Looking Back – Looking Forward

Curt Hill

Mathematics and Computer Science Department

Valley City State University

Valley City, ND 58072

Curt.Hill@vcsu.edu

Abstract

It is impossible to predict the future! However, what does the past suggest about the future? In particular what does the history of computing and computing education imply about our own future? This paper will attempt to illuminate some of the possible paths that are possible in the future.

This paper and presentation makes a call to flexibility and adaptability. It will be characterized more by the asking of questions, challenging thinking and suggesting possibilities than by the stating of results.

Introduction.

Some years ago at a previous university I took a Greek language course on Plato from the Classics department. The textbook for this course was published in 1897, so I was able to photocopy the needed portions without violating copyright. In contrast to this situation is the Computer Science book that is still read after ten years, which must be considered a classic.

If we learn anything from our past it is that change is the constant factor of our life in a technological discipline. We cannot look to the past to predict the future, but we can observe the types of changes that have occurred to help to get ready for those yet to happen.

Advancement.

The first principle that is clear is that technology will continue to advance and this advancement will be driven by economics. As long as the nation and world possess a robust economy, accelerating change will be the normal case. The technological fact of life is that technology will continue to advance. There have been centuries where the life of a person's grandfather was not significantly different than the life of the person's grandson. This is no longer the case. The computing environment changes somewhat on an annual basis. The environment of a college freshman has changed by the time that student enters graduate school or the marketplace.

Consider the progress of our machines. The first hobby computer was the Altair in 1976. It cost about $500 and used the second of Intel's 8 bit processors, the 8080. This configuration came with no I/O, other than the front panel switches and an amazing main memory of 256 bytes, that is 0.25K. (Today, $500 buys a low end laptop.) Five years after the Altair, the first IBM PC made a tremendous impact on the business community. This machine was sophisticated for its time, comprising an important business machine. The appearance of the first IBM PC was itself triggered by large sales of Apple's initial models, especially the very popular Apple II. In those five years most of the measures of the computer had increased by a factor of 10 or more, except the cost. Each of these three machines made the previous one look like a toy. Since that time successive models have continued to rush their predecessors towards the landfill. An educational institution cannot afford to let it be known that it possesses five year old computers.

Economics has favored advancement. The producers have a need to offer, faster yet less expensive processors to maintain their sales levels, while the consumers have continued to want more power at less cost. In theory, we should someday arrive at a machine that has more capability than the ordinary person may utilize, but in practice that has not occurred yet. Instead, capabilities, such as rendering video for the production of a DVD, that were beyond the reach of all but a few are now available to nearly all.

Government and business research spending may vary between pure and applied, but that research will affect the computing environment as well as society in general. This change is inevitable and educators will adapt or be left behind. Only time can tell what is truly fundamental and what is merely the fashion of the time.

We may surmise then, that technology will continue to advance. What we cannot know is where the advancements will lead. It has been said for some time that silicon circuit technology progress must slow. Advances in miniaturization cannot move past certain atomic boundaries. An initial sign of this was the introduction of pipelining techniques. This was an acknowledgement that increased speed could not be found merely by making things smaller. A more recent sign of this was the introduction of multiple core processors [Hill, 2005]. This was a means to increase computing power without increasing heat production. It is estimated that in about twenty years we will be near the end of line for making silicon transistors any smaller.

The end of shrinking silicon circuits does not mean the end of the advancement of technology. Indeed a plethora of new technologies are waiting in the wings to usurp silicon's dominance. The hard part is determining which ones will make the kind of impact silicon integration technology has made. Some of these include alternative materials such as gallium arsenide[Kloeppel, 2006], quantum computing [Shreve, 2006], photonic computing [Greene, 2007] and biological computing [Mearian, 2007].

Some of these technologies will have different impacts other than increased performance. The use of gallium arsenide or other material may will not likely change the approach of anyone not employed by the manufacturer of the device. Outside of the factory, nobody will care how they increased the speed, reduced the heat or whatever improvements are accomplished. Other changes will have a pronounced effect. The software development community is just now realizing that the multi-core processors are showing how poorly parallelism is handled compared with sequential techniques [Bier, 2007].

Another issue that will affect the next technology is the maturity of the current. When the transistor replaced the tube, its inherent advantages quickly displaced the older technology. Well before integrating multiple transistors on a chip, transistors were clearly superior to the tube technology they replaced. However, most of the new technologies are not so clearly superior that they will displace silicon quickly. King Silicon is well entrenched and some of these technologies may be abandoned before they get a significant fraction of the research that silicon has received.

Constraints and Vision.

Technology constrains us. It constrains what we can do and sometimes even what we dare to imagine. When the constraints change it takes a while for the vision to expand into the new frontier. People in general and educators in particular often subconsciously assume that the environment at a certain time of their lives will continue forever. Nothing could be farther from the truth.

