CISC 322

Assignment 3
Architectural Enhancement of Apollo
March 31, 2022

Zephyrus

Yucan Li 18yl259@queensu.ca

Yuzhe He 18yh46@queensu.ca

Xuchuan Mu 18xm24@queensu.ca

Yiming Zheng 19yz38@queensu.ca

Wenran Hou 18wh10@queensu.ca

Mukun Liu 19ml13@queensu.ca

Table of Contents

Abstract

After the previous two analyses about the Apollo autonomous driving system, our team has developed a deeper and comprehensive understanding of the overall Apollo structure in terms of conceptual and concrete aspects. By analyzing the concrete architecture of Apollo, we found out how data and state transmission within a publish-subscription architecture style including which function call is used to publish and receive data. In this report, we are going to introduce a brand new module called “mode control”. This module improves the efficiency of the vehicle through changing the vehicle’s state and helps the vehicle largely save their energy consumption and improve driving experience in different scenarios. Furthermore, two different implementations to this enhancement will be presented. A SAAM analysis will proceed so that the properties of the software architecture could be clearly recognized.

Introduction

In this report, our team is going to introduce a new module called “Mode Control”as our enhancement to the Apollo autonomous driving system. The main function of this module is to maximize energy efficiency or reduce the energy consumption (gas or electric power) by shifting gears or other operations that affect the travel of the vehicle. Professional drivers always know when to shift the gear to reduce the gas consumption. Another scenario is that people will drive faster if they are about to be late for work. So we try to make these "human driven" scenarios that could be automatically detected and taken operations by the apollo. This enhancement will utilize the Perception, Prediction module and make analysis on current road situations to shift between modes "Energy-Saving" mode or "Time-Saving" mode. Other alternatives to achieve this goal is that the user is able to use the HMI module to directly interact and operate on the Model control module instead of auto-adjusting state of the mode through autonomous detection by the Apollo.

We found this enhancement is well-motivated and not included in the original Apollo system. Confirmation messages from professor Bram also strengthen our confidence in presenting this new feature into Apollo. In addition, this new module needs information from several submodules inside Apollo to achieve its functionality and dependency.It depends on the CANBus module to acquire chassis information, then further vehicle surrounding information will be acquired from Prediction and Perception modules. Finally, this module will access the current Planning module and make adjustments on temporary planning.

We will also create two different user scenarios with sequence diagrams which helps construct and illustrate a full view of interaction between the Mode Control and other relative modules. Besides, the affection of this new enhancement on the functional and nonfunctional requirements of the Apollo system will be discussed in terms of its maintainability, evolvability, testability and performance. In addition, a SAAM analysis will be performed on both approaches. The major stakeholder will be identified and it will be conducted with previous system NFR in order to analyze how this new enhancement makes an impact on it. Finally a test will be planned and taken place after all these analyses in order to figure out if this enhancement is applicable and eligible. Last but not least, enhancement related risks will be discussed and analyzed at the end. Some potential risk which might affect the performance or efficiency could result in a deadly effect on the whole Apollo system.

