CISC 322

Assignment 3
Enhancement Architecture of Apollo
April 9, 2022

Runfeng Qian (Group Leader)	18rq10@queensu.ca
Wonton Zhou	19bz6@queensu.ca
Ziheng Yang	18zy59@queensu.ca
Runze Lin	18rl30@queensu.ca
Junyu Yan	19jy28@queensu.ca
Haoyun Yang	18hy58@queensu.ca

Abstract

This report proposes an enhancement feature on the open source Apollo autonomous driving platform. We added an enhancement to the detection of the abnormal and complex road conditions for the Apollo system, also an adjustment to the performance of the driving strategy when encountering congested road conditions, and adding countermeasures to special conditions. This proposal goes through the SAAM analysis method that defines the NFR of the functionality and compares the enhancement implementation with the NFR and stakeholders. Then we came up with two use cases with the enhancement feature we proposed in the form of sequence diagrams, showcasing how the enhancement performs in the Apollo system. Hence we concluded the limitations we encountered and lessons learned and eventually made a summary of the enhancement feature we proposed and beyond.

Introduction

With the previous reports of the deep analysis of the conceptual and concrete architectures of the Apollo program, we have the knowledge base needed to propose the enhancement features to the Apollo program. In this paper, we add functionalities that allow the Apollo autonomous to perceive the road condition such as how slippery the road surface is and the traffic conditions, and then choose the most proper driving strategy corresponding to it. Additionally, we also enhanced the architecture by adding functionalities to perceive nearby vehicles of special types, such as ambulances and fire trucks, hence the Apollo system automatically takes actions to keep a safe distance from them.

To approach these enhancements, we will add a brand new module called RT Information API, which is used to store information online about the special types of vehicles and special weather that we are about to encounter. The information can be the features of the special vehicles in appearance or the amount of the frictional force on a snowy day or anything that can be used to compare with the currently perceived state and condition. In this enhancement, planning, prediction, perception and RT info API are the four key modules that we focus on.

The planning module will equip two new submodules called Environment Condition Handler and Emergency Vehicle Handler. Just like how these two are named, the Environment Condition Handler module is responsible to plan a proper route for the vehicle according to the information received from the API, and on the other hand, the Emergency Vehicle Handler submodule will take charge of gathering information about the surrounding special vehicles' positions and their potential driving strategies. In the perception module, we introduce two new submodules as well, called Road Condition Sensing and Special Vehicle Recognition. The RCS module perceives the real-time road condition while the SVR module will recognize the movement of the special vehicles by analyzing the output of the map module and the perception module (radar, camera and other perception tools).

Next, we will briefly explain how the enhanced mechanism works. To start with, we will create a new module, RT info API which can connect to online databases(as introduced above) and localize it. The planning module will take the information outputted from the RT info API as input and generate a route and driving strategies that are proper and safe in the new newly added submodules, the ECH submodule and the EAH submodule, based on the road condition or the positions of the special vehicles perceived in the RCS and SVR submodule equipped in the perception module. In addition, the map module will also provide information about the traffic condition to support the planning module to generate a route that avoids traffic problems as much as possible. Finally, the control module will receive the

driving plans from the planning module and follow the driving strategies to control the vehicle. This is basically how our ideal enhancement on the Apollo autonomous works. In order to achieve such functionalities in real life, there are still a lot of practical issues to solve and many more experiments are required.

Enhancement Feature

The feature we proposed is the function of detection and response to special road conditions. It includes special vehicle avoidance, special road surface conditions handling and congested road avoidance.

For special vehicle avoidance, the autonomous vehicle will detect the type of the rear target (such as vehicle type containing ambulance, fire truck, police car, etc.), detect the possibility of lane change from the adjacent lane in front of the car, and actively initiate and execute the lane change control request to give way to the car behind.

The vehicle will monitor the driving road surface through the sensor in real-time for special road surface conditions. When the road environment has potholes, bumps, and slippery ice, it will pre-judge whether the car can drive through the irregular road surface and use the intelligent suspension, safety brake, safety steering, and other ways in advance.

For congested road avoidance, The vehicle can get the real-time congestion information of the road from the cloud and then calculate the optimal route.