For example when the first “portable” computers were introduced they featured a small CRT which dominated the size of the unit. These were ungainly, usually weighing in at more than 20 pounds and with computing power limited much like the rest of the personal computers of their day. They were a niche market at best. Who knew then that this was the future? All that was required was the development of lightweight display technology and increase in speed and storage. Today, the desktop computer is in danger and the laptop, perhaps with accessories such as an external display, is changing how we view computing platforms. The even smaller devices are now moving towards wresting much of the laptops market share.

The previously mentioned early hobby and personal computing machines had an unusual feature. They were usually turned off – every day. Prior to this, a running computer was a 24/7 type of endeavor. They were only turned off for maintenance or repair. After the advent of the internet, this unused computing power spawned the concept of grid computing. Instead of building supercomputers, we merely need to exploit the unused computing power available in almost any organization. However, there was a considerable time gap between the capability being available and someone realizing that it could be utilized. Such opportunities will arise again and again as our technological capacity increases.

This concept of unexplored new capability is particularly apparent within education. We have this tremendous power in the computer, but how can we utilize it to educate students in a way superior to that of Socrates? Online and distance education enthusiasts have seen the use of the connected computer as an easy way to reach students unable to attend conventional classes. We have also seen several ways to enhance traditional classroom setting as well. What we have not seen is the “killer app” of education. This is the software that is so superior that we wonder why we did not think of it before.

In the recent past the use of threads was limited on most applications. A task with long running time might have one thread maintain the user interface and another to execute the task at hand. This was quite rare in the classes of the undergraduate curriculum that did not specifically deal with parallelism. With the advent of multi-core CPUs this will become very common, even a design goal.

For example consider the well understood process of compiler construction. A typical compiler would often consist of four phases: a lexical analyzer, a parser, a code generator and an optimizer. These made for nice logical subdivisions of the whole product. On a machine with very little memory they may be four separate programs that passed files between them, but those times have mostly passed. However, in the multi-core environment these should be four separate threads that could be parceled out to separate CPUs. This is a bad idea on a single processor, for the thread overhead and synchronization issues will lengthen the run time. Yet the single processor time is also passing and we need to change our thinking as our constraints change.

Differentiation.

The maturation of the industry is providing more differentiation and more specialization. The trend is clear in hardware. The CPU was the first intelligent device. Since then we have embedded intelligence in many of the other support devices such as I/O processors and graphics processors. This intelligence now includes controllers for a host of devices that are not usually considered computers. We see similar trends in programming languages and other areas and this trend will continue.

The trend in programming languages seems to be towards interpreted and scripting languages rather than general purpose languages. The big three of C/C++/Java continues to dominate the general purpose programming languages [TIOBE, 2007] by survey of popularity. However, following these is a host of relatively new interpreted languages: PHP, Visual BASIC, Perl, Python and JavaScript. This seems to indicate that the large scale software development will continue for the foreseeable future, but there will also be considerable effort devoted to smaller, tactical programs. The growth of CPU speed has made the advantages of the highly-optimized general purpose languages significantly less important than in the past. In an era when computing power is very inexpensive the importance of efficient code is insignificant, while the programmer productivity issues are monumental. This is not a new trend – it has been in play for decades. This is exactly what justified the first compilers.

This specialization has been a good thing. Instead of a one size fits all mentality there is a programming language for a specific class of programs. What the education sector needs to do is find the languages that make the most sense in their program. The current low enrollment in Computer Science could use a little programming in Alice at the high school level [Hill, 2006]. Unlike most of the previously mentioned languages, very few people will hold developer positions using Alice. Instead Alice may be the model of the part time language. It is an easy to learn and easy to use language that can be utilized by people who are not mainly programmers. In the future we may have a small number of people whose principle job is software development and a much larger group of people whose job requires less than five hours of software development per week. This trend may mean that the Computer Science department will become like Mathematics, a service department. Today we see Mathematics departments serving very large numbers of students in a few courses, but with relatively few majors. How will this paradigm change affect our thinking in regards to our course offerings?

Operating Systems.

The interface of a user to an operating system is another good example of change. The evolution from batch with a control language to interactive with a console has not finished. The current Windows, Icons, Menu and Pointer (WIMP) interface is the current king, but history argues that the WIMP interface's days are numbered.

Each stage of the process has generally moved from a harder to use towards easier to use interface. This greater ease of use has been accomplished with greater complexity of the operating system. This has been made possible by the increase of power and decrease of cost of computing. The mainframes of the 1960s did not have the computational power of the desktops of today, so even if the operating system consumes the same percentage of the machines capabilities the functionality it provides should steadily increase. The Windows NT family has steadily grown in size from approximately 6 million lines of code in NT 3.1 to 50 million in Vista[Wikipedia, 2007].

We only see two or three major steps in the interface. The batch operating systems usually used a control language with its own particular syntax. Since the control language and data often occurred in the same card deck, it was preferred to have a unique syntax for the control language. When the move to timesharing occurred this syntax was generally simplified with the data coming from files. However, the jump between control language of a batch system and timesharing commands was not large. The major shift was from these to the Graphical User Interface. The team at Xerox PARC led by Alan Kay devised this new interface. Although it has evolved since then, the current examples are clearly recognizable as refinements of the original.