Recap of Conceptual Architecture

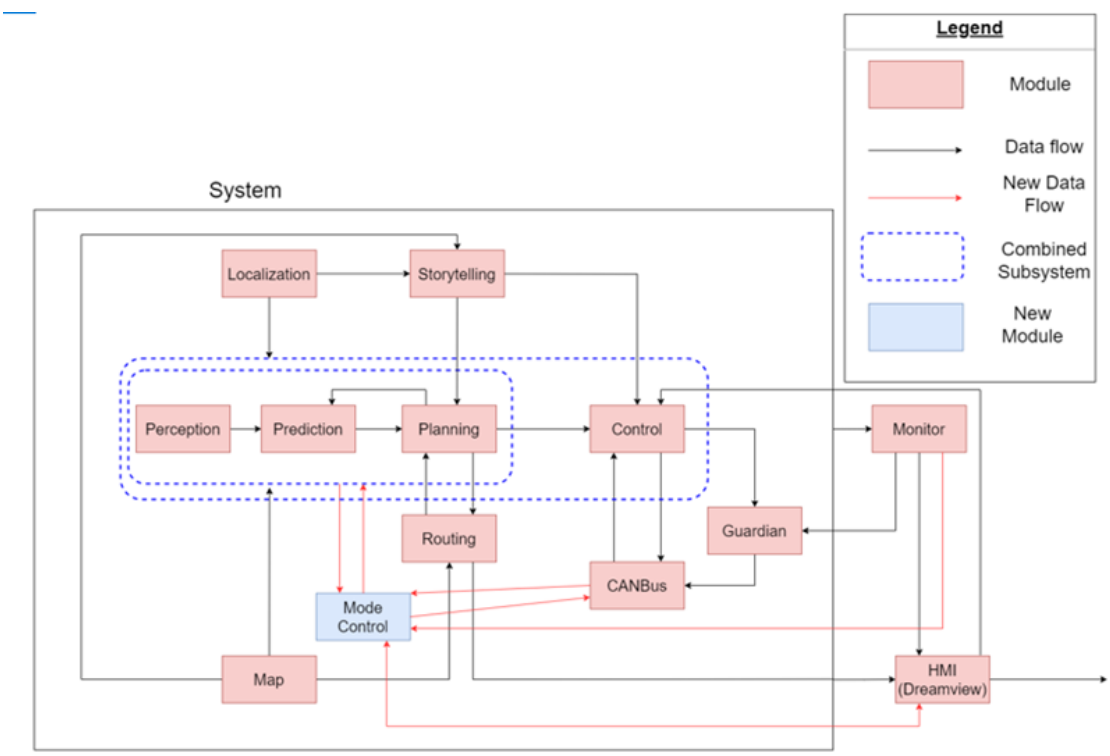

Figure 1: Modified Conceptual Architecture

In terms of the conceptual architecture above, the modules colored in orange are the original modules in the Apollo system. And the module in blue is the Mode Control module we introduced into this system. The concrete line with the arrow is the data flow among the system.

As we can see, the new mode is a complex module, because it contacts a series of modules. As we mentioned before, the car has two patterns (Energy-Saving mode and Time-Saving mode). For the Energy-Saving pattern, the mode module has a dependency relationship on HMI, Canbus, Monitor, and Planning modules. When the car is driving, mode switches the car to the Energy-Saving pattern by receiving the needed information. To be more specific, mode and HMI modules depend on each other by giving the request and providing the data. Planning has the similar communication with mode module as HMI. The Mode Control module gets the current plan of the car from planning by sending a request. It also depends on the monitor module because it needs the information of monitoring. For Canbus, we think it is very important, since it contains the key data that mode control mode has to use to calculate the mode it chooses. As for the Time-Saving mode, it has a dependency relationship with two additional modules which are prediction module and perception module, they are in charge of providing the obstacle and traffic light information for mode module.

Sequence Diagram

The enhancement we added aims to help users to deal with different scenarios of driving. For example, with the cost of fuel increasing and pollution we have today, using some driving techniques such as control of the acceleration and deceleration are able to save more energy; also, sometimes when users are in a hurry for some extremely important occasion, driving fast and safely becomes important. We add a module call mode control to help users deal with different situations.

