the two groups report their work. Generally, a nerd will publish an extensive in-depth and detailed report of what he did and found; a geek makes known from her work only what the others absolutely have to know to build off of it. Such a terse report may frustrate (or be flat out rejected by) a nerd. Likewise, a nerd's excruciating concern for detail and care of *how* he acquired such and such would probably drive a geek loony. This is why a Biology project will often include both a generalized review drawn from many different studies, as well a very precise report on a particular experiment. This is why a Computer Science project will generally contain everything one needs to know about a program in the program itself, with in-line documentation (author's notes interspersed in the code) and a brief descriptive paragraph at the head of the program.

### **Accumulative Nature**

This last point of difference in the *Method of Procedure* leads right into the concept of the *Accumulative Nature* of the sciences, or the "group effort." While nerds and geeks may not hold to similar levels of information exchange, they both agree on its utmost importance. No Biology experiment is of much use unless it can be shared with others to expand on, critique, and use for research. No Computer Science program is of any good

unless it can be built off, modified, and applied to new problems. Similarly, it is absurd to think of any nerd or geek "going solo" and doing it on his own. There is no experiment that does not draw on previous scientific knowledge, and no program that does not build off of what has been coded in the past. This is what begets the term "Accumulative Nature."

These two sciences (and one may argue, all sciences) are really giant projects made up of the contributions of millions and millions of smaller discoveries and works done by people throughout the ages. Later, the *Progressive Goal* section will discuss how indeed even the various areas of science themselves can be thought of smaller studies and efforts all contributing together. This by no means takes anything away from individual contributions - for the high cannot stand without the low. One can imagine how much of a mess it might create if one nerd were to misrepresent some data here or there and end up spawning a whole slew of subsequent experiments by other nerds that were based off of that original misleading fact. This is perhaps even more obvious for geeks. When writing a program, a geek will include code and program fragments made by others, always assuming the code functions as it should and returns the claimed result. For example, one might borrow code snippet for alphabetically sorting a bunch of names; a geek does not actually go through this code to make sure it works - she goes off the assumption that the last geek did his work well. Unfortunately, it does happen that documentation is 'misrepresented,' and this can lead to many hours of fruitless debugging for the unfortunate geek. Luckily enough for both nerds and geeks, there already exists a pretty reliable system of peer review and project critique. This strengthens the Accumulative Nature of the sciences, as works will often have to pass through extensive testing and review before they can be widely accepted in that grand library of "accumulated knowledge."

### **Precision Importance**

Notable above is a difference in the necessity of precision. No doubt an inaccuracy in either field will disrupt results, but things can be stretched to differing extents for the two. For example, there was a time when it was believed that maggots spontaneously spawned on rotting meat, and this (presumably with other similar examples) led to the idea of spontaneous generation. This is now known to be false, as it was disproved in the latter half of the 19<sup>th</sup> century, but it is very likely that many useful and necessary

experiments were still carried out during this time despite (or even because of) this misdirected belief. There are few, if any, similarities among the geeks - granted, of course, that Computer Science is still a very young science. This is just one aspect of what is here called Precision Importance. Biology has (and must have) less of a dependence on precision in the big scheme of knowledge acquisition than does Computer Science - must have, because of its core difference in Method of Procedure. That is, Computer Science builds off of things that people have created and agreed upon to be true. Biology, on the other hand, realizes that there is a very big world of unsolved mysteries and unexplainable to deal with. The nerds must then either give up altogether, or else proffer up various different theories, accepting that they may well be proved false in times to come. Thankfully, the current system would rather use these theories despite their possible fallacy than to give up altogether.

