Submission to the VRC Middle School - Career Readiness Online Challenge

Coding Our Way To Reality
One Robot At A Time

Team Number: 47114A

Location: Great Neck, NY

Author: Chloe Ning

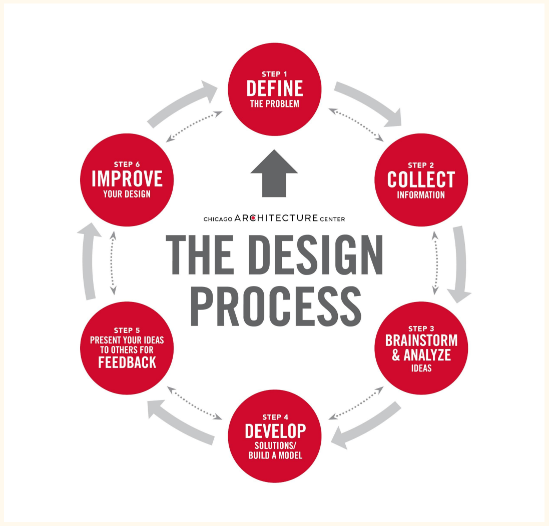
Software Engineering

As the programmer on my VRC team, the occupation of a software engineer has always been intriguing to me. I find it interesting how it combines the creativity involved in inventing solutions and the technical skill of computer science. In fact, “software engineering” was a term

Margaret Hamilton promoted while she was working on the Apollo Program to “acknowledge that the work should be taken just as seriously as other contributions toward the advancement of technology (2a).”

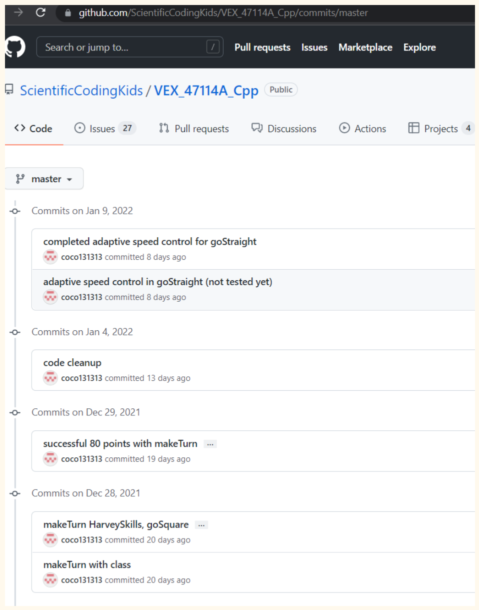
Software engineering revolves around solving problems by developing code, and it is very well paying (the national average is around $108, 249 (2b)) due to the challenging and critical nature of these problems. Becoming a software engineer requires a high level of education, usually a bachelor or master’s degree in software engineering or computer science.

The Design Process in Software Engineering



The design process is very prevalent in the world of software engineering because an object is being created—it may not be a physical object, like a robot, but the same rules and processes of creativity apply to it. Software developers have to ask questions to figure out what software needs to be created. The engineers then need to design a solution to the problem they have found, and after creating it, they need to test it. Then the cycle repeats until the solution is proven to be satisfactory.

The testing and debugging is one of the most important steps in engineering problem-solving. A successful software engineer has to be reactive to new problems and enjoy the problem-solving process. VRC can prepare people for this aspect of the occupation and I am an example of that. The years I have spent on a VEX team have exposed me to many unexpected problems, and it has made solving these problems second nature to me. A malfunctioning remote controller or laptop can usually be resolved by restarting. A battery cord that’s been chewed through by a particular pet rabbit is as good as new in five minutes with a candle and some electrical tape.



ScientificCodingKids / VEX_47114A_Cpp Public

Commits on Jan 9, 2022

completed adaptive speed control for goStraight
coco131313 committed 8 days ago

adaptive speed control in goStraight (not tested yet)
coco131313 committed 8 days ago

Commits on Jan 4, 2022

code cleanup
coco131313 committed 13 days ago

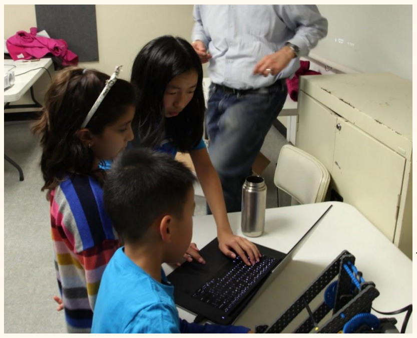
Commits on Dec 29, 2021

successful 80 points with makeTurn ...
coco131313 committed 19 days ago

Commits on Dec 28, 2021

makeTurn HarveySkills, goSquare ...
coco131313 committed 20 days ago

makeTurn with class
coco131313 committed 20 days ago

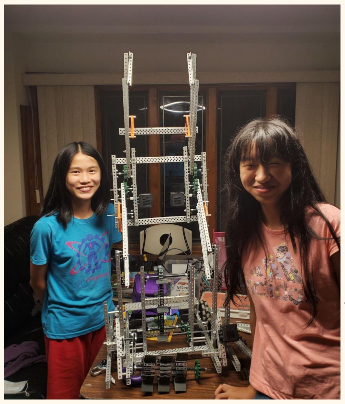


Despite the similarities in the design processes of a software engineer's and my team's, there is a different objective for the end result. For example, if an engineer is developing an app, then it needs to withstand all types of bugs or malfunctions for a longer period of time. Meanwhile, the purpose of a VRC team's work is to build and program a robot that needs to perform well for a specific competition. This gives me more freedom, and allows me to use "band-aid" solutions that software engineers aren't. For instance, if my robot is always turning 93 degrees when I ask it to turn 90 and time is running out, I might just change the instruction to turning 87 degrees in order to compensate. This is not equivalent to *fixing* the problem like a professional might be required to do, but it will serve my purpose.

How VRC Shapes *the Future of Learning Software Engineering*

A complaint students often have in school is *how will this help us in real life?* Before I got into VRC, I knew a smattering of Python and Java but I had no idea how to apply it. After five years in VEX, I've learned C++ and how to apply and code algorithms such as proportional control to ensure that our robot moves with precision during the autonomous period. I found myself applying certain skills that I never thought I would to a tangible thing: I figured out the length of the C-channel needed to support our four-bar using the Pythagorean Theorem, I used the circumference formula in my PID program, and the principle of potential energy to strategically place rubber bands that would help our robot lift bear the weight of mobile goals.

Once the context of what I was learning sank in, the information I was absorbing was more



interesting to me. This encouraged me to do more research on programming and the different ways code could impact the reality around us. This effect that VRC has can spread to more people, which spreads the love for coding that prerequisites becoming a software engineer.

In addition, VEX has the motto that "drive forward is the new Hello World (2d)." This further highlights how *context* is so important in making learning interesting to beginners—seeing your robot drive forwards shows you a tangible outcome from the abstract lines of code.

Credits

1. Image Credits:
    a. https://medium.com/@sarabadani.meysam/how-to-talk-to-a-software-developer-3f78a29d99dc
    b. https://discoverdesign.org/handbook
2. Sources:
    a. https://en.wikipedia.org/wiki/Software_engineering#Etymology_of_%22software_engineer%22
    b. https://www.glassdoor.com/Salaries/software-engineer-salary-SRCH_KO0,17.htm
    c. https://www.careerzone.ny.gov/views/careerzone/search/occupation/occupationProfile.jsf#.YeTMPS9Okgo
    d. https://twitter.com/VEXRobotics/status/1158853942849232896