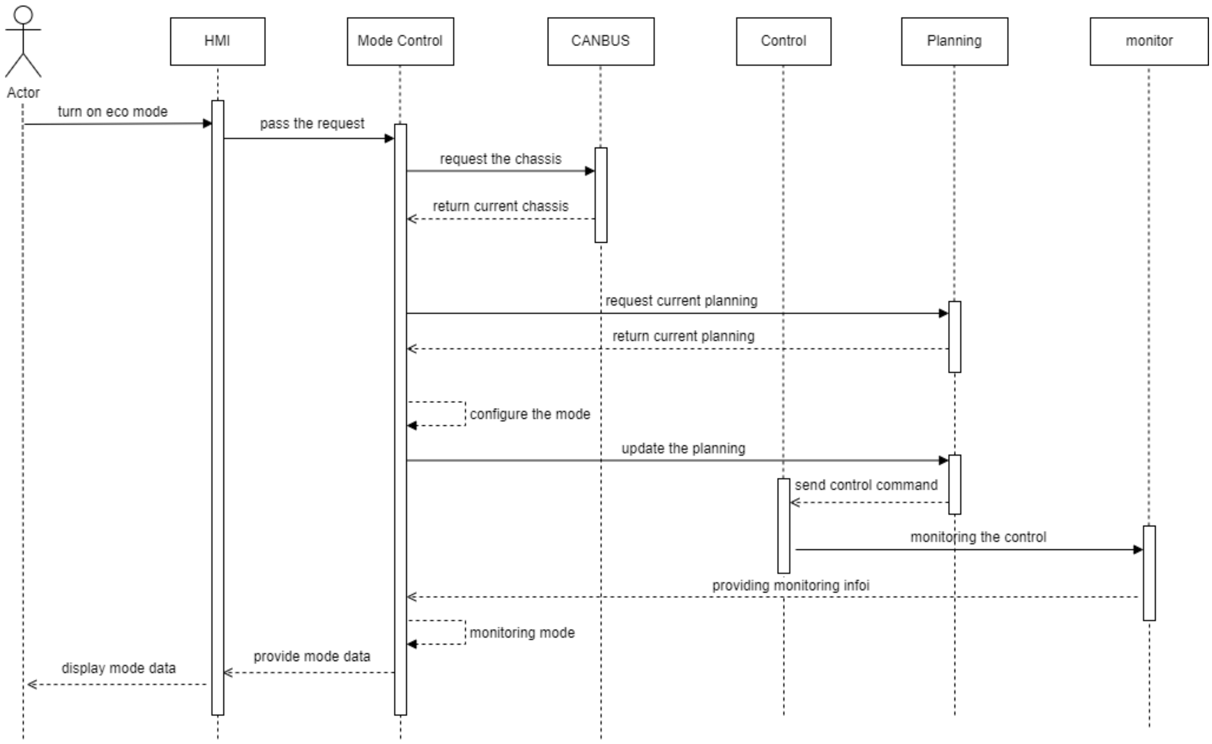

Figure 2: Sequence Diagram for the first use case

One of the use cases we introduce to this module is the eco module, in this case, the vehicle is able to start and stop smoothly to achieve the goal of energy saving. The user can select this mode in HMI and HMI would pass this message to mode control. Once the mode control receives a new mode request, it would first go through the chassis information and planning information to configure the new mode command. Once the information is collected, it will update the planning into what mode control has implemented, a new planning object. The planning will use this new planning object to control the vehicle as usual. Meanwhile, to ensure the whole driving process follows the mode smoothly, the monitor would provide information from control and let the mode control module make judgements. Everything would be shown to HMI for users to have a better view and understanding.