Stepping down into one further level of precision one encounters the case of actual numbers. Here, Biology definitely wants and needs precise numbers (say, for measurements); yet the extent of them may not be as extreme as those demanded by Computer Science. Nerds will often utilize the technique of "significant digits," that is, measuring to the point of importance and dropping the rest. No doubt this is occasionally done by geeks as well, yet not so often. Geeks will actually spend quite a bit of their time figuring out just how precise their numbers are. This 'obsession' with exactness in numbers has many visible symptoms in Computer Science. When there are a million different ways of carrying out the same operation, speed and space become the determining characteristics as to which method will be used. In fact, a very large amount of time and effort is spent by geeks in calculating and comparing just how precise and fast and bulky their creations are. There are many many applications of this, but one good example is the infamous Big-O (infamous, perhaps, only to the poor college student who must learn to calculate it in a very many different ways). This is a method of determining and comparing the speed and 'order' of a particular algorithm in a program. To give a specific example, some geek may desire to sort the entire set of works in the Library of Congress based on the number of letters in each work. This geek would remember from her college days that there are quite a few different sorting methods (Bubble, Insertion, Selection, Bucket, Shell, Heap, Merge, and Quick, to name just a few). The various sorting methods fall under different Big-O

categories, which correspond roughly to how many times the particular algorithm would have to be run to reach the desired result. While this is not necessarily a direct determinant of the fastest method, one program is likely to be much more inefficient if it must run an exponentially higher number of times than another. Starting from this decision, the geek can then account for the variable of available memory (i.e. Could a computer remember the size of all the works while comparing them all at the same time?) and determine which algorithm best suits her particular needs. In this case, precision could mean the difference between a number of hours of execution, and a number of years! So obviously, a geek does not want to forget to take precision into account. This is not to say in the least that a nerd would, however. Most good scientific reports on Biology experiments will either explicitly say or give the reader a good idea of the estimated percentage of error for that particular experiment. It is the case with most things in nature that there will very rarely be anything with a zero percent error. Nerds will often assume there is error - in fact, there are quite a few charts and tables for estimating this. Here is an example of one from class:

<b>Sample Size</b>	25	50	100	250	500	1000	1500
<b>Percent Size</b>	22.00%	14.00%	10.00%	6.00%	4.00%	3.00%	2.00%

Geeks will also acknowledge imperfection in their work, but, true to their character, they want to know *precisely* how far off it is. Thus, there are many geeks totally devoted to the field of determining overall efficiency (something that is not unknown to nerds). Because Computer Science is very much a created science, there is much about it that is assumed to be 100% correct. Indeed, a program that had even a 99% accuracy could still be considered rather flawed and perhaps even unusable, depending on the situation. Especially when dealing with any kind of safety and/or human interaction with machines, anything less than 100% assurance is unfit. Robots are an excellent example of this: the robotic arms that work alongside humans to construct automobiles, for example. There is so much power and strength demanded from these machines that if there were known to be even a 1% inaccuracy in the coding of the software that controlled a machine, it would be deemed unfit to work anywhere near humans. For these applications, and the above-mentioned applications of speed and efficiency of algorithms, geeks repeatedly measure all of the

programs they create. Most likely any program that one might bump into on a home or commercial computer these days had to prove itself better than a slew of similar programs by being filtered through a variety of different types of precision tests. It is easy to see why Computer Science has a very high level of precision importance.

Nerds and geeks may disagree on much, but ultimately (one would hope) they are all striving towards the same end. As mentioned earlier, no individual science experiment or computer program is a secluded work - and likewise no individual group of study is a secluded journey. Both Computer Science and Biology are made up of a conglomeration of smaller parts, and ultimately are both working towards improving human knowledge in the hope of bettering life. Noted, there are abuses of these fields of study (hacking for geeks, biological weapons for nerds, and viruses for both, but overall they exist for honorable ends. When so many things seem so strikingly different in comparing these two disciplines (and others, for sure), it is reassuring to note that they indeed exist as partners to serve the same goal. So whenever the discrepancies of procedure arise, whenever the details do not match up, whenever the group work just does not seem to fit, one can be assured that, in the end, nerds and geeks will be able to set their differences aside and unite with each other, for the good of their studies and of their fellow humans by using their respective skills of microscopes and flash drives, Bunsen burners and pocket protectors - fighting, arm in arm, in suspenders and glasses, for that better world. Lord help us on that day. =)