Daniel Campos Prof. Carothers

CSCI-2500 Computer Organization December 7th, 2012

Soul of a new Machine Extra Credit Assignment

If at this present moment you were to go visit any company that is working with computers and you stood back to observe one thing would quickly become apparent to you. There is always a fight between software and hardware. Whether it is in the startup producing application or it is Apple Inc. both have to find a mid-point and compromise between software and hardware. For Apple, they continually fight to have a faster processor to run the old OS when a new OS comes in and makes the processors outdated. In the startup they must fight to give all the app functionality they want without breaking their platform or destroying battery life. This need for compromise has been something that has been part of the computer industry from the start, and although the companies may rise or fail, new types of products come and go, the computing industry as a whole and as an extracted concept, stays the same.

The first thing that I noticed while reading was ever since IBM introduced the concept of backwards compatible software and applications and our protagonists struggled to create it, every single computer related device from Operating Systems to Videogame consoles has tried to keep doing it to this day. In the book it is said “They announced, in the mid-sixties, all at one time, an entire family of new computers – the famous 360 line. In the commerce of computers no single event had a wider significance, except for the invention of the transistor. Part of the 360’s importance lay in the fact that all the machines in the family were software compatible.” (Page 42). Presently we can look at the most recent major Operating System release, Windows 8. Though this OS was a complete rewrite of what used to be and really was a new from scratch concept the OS is completely backwards compatible in applications that Ran on Windows 7, Vista and XP. Besides just being compatible many of these applications have even been optimized since it is understood that it is very important for people to keep on using their old programs on their new machines. IBM through that move made sure that the computing industry would always be able to have interchangeable parts.

The first part of this hardware vs. software fight that comes to mind is the continued fight between the programmers desire to make development easier while the hardware developers want more of the work to be carried by the software so they don’t overburden the hardware. In the book, it is stated : Holland and his troops would make their changes and the Hardy Boys would look at UINST and say, ‘There’s no way we can do this function and this function in hardware.’ The two sides would argue and work out those problems. Then the microteam would discover something else that was hard to do in microcode, and deciding that it should be done in hardware, they would insert in the next issue of UINST this wish for a change. The Hardy Boys were on guard. They’d scan UINST carefully, looking for new microteam mischief, and finding this new item, declare, ‘No way we can do this in hardware.’ And it was back to bargaining again.” (Page 159). Back then the software boys really did not like coding complex functions in microcode and to avoid doing it they would try to build it into a hardware function. Presently, similar things can be seen with the switch to multi core processors. These new processors have multiple cores and therefore need to have the information divided amongst all of them. The software developers do not want to code the low level style code necessary to deal with information distribution and processing distribution and try to get the hardware makes to do these functions. The hardware boys are already overwhelmed by the ridiculous needs of these processor and try to make the division of labor be a software thing. The fight also comes since less people, time and money are usually used when dealing with the creation of these kind of things and issues from a software standpoint vs a hardware standpoint. Software is also faster to implement but will end up not running as fast as the hardware and the software also will interfere with the other processes while the hardware wont. In the end it’s a conversation that has equal amount of valid points on each side and because of that it is so hard to decide.

A separate similarity that one may draw from the different scenarios is the commodification of the hardware present in computers, the decrease in its cost and the increase in its availably. Where the book states “By the mid-1960’s, a trend that would become increasingly pronounced was already apparent: while the expense of building a computer’s hardware was steadily declining, the cost of creating user and system software was rising.” (Page 42). This trend has continued but in a different format. It has continued more in the market of the mobile smartphones (that for all intents and purposes are full blown computers). Their prices have been dropping and dropping since it is better understood how to develop the chips in the way that the hardware needs and the huge amount of production (the same processor goes in every iPhone and since 73 million have been sold the production cost to produce these ARM processors has gone down tremendously). The software persei has not increased since most mobile phones have it free but the apps that they now use and the amount of apps available and time and money spent creating this application has grown exponentially creating this multi-billion dollar industry centered around these ‘App Stores’.