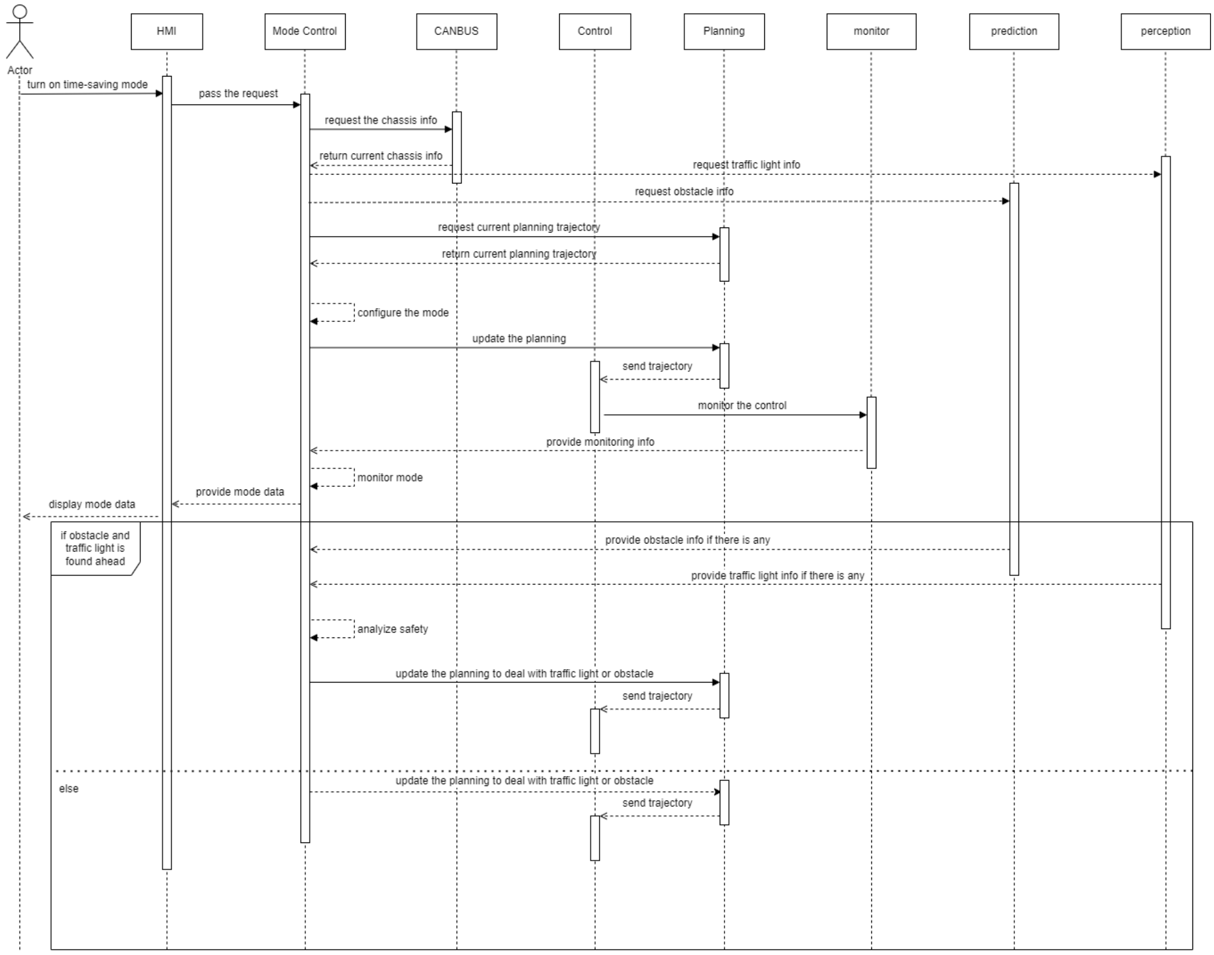

Figure 3: Sequence Diagram for the second use case

Another use case we would like to implement is Time-Saving mode. In this case, the vehicle would drive fast and safely to let the passengers get to their destination faster. Similar to the cases above the users select this mode in HMI and pass to the Mode Control module. Once the mode control changes to Time-Saving mode, it would go through the chassis information and planning information to configure the new mode command. Also, it would send requests to prediction and perception for future obstacles and traffic light information. It would update planning based on chassis and planning. Similarly, planning would control the vehicle and monitor would provide information to mode control to see if it stays in such mode. However, as Time-Saving requests drive safely and fast at the same time, once the obstacle information and traffic light information are sent to the Mode Control module, it would first do a rapid analysis of the safety. Once it gets the result, it would update the planning based on this safety result and the planning module would pass this into the control module.