Conceptual Architecture Recap

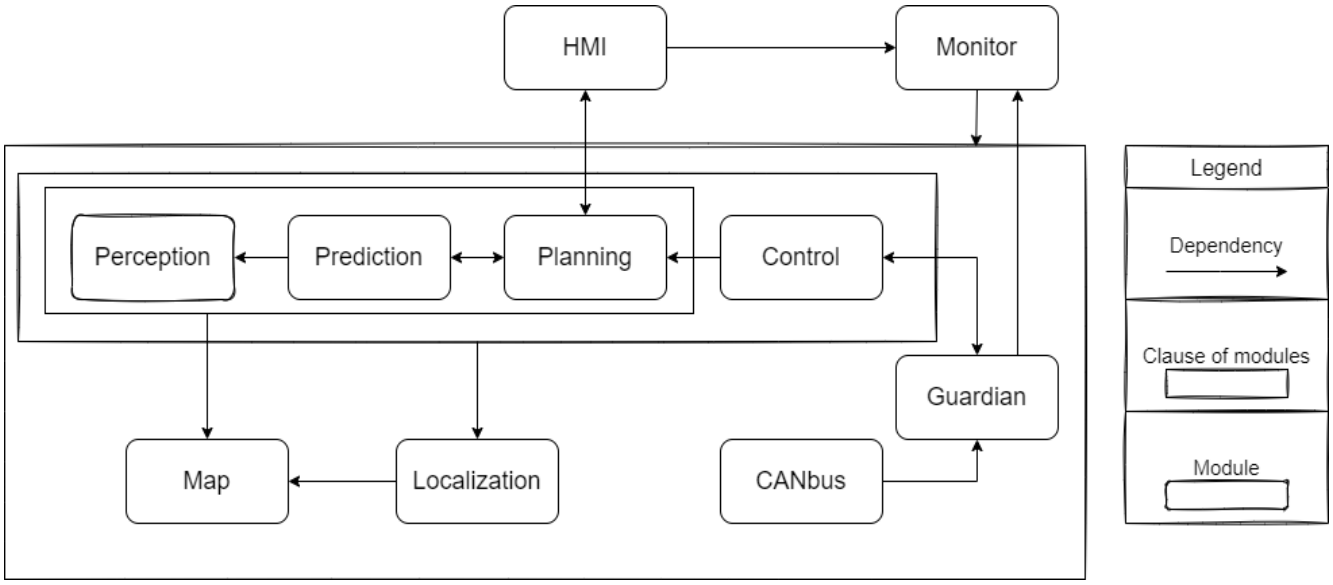

figure 1: top-level conceptual architecture

Map Engine sends map information to the localization component. The localization component obtains the precise location on the map and sends them to other modules. The perception module collects environment information from sensors and receives the information from the localization component. The Perception module sends the 3D obstacle tracks to the prediction component. After combining the outputs from Map, Localization and Perception module, the prediction module will calculate and output information of obstacles

annotated with predicted trajectories and their priorities to the planning module. After obtaining information from previous modules, the planning module generates a feasible trajectory and sends it to the control component. Depending on the planned trajectory, the control module sends the control command to canbus[1]. HMI provides a human-machine interaction interface and visualizes information from all other modules for easy manipulation. Control and guardian modules interact with each other. The monitor module receives data from all other modules.[2]

Non-Functional Requirement

There are three Non-Functional Requirements we have concluded based on our enhancements. These NFRs include performance, accuracy, maintainability and security.

Performance: High safety measures in extreme weather conditions. No sliding, better response time to extreme road conditions, and being able to respond within 0.1ms. Faster recognition and response to special vehicles. The avoidance strategy should be made within 0.1ms. We need to reach a high-performance level in order to be more competitive in the market compared with other companies who are potentially doing similar product research.

Accuracy: 99.9% of accuracy in the planning module of the system. When encountering special vehicles, the Apollo autonomous will make sure that it keeps a safe distance from them and the accuracy of the error of the distance between vehicles judged by AI is not less than 99.9%. Additionally, the perception module will recognize and classify different road surface conditions when perceiving the road surface condition. The accuracy of the classification should be higher than 99.9%.

Maintainability: the system of this feature should be able to be tested easily. When the system fails, the system of this feature must be able to be easily fixed without affecting the functionality of other systems. Having a high maintainability level can also benefit the after-sale service. This means that the cost of fixing the system will be lowered financially and physically.

Security: In order to detect special vehicles earlier, the system will get real-time information of special vehicles from the cloud. However, this information is often non-public and needs to be kept confidential. Therefore, Special vehicle information obtained from the cloud should be well encrypted to prevent data leakage. A high-security level will prevent the secret data from being revealed by hackers’ attacks or viruses and protect the Apollo company's rights and its passengers.