Another part of the computing industry that has stayed constant is a company’s willingness to make a bet no matter how large or well-established of a brand it is. Unlike many other industries where the older players tend to be very conservative since they are well established, the volatile nature of the computer industry. Back then, the setting has IBM giving a vast bet on their 360 computing line and if it didn’t work well they would go bankrupt they said “‘We’re betting the company’ one IBM executive remarked. Indeed the project cost somewhat more than the development of the atom bomb, but it paid off handsomely.” (Page 11). Other companies, like our protagonist, Data General, stakes everything they owned on the Eagle computer. It can be considered nothing more than a gable since they had a sole chip supplier and that supplier had many issues” “sole supplier ( might be about to declare bankruptcy…They could not get all the PALs they wanted. Never mind getting the thousands they would need to mass produce Eagles; they scarcely had enough to repair errors in the prototypes. Rasala took to keeping a list of how many PALs they had on hand. For months, they teetered on the verge of running out. ‘We could lose the whole thing right there’ said West.” (Page 231). This Rasala is the RPI graduate student that you asked us to find and say what he did. This trend to always keep betting and transforming is still present today. They are many cases where the betting turns out well (Apple with the iPhone) and they are even more cases where large corporations who seemed to control the whole market well made a few wrong steps and because that are doomed to a slow destruction (Blackberry). This is something that is also being seen now more than ever with the prevalence of the startup culture. “The technology of computers changes constantly; every year it’s a struggle to keep up with the youngsters fresh out of school.” (Page 104) is a perfect example because unless companies are smart and work well then they are soon overtaken and since it’s not like people need a PHD to create the next big thing(aka Facebook) these big companies have to continue to bet big and fast.

Another area where we have continued in the old fashion is in our desire to anthropomorphize computing. Back at data general “Two computers that the Eclipse Group used, the engineers had given the names Woodstock and Trixie, after characters in comic strips. They often spoke about these computers as if they had personalities”( Page 90). Now we continue this fascination with trying to add a human aspect to things but have taken it to a whole new level. There are things like cleverbot and Siri, both different stabs at trying to make things seems like a fully cognoscente AI. We are now, just as we were then, not happy with merely having supper fast computers that are able to do huge calculations. We want to add a human face to these computers, and have them act as human as possible towards us, when in reality they are coded to answer in a certain way.

Another such area of similarity that I have learned through this class is the usefulness of knowing assembly language and basic languages of the computer. Back then, the books aid “It is commonplace today for programmers to stick exclusively to high level languages and never look inside their machines”( Page 98). This is especially true in this day and age of crazy programming languages that do everything for you and never force you to look for what’s beneath the hood (Python, etc.). Even with C and C++ people never really want to understand what is happening at the assembly range since it is complex and hard to change or understand. Though it is as so Asling is correct when he talks about “learning assembly language during his time of midnight programming. ‘It was neat to learn it. I could skip the middleman and talk right to the machine. It was also great for me to learn that priestly language. I could talk to God, just like IBM ’ ”(Page 98). I found the exact same when I learned assembly and with a lot of higher level workers I have been told by them that they feel much the same. When they finally learn to write and understand assembly they can understand their program better and many times they can write it better.

A separate area that has remained quite constant in the computing world would be the way people work in it and how the structure of the company many times works, informally. “Castro liked to see a little competition stirred up among teams. Let them compete with their ideas for new products, and bad ideas, as well as negative points of good ones, are likely to get identified inside the company and not in the marketplace”(Page 110). While this is something that may be applied to many companies, I feel this has been kept especially true in the computing world. Companies like Google, Microsoft and Apple are constantly placing their groups against each other both for critique and defense of product. That way they are able to see all the issue that their products may have.

A final point of similarity is the common place of litigation and legal action that companies take to each-other. In the 1970s and 1980s “IBM virtually resided there (in court). Everyone sued IBM, it seemed. The biggest suit, the Jarndyce v. Jarndyce of the industry, involved the Justice Department’s attempt to break up IBM Virtually an entire large law firm was created to defend IBM in this case which by 1980 had run ten years and been in continuous trial for several.” (Page 15). This is true today with other giants like Apple and Samsung. No matter how many products are released and how different or similar they are it is an immediate step for both parties to try to sue each-other and get the other product banned, imposed a tariff on, or any other practice. This keeps many of the companies in court non-stop for the last 10 years.

In conclusion, throughout the many faces of the computing business it may seem like a lot has changed or has nothing to do with the past but when a good look is actually given into the industry and how it actually functions and works we find that A lot more has stayed the same, merely evolved, instead of changing. This is especially with true with the cycle of Giants being toppled by tiny competitors and then those competitors becoming giants and being toppled. Whether it is IBM losing it monopoly or Microsoft or Google, unlike most industries, size does not guarantee longevity.