Another example of change in operating systems are the services that they provide. The prevailing trend is incorporate as services the functionality that is used by a set of common applications. Once some threshold percent of applications implement a certain functionality the operating system will typically bundle a program that performs that function or integrate it into the system as a service. An early and controversial example is the free distribution of Internet Explorer with versions of Windows. A later example is support for zip files in Windows XP and Vista. Some would decry these additions as monopolistic tactics or operating system bloat. Of course, Microsoft did not start this particular trend, it has been present in operating system development since the beginning.

What is next? The impossibility of predicting the future has already been stated, so I dare not give any prediction as definite. It seems clear that the trend to incorporate features based on marketing and frequency of use must continue, as must the trend for ease of use. The Microsoft Vista interface seems to moving towards an easier text to speech and speech recognition interface as well as glitzier three-dimensional representations. Presumably this line of development will continue to evolve. However, the major paradigm shift from command-line to GUI was made by just a handful of people. We should not discount the possibility of an even better interface appearing. We should also recall that even after the GUI was clearly dominant, there were many still teaching exclusively with the command-line interface.

The three major operating systems used on desktops include Windows, MacOS and Linux. The market forces have tended to make these look more and more similar to one another. However, as our machines diversify we should also expect the operating systems on them to diversify. One would not think of putting any of version of Windows (or the other two for that matter) on the controllers in an automobile. As the world fills with intelligent devices and internet appliances the custom operating system should become important. Is this an area that the curriculum has addressed?

Consider one other possibility. The One Laptop Per Child project has moved away from rotating magnetic memories to flash memory for secondary storage. This has been done to minimize cost, improve durability and to reduce power consumption. Suppose that truly disk size capacities could be stored in flash or other solid state device. If this were the case then the differences between primary and secondary storage would be greatly reduced. The paradigm could shift our whole view of disk. Imagine that the secondary storage is the site for all execution. The operating system does not load the executable into memory, it merely executes it directly from the slower memory. This would probably include a larger cache for this slower storage, but the split cache has been common for some time. The very existence of large caches indicates that designers are already thinking of memory as a slow device. If our executable memory now encompasses what we now think of as disk, how will we organize this structure?

Conclusion.

The point of this essay is that there is no conclusion. We will continue to be assailed by a rapid stream of changes to our technology, which will require continued adjustment to our courses. The treadmill will not stop and we cannot get off. Instead, we must see each new advancement as an opportunity to make our students better prepared for a world that will change more in their lifetimes than in ours.

References.

Bier, Jeff (2007). Multithreading invites string of nasty bugs. EE Times. http://www.eetimes.com/op/showArticle.jhtml?articleID=196802368. Date accessed 2 March 2007.

Greene, Kate (2007). Intel Speeds up Silicon Photonics. Technology Review. http://www.technologyreview.com/Infotech/18087/ Date accessed 7 March 2007.

Hill, Curt (2006). The Importance of Alice. MICS 2006. http://www.micsymposium.org/mics_2006/papers/Hill.pdf. Date accessed 9 March 2007.

Hill, Curt (2005). The Challenges of Parallelism. MICS 2005. http://www.micsymposium.org/mics_2005/papers/paper43.pdf. Date accessed 9 March 2007.

Jones, George (2007). The Top Five Technologies You Need to Know About in '07. http://www.computerworld.com/action/article.do ?command =viewArticleBasic&articleId=9011969&pageNumber=2 Date accessed 2 March 2007.

Kloppel, James E (2006). World's fastest transistor approaches goal of terahertz device. http://www.news.uiuc.edu/news/06/index.html#December. Date accessed 5 March 2007.

Mearian, Lucas (2007). Scientists: Data-storing bacteria could last thousands of years. http://www.computerworld.com/action/article.do?command=viewArticleBasic&taxonomyId=19&articleId=9011945&intsrc=hm_topic Date accessed 5 March 2007.

Merritt, Rick (2006). Multicore faces a long road. http://www.eetimes.com/news/semi/showArticle.jhtml?articleID=196701586. Date accessed 2 March 2007.

One Laptop Per Child Project (2007). Specs. http://laptop.org/laptop/hardware/specs.shtml. Date accessed 2 March 2007.

Shreve, Jenn (2006). Computing in a Post-Silicon World. http://www.citris-uc.org/newsletter/december_06_newsletter Date accessed 2 March 2007.

TIOBE Software. TIOBE Programming Community Index for March 2007. http://www.tiobe.com/tpci.htm Date accessed 7 March 2007.

Weiss, Rick (2007). Putting the Brakes on Light Speed. http://www.washingtonpost.com/wp-dyn/content/article/2007/01/18/AR2007011801683.html. Date accessed 7 March 2007.

Wikipedia, (2007). Source lines of code. http://en.wikipedia.org/wiki/Source_lines_of_code Date accessed 8 March 2007.