Implementation

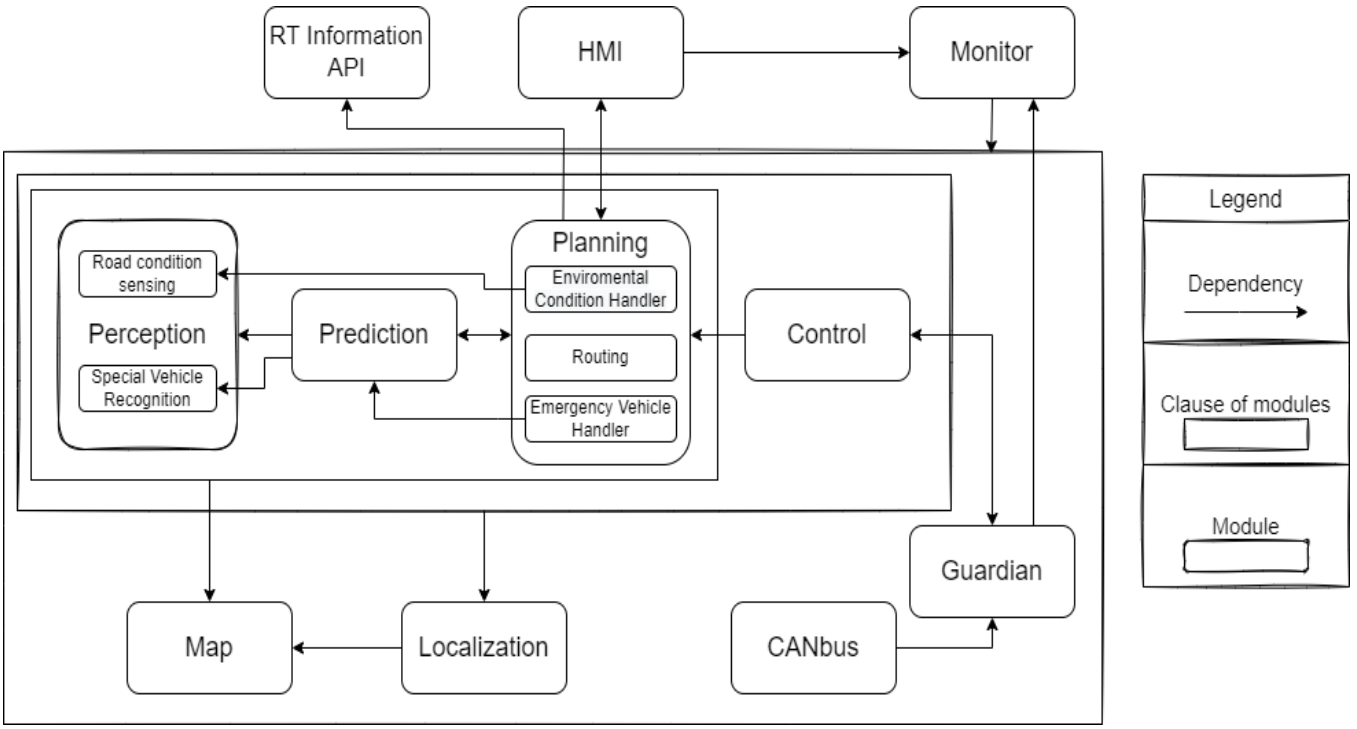

Figure 2. Implementation of features into conceptual architecture subsystems

The above figure is our first implementation design for the extension. We added a new top-level module, RT information API. There are two new second-level submodules in the perception module and two new second-level submodules in the planning module. The architecture style does not change; it still has a pub-sub architecture style and a pipe and filter architecture.

The RT information API is a cloud data interface that will collect some real-time information such as real-time climate conditions, real-time road congestion and real-time special vehicle traffic situations from the cloud database.

The two added second-level submodules in the perception module are road condition sensing and special vehicle recognition. Road condition sensing module can observe abnormal road conditions, such as potholes, bumps, water, ice, etc., through sensors such as LIDAR and cameras. Special vehicle recognition modules can identify special vehicles in the rear, such as ambulances, fire trucks and police cars on the mission.

The two added second-level submodules in the planning module are environmental condition handler and emergency vehicle handler. The environmental condition handler will set the appropriate driving strategy for different road conditions. For example, it will calculate a slower, smoother turning trajectory when the road is icy. The emergency vehicle handler module can calculate the trajectory in order to make reasonable avoidance of special vehicles coming from behind. The second level in the planning module, routing, is also modified. It can collect data on real-time congestion conditions from RT information API and compute the long-term route based on that information to avoid congested roads.

There are also some dependencies added to the original conceptual architecture. The environmental condition handler depends on road condition sensing. The emergency vehicle handler depends on the prediction module. The environmental condition handler and The emergency vehicle handler also depend on RT information API for cross verification. When it

senses icy weather, the environmental condition handler will check the weather information to confirm if the sensor is wrong. In addition, the emergency vehicle handler can get information via RT information API on whether a special vehicle will pass before it is observed.

Alternative Implementation