Implementations

Implementation 1

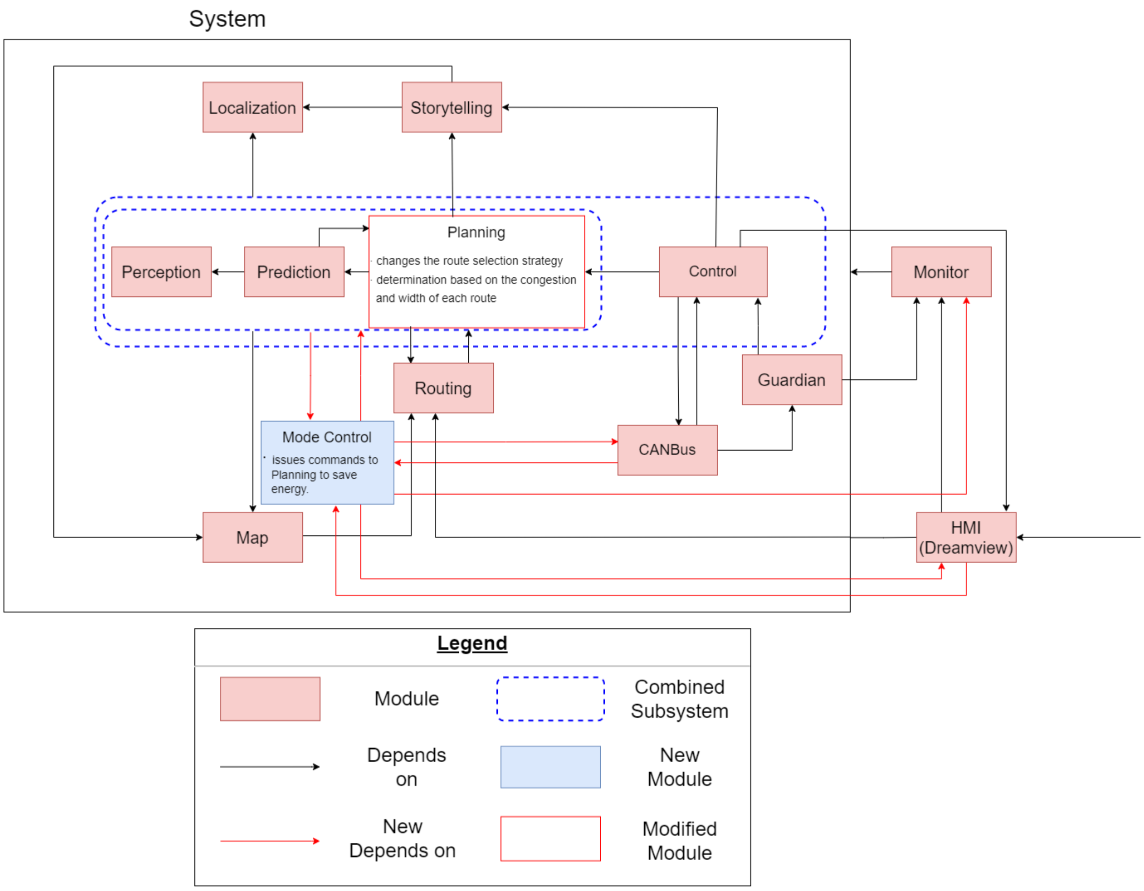

Figure 4: Implementation 1 for saving energy by modifying the Planning module.

Implementation 1 mainly invokes the mode control module with some modifications to Planning. This modification maintains the architectural style of the Pub-Sub of Apollo Autonomous Driving System.

For power saving, this implementation focuses on changes to the Planning module. When the driver selects the Energy-Saving mode, the Mode Control module issues commands to Planning to save energy. Planning will change the route selection strategy. Instead of choosing the shortest route, the least power-consuming route will be selected. The least power-consuming route is determined based on the congestion and width of each route. According to the route plan provided by Routing, Planning will give priority to the route with fewer vehicles and wider roads. In addition, Control also learns that the route is aimed at saving power, so Control also avoids unnecessary gear changes during the car driving process. In this way, the car will drive at a more balanced speed, thus reducing energy consumption.

