Tyler Sulsenti

Driverless cars

Homework 01

We All learn to drive a car in our teenage years and continue to practice on a daily basis for most of the rest of our lives. Driving becomes second nature to us that each action, such as turning lanes feels natural. This problem, of switching lanes seems normal and easy to do when we so are accustomed to it, but what happens when we trust a computer to do this task? The main problem here is safety. The car must successfully switch lanes, without stopping and without hitting obstacles, other vehicles, pedestrians and any other road hazards. It also must be done in a reasonable amount of time for safety and efficiency reasons. The creation of the driverless car will be expensive in terms of labor and cost as we work to solve this problem.

We could use the waterfall method to approach this problem which would then require much planning and thinking about our possible solutions and methods. We must first know the requirements. I previously mentioned many requirements for this problem, the biggest one being safety. It is required that the car perform a safe and legal operation to avoid any injury or penalty. Other requirements will include the teams, hardware and various resources it will take to build working models of the driverless car. Next, we must design the actual model. The team must decide the best hardware and software for the job. Many different types of sensors and motors will be required for the operation. We must know these so that the software can be properly designed around them. After this design, we must implement the designs and put to use the gathered and identified resources. Prototypes will be created in this implementation. To verify the product, it must be tested. With safety being the biggest issue, the product will be tested off road someone on premise. These tests will validate and verify the production. The final step will be to release the product and keep it updated. The world is always changing, and software can always get better. Developers will always try to come up with newer and more efficient algorithms to keep the driverless car up to date as well as with any new traffic patterns or designs. Doing this will lead to a successful project.

This method calls for little to no failure at each and every step, which can be very well appreciated at the end of the process because it can avoid many catastrophic unforeseen errors in calculations or development that could cost money or produce an unsafe product. This is a huge advantage to this SDLC. Consequently, this approach requires a lot of time and investment at each step because it must be thoroughly checked for perfection, which is a tactically disadvantage for this SDLC. A large team would perform better in this situation as there are more checks and balances throughout the process.

We can also approach the driverless car situation in the Rational Unified Process SDLC. This process is not concrete. The team would use it to mold to their current needs as they initially analyze the problems of safety and efficiency. Using UMLs, the teams would envision their product and attack it Iteratively, in phases. Each phase allows the team to assess the situation and decide the best direction to go there forth. After the envisioning of the project is deemed stable, the team will start production to build a demo of the product in real life. Product verification will take place on this constructed demo. When the demo is deemed stable and safe and meets all of the requirements, it will be released and constantly supported to ensure the best quality, safety, efficiency and overall project success.

This approach is advantageous because it allows for a faster project development since less verification per step or phase, contrary to the waterfall method. However, this approach is disadvantageous as it poses a higher risk than that of the waterfall method because of the verification happening less. This risk is high especially in a project like this because safety is the number one concern. A mess up could cost lives which in turn could then cost the company millions.

In eXtreme programming, the team is going to focus on what’s the absolute easiest tasks to tackle first and get them out of the way. A certain team focuses on revenue and return will decide which tasks will provide the highest return and then the project will move forward as such.

These small fixes and tasks are advantageous because they provide a very close connection with the client. Customers and the team review the project and any issues are fixed in a bug release by the team. It is constantly monitored. But, while this is great for fast development, in our case with a driverless car switching lanes, we should focus on the whole project at once. Small, easy tasks leave room for error in the bigger picture especially if one piece fails. Safety is still the number one concern in this project and should not be compromised.