Much like other fields, software engineering changes over time. What was common practice in the early 1950’s is different than common practice today. With a field so heavily dependent on new technologies, it is clear that if we do not change our focuses from decade to decade, we are not using the newest, latest and greatest technologies to our advantage. In Barry Boehm’s article “A View of 20th and 21st Century Software Engineering” the evolution of the software engineering process is covered, as well as predictions for what will happen to the field in the future.

Beginning the article is a summary of software engineering from the 1950’s, where Boehm states that people were told to “engineer software like you engineer hardware.” Today, this is far from common practice, as the hardware has become so well developed that it is no longer a limiting factor in our software. However, in the mid 1900’s, hardware was the critical component in the software engineering field.

As Boehm’s article moved on to discussing the field during the 1960’s, we see that people have realized that hardware is much more difficult to change than software, and began realizing that it is much more beneficial to engineer software which could be changed, versus having to change the hardware. We see that software took off at a much greater pace than hardware, and the primary focus shifted towards code, versus having engineers work primarily on the hardware. Code was being written using a “code and fix approach,” which is more of what we see in industry today. We see that the 1960’s was a great time for software engineering, as it really took off by having colleges/universities offer CS, the introduction of COBOL and Fortran, as well as for-profit software development companies.

After the large boom in software during the 1960’s, the 1970’s, according to Boehm, focused on various processes of software engineering, for example, the waterfall process. It is becoming evident
that by the 1970’s software was going to dictate the bulk of the production cost, and companies began to find ways to make the process much more streamlined. With this came the waterfall process of software engineering, in which people were responsible for having their own section of code, and passing it down the line until reaching a final product.

According to Boehm, the 1980’s were primarily about productivity and scalability. With software development driving the overall cost of a project (no longer the hardware) people needed to find a way in which they could write software to be more general purpose, and reuse most of the code for various assignments. A focus was given to having productive software, not just software which was written to do one specific task. The 1980’s also brought higher-level programming languages, which allowed software developers the opportunity to code up more versatile software.

By the 1990’s, software engineering had come from hardware limitations to high-level programming languages such as C/C++ and Java, object oriented programming, and a Unified Modeling Language (UML). These newer technologies and design processes really allowed software engineers to develop robust, powerful, and versatile programs which can do a multitude of things. Another focus during this decade was more rapid development, meaning that a goal was to decrease time-to-market. Another great advancement was the idea of concurrent programming. Software was also being written focusing more on GUI than ever before, as more and more people began using the computer, and were no longer using text based operating systems, with the release of more visual OS. Open source also began to take off.

The 2000’s brought more changes to the field of software engineering, such as the agile method of software engineering, and a focus on value based software engineering. Time-to-market was still a very important issue, and agile development was a great way to improve the time-to-market, as people
worked in groups to more rapidly develop a product. At this point, developers had to also deal with integrating older technologies, as well as dealing more with older legacy code.

As we stand now, software engineering has evolved a great deal over the last 50-60 years. Luckily, the limits are still far way, as technologies are constantly changing, and more ideas are being dealt with. We will be more focused now on dealing with having computers try to solve science issues, such as the newer fields like bioinformatics. Software developers now need to make sure that they stay within what is possible, as newer programs are beginning to push the current limits of what we can do. However, it is this drive to try to have technologies solve more issues that will ultimately drive us to develop even better technologies and programs, and allow us to have new issues solved, ones that could not have previously been solved without the aid of computers.