Implementation 2

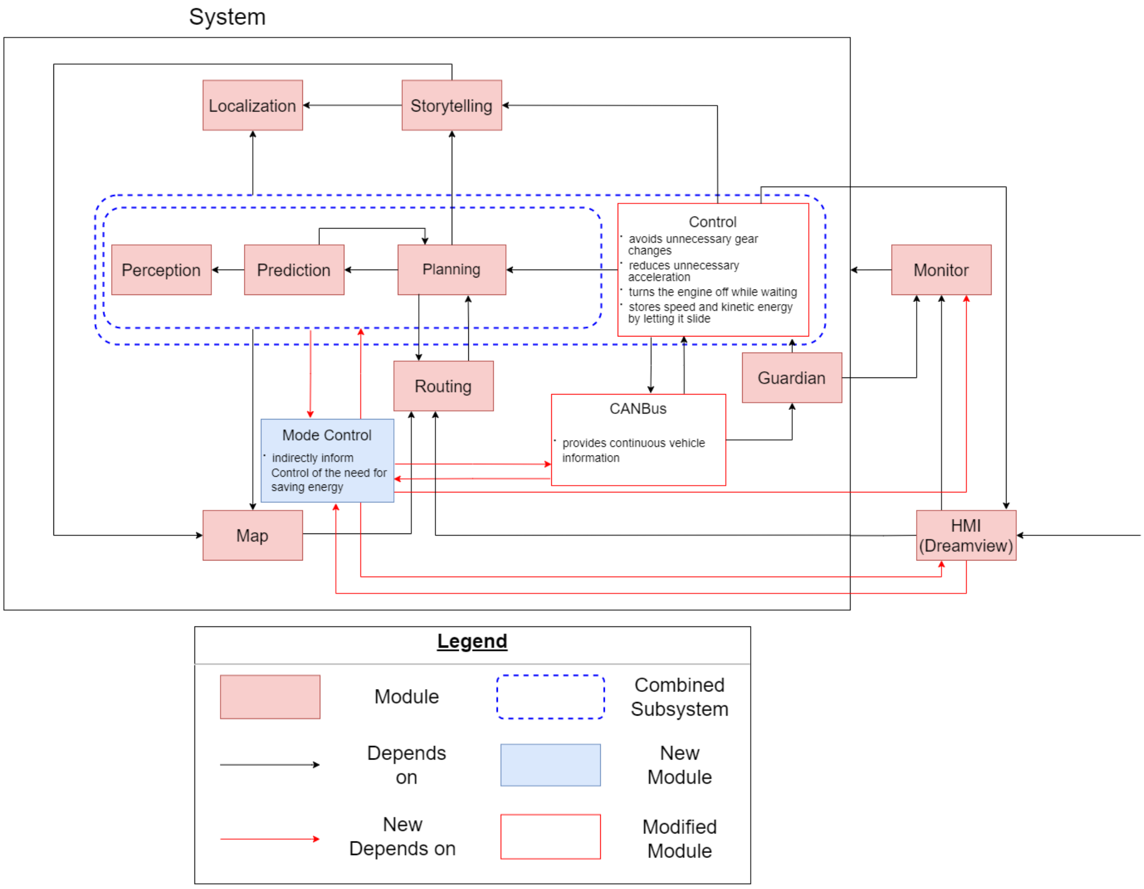

Figure 5: Implementation 2 for saving energy by modifying the Control module.

The new module Mode Control indirectly informs Control of the need for energy efficient driving by communicating with the Planning module. The modification is mainly for the Control module. By changing LonController to achieve the purpose of power saving. There are four main ways of this change. The first way, during driving, Control will try to reduce the number of gear shifts according to the real-time information from CanBUS. The second way, Control will reduce the unnecessary acceleration, so that the car can maintain a uniform speed of driving. In the third way, Control will briefly turn the engine off while waiting for a traffic light. When it is necessary to continue driving, the engine will restart again. This saves power by avoiding idling. In addition, in special situations such as going up and down hills, Control can also store speed and kinetic energy by letting the car slide. This makes the subsequent uphill climb consume less energy. This change mainly improves the kinetic efficiency of driving and thus reduces energy consumption.

Impact Subsystems

MC -> Canbus:

The Mode Control module will subscribe to the chassis_topic and chassis_detail_topic which the Canbus module published on the Common module. The Mode Control module will implement a reader and writer function that will receive the data coming from the two topics. As for some of the files in the Mode Control module they will depend on the proto file from the Canbus module just like the other modules.

MC -> Planning:

The user cases for both scenarios require dependencies on the Planning module. The Mode Control module will subscribe to the planning_trajectory_topic published by the Planning module which is the ADC trajectory.

Monitor:

The monitor module is one of the most important modules that keeps monitoring the status of the system. It will depend on many modules that directly relate to autonomous driving. In this case the Mode Control module will also be monitored. The mode_control_command_topic published in the common module will be subscribed by the channel monitor in the monitor module. Relatively the system status coming from the monitor will be subscribed by the Mode Control module

MC -> Perception & Prediction:

For these two models the Mode Control module will depend on them in time-saving mode. The subscriptions on Perception and Prediction will both contain perception and prediction obstacles topics. One more subscription on the Perception is the traffic light detection topic which the Mode Control will depend on the traffic light information to provide the output.

Planning -> MC:

The Mode Control module will publish a topic in the Common module called mode_control_command_topic that will store the output. The topic will be subscribed by the Planning module that will alternate the planning trajectory. The alternated output will then be sent to the Control module to accomplish modification to the vehicle.

HMI:

HMI is a module that can visualize the current output of autonomous driving and modify some of the module’s output. In this scenario the Mode Control module’s pattern selection will be directly controlled by the HMI. The Mode Control module should subscribe to the HMI worker and HMI status. The HMI worker is a part of HMI that can change mode, map, vehicle and driving modes [1]. Therefore the user's selection of which mode should the autonomous driving will be is sent to the Mode Control module through the HMI worker. Conversely HMI will subscribe to the mode_control_command_topic that can visualize the output of the Mode Control module.

Software Architecture Analysis Method (SAAM)

Stakeholders

The goal of our enhancement is to provide a smarter and more efficient method of driving. The stake-holders relating to this feature includes passengers, development team and the operation department of Apollo. They have essential effects on the software requirements and quality attributes of the enhancement.

Maintainability:
Both the development team and operation department value this attribute, since the development creates the code and implementation of enhancement and the operation department's job is to maintain the operation of the system.

Evolvability:
The development is the stakeholder of this since they are in charge of the following enhancement and upgrade after developing this.

