Jennifer Cafiero

SSW 555 Agile Methods for Software Development

Professor Rowland

30 August 2018

Agile Methodologies for *Driverless Cars*

After being hired as an Agile Methods Consultant by *Driverless Cars*, I am prepared to deliver my proposal about the pros and cons of a few Agile methodologies that can be employed in the development of a new software module that allows cars to autonomously change lanes when the driver toggles the turn signal. This new software will also need to be designed to successfully avoid objects in the road and pedestrians in order to prevent injury or death by integrating sensors and cameras in the vehicle. After in depth analysis of the different Software Development Life Cycle methods (SDLC), I have prepared a breakdown of the advantages and disadvantages of using the following three methods: the Waterfall Method, the Rational Unified Process, and eXtreme Programming.

The first agile methodology under consideration is the Waterfall Method. The Waterfall Method involves planning everything in sufficient detail so that it can be developed correctly the first time. This methodology is popular with traditional engineering and architectural problems such as building bridges and usually works best with a large team for a project of our size. One benefit of selecting the Waterfall Method is that it forces the team to dedicate sufficient enough time at the beginning of the project toward planning. Creating a solution for changing lanes autonomously involves a lot of planning, so it is good that the Waterfall Method would force the team to dedicate all of its efforts towards planning for a specific amount of time. However, in the Waterfall Method, the Implementation phase cannot begin until after the Design phase is completely finished. Because there are strict handoffs between stages in the Waterfall Method, the team cannot go back and redesign the plan if they realize half way through the Implementation phase that there is a major design flaw. Also, because this project is high-risk and not all of the requirements are explicitly defined, this would make the Waterfall Method a bad choice. There is no room for error in the design of our software module, otherwise accidents can happen and pedestrians might be hurt.

The second agile methodology under consideration for this software module is the Rational Unified Process (RUP). RUP was developed based on the six best practices of software engineering including, develop iteratively, manage requirements, use component-based architectures, model software visually, continuously verify software quality, and control changes. While developing software iteratively using RUP, the team focuses on the highest risk items in each iteration and accommodates for changes in requirements as they appear. This would be beneficial to our application because it will focus on the more important requirements like safety before moving on to less risky tasks. The RUP is best for larger projects with teams of over forty people or timelines that last longer than one year because the RUP depends on having multiple iterations of the development process. The amount of time and the cost of running a series of testing processes that this methodology requires might make it an unattractive solution for developing our lane changing software module.

The third agile method to consider employing is eXtreme Programming. eXtreme Programming depends on having much more close collaboration between customers and developers than traditional plan driven approaches typically have. This methodology also focuses on having many small releases of a product which wouldn’t really be beneficial to our application. Because of the nature of this project, it would be too risky to release individual components of our software. eXtreme Programming also focuses on collective ownership so that all the team members are at the very least acquainted with every part of the project and every team member feels obligated to make improvements to the project. In the eXtreme programming method, the customer is a member of the team, which is beneficial because their input is readily available throughout the development process so the final product tends to more closely match the customer’s expectations. This method also implements continuous integration which is the ideology that code should be integrated and tested every few hours so that the project is continuously tested and it is not put off until the very end with a deadline approaching. This would prove to be difficult for our team to integrate into a car and test multiple times a week. If the team is separated geographically, eXtreme programming also would not be a good solution to select for this application.