Testability:
The development team is the main stakeholder relating to these quality attributes. Because they are the only ones who are in charge of testing before the release.

Performance:
The stakeholders of these attributes include users, development team and the operation departments. The user is the most important stake-holder for this attribute since the overall goal of this enhancement is to improve the user's experience through improving the performance. The development team is responsible for the performance of the enhancement during the development, and the operation team is requested to maintain the quality of the system after it releases.

Quality Attributes

As we talked about previously, our enhancement has done a great number of effects and modifications to the overall system of apollo. Such impacts also appear in the non-functional requirements of the Apollo system. The effects of enhancement on maintainability, evolvability, testability, performance of the system will be discussed:

Maintainability:

Our enhancement will have a minor effect on the maintainability to the maintainability to the overall system. The software's maintainability defines how well software can be maintained with correct defects, repair faults, prevent unexpected situations and maximize a product's life[wiki]. This means that the system has to be understood and repaired. [www.castsoftware.com]. Firstly, the Apollo system still can be easily understood (we have discussed it in XXXX). Secondly, the degree of difficulties in repairing the system does not increase a lot after deploying the enhancements. Our enhancement mainly focuses on creating module Mode Control and modifying some original modules. If bugs or defects relating to mode control are discovered, the repair work will be mainly focused on these modules.

Evolvability:

Evolvability refers to the poverty of how easily the system can be improved or upgraded. Our enhancement does not change the original architecture style of the system which is pub-sub. Pub-sub architecture has very high modifiability which means that it is easy to develop evaluation in it. Moreover, our enhancement does not modify the dependent relationship of pre-existed modules, so it does not influence evolvability significantly.

Testability:

The enhancement will complexify and increase the process of testing the overall system. Our enhancement brings the system a large functional update that uses the cooperation of multiple sub-modules, in order to test the feature of the enhancements, multiple other test methods have to be developed.

Performance:

Our enhancement will highly strengthen the performance, especially the user's experience. The goal of our enhancement is to provide a smarter control solution based on the driver or passengers' choices. Our improvement in performance can be measured as the time saved or energy saved compared to the original solution based on different situations.

Non-Functional Requirements

Maintainability: "99 percent of the system updates will not affect the user's driving."

Evolvability: "New traffic information can be updated through the Internet."

Testability: "More than 3 percent of energy will be saved when users drive in ECO mode."

Performance: "CanBUS can provide vehicle information to the Control module in real time."

Comparison of Implementations

At the maintainability level, implementation 1 has the advantage over implementation 2. The reason for this is mainly because the failures in implementation 1 will mainly occur in the problem of road selection. Such faults can be solved by optimizing the algorithm for the developer. Therefore, it is easier to maintain. Implementation 2, on the other hand, generates a variety of errors, such as unstable gears and speed control. These errors are often caused by different factors. Developers need to fully understand the causes in order to perform maintenance. This undoubtedly makes maintainability much worse.

For evolvability, implementation 1 is a bit better. This is mainly because implementation 1 is achieved mainly by changing the original algorithm. The algorithm can be improved over time. Therefore, implementation 1 has better evolvability. However, implementation 2 has some disadvantages in this region. The main reason is that this implementation is mainly done by changing the vehicle state. This makes the developer constrained to improve the software because of the limits of hardware (e.g., the vehicle's engine, transmission, etc.). This limits the evolvability of implementation 2.

In terms of testability, there is no difference between these two. Implementation 1 requires a lot of testing to ensure the stability of the solution in order to find the best route to save energy based on different urban road conditions and traffic congestion at different times of the day. In addition, implementation 2 requires a lot of testing to ensure the stability of the vehicle's speed and gearing. Therefore, both have excellent testability.

Implementation 2 is much better in terms of performance. By optimizing the vehicle (speed and gears) and road conditions (signals and hills), implementation 2 shows its performance advantage in terms of breadth. On the other hand, implementation 1 is slightly inferior in terms of performance. This is because the performance of implementation 1 is only in the area of route selection.

In a comprehensive comparison, although implementation 2 excels in performance, implementation 1 shows its unique advantages in other aspects. Therefore, implementation 1 is the more appropriate choice.

Plan for Testing

Our enhancement provides multiple new drives that are active based on the user's decisions. As we have discussed previously, the feature is implemented through the cooperation of various existing modules and a new module mode control. New tests with regard to this feature are necessary. We expect to use both black box testing and white box testing for it.

For the black box testing, we plan to develop an input coverage test to cover all the possible inputs. We plan to analyze all the possible inputs and create test cases based on that. For two modes: energy saving and time-saving mode we will partition the inputs into simple road conditions and complex road conditions. While in the simple road conditions we create enough test cases in simple routes with a small number of traffic lights and obstacles to analyze if the goal of the mode (time saved, energy saved) is met or not. In complex road conditions, we will create test cases located on complex routes with a large number of traffic lights and obstacles. In this situation, we will calculate the expected outputs including driving time, trajectory, and fuel consumption, and compare them to the actual result in the test case.

For white box testing, we plan to create test cases to cover and execute every method statement or instruction of the code that relates to the new feature. We will design different variables to input to the program that will lead it to execute every path and if-else branch, true and false for every Boolean and both lower and upper boundaries for every variable, and study and analyze their results to debug. This process will take a long time, but it is necessary since our new feature is strongly related to the passengers' safety.

Potential Risk

One of the potential risks of this mode control is reliability. Since there are so many safety concerns and situations changing every moment on the road, mode control cannot simply use some linear functions to update the planning each time. Therefore, a deep learning network is requested for mode control to detect and provide information. Especially for the Time-Saving mode, driving fast and safely at same time for a person can be hard. So, for a machine, it is hard for it to learn to deal with some situations that it has never learned before. Another concern is that it can cause bad users' experience. In daily experiences, some passengers are not comfortable with centripetal force when the vehicle is at a fast speed. It is hard for AI to balance the drive mode that different passengers are comfortable with.

Limitations and Lessons Learned

At the beginning stage of this report, our team conducted several meetings in order to find a proper topic as a new feature/enhancement to be placed on the Apollo system. At first, we tried to implement an information share module that took place between two adjacent Apollo autonomous driving vehicles. We have asked the advice from TA and professor, both of them mention that some of the function is actually contained in the V2X adapter. As a result, we conducted another meeting and came up with the idea that we are currently presenting. In addition, the research and development cost might not be as worth as installing Apollo to electric cars since electric cars have become more and more popular and environmentally friendly. Last but not least, the alternative approach that we are proposing might meet problems like the need for extra external hardware. There is no existing hardware that supports saving energy through driving down slopes and this problem is impossible to solve just in terms of the coding level.

In this assignment, we have learnt valuable courses. The new feature enhancement process helps our team develop a stronger and deeper understanding of the interaction between original Apollo modules. Besides, we have a chance to know diverse interesting features that were and will be applied into vehicles during the investigation. The two approaches we are presenting are two iconic examples. If we do not get into this part we will never know power could be saved during the sliding down slopes.

Conclusion

In conclusion, we implement an enhancement on increasing the driving experience. Since it's not a revolutionary feature and the architecture style still follows the publish-subscription style, this enhancement is aiming at helping drivers get vehicle configurations in order to suit different driving scenarios that might come up in real life cases. Although there still exist some potential risk and limitations and this enhancement still needs to be polished and honed.

Glossary

V2X: Vehicle-to-everything, is a communication between a vehicle and any entity that may affect, or may be affected by, the vehicle. It is a vehicular communication system that incorporates other more specific types of communication as V2I (vehicle-to-infrastructure), V2N (vehicle-to-network), V2V (vehicle-to-vehicle), V2P (vehicle-to-pedestrian), V2D (vehicle-to-device). [2]

SAAM: It's a method for analyzing the properties of software architecture[3]

Reference

[1]ApolloAuto. (n.d.). *Apollo/modules/dreamview/backend/HMI at R7.0.0 · ApolloAuto/apollo*. GitHub. Retrieved April 7, 2022, from https://github.com/ApolloAuto/apollo/tree/r7.0.0/modules/dreamview/backend/hmi

[2]Wikimedia Foundation. (2022, February 27). *Vehicle-to-everything*. Wikipedia. Retrieved April 7, 2022, from https://en.wikipedia.org/wiki/Vehicle-to-everything

[3]Kazman, R., Bass, L., Webb, M., & Abowd, G. (1994, January 1). *[PDF] Saam: A method for analyzing the properties of software architectures: Semantic scholar*. undefined. Retrieved April 9, 2022, from https://www.semanticscholar.org/paper/SAAM%3A-a-method-for-analyzing-the-properties-of-Kazman-Bass/c22fce05aa0a8333284bea57401fd7d07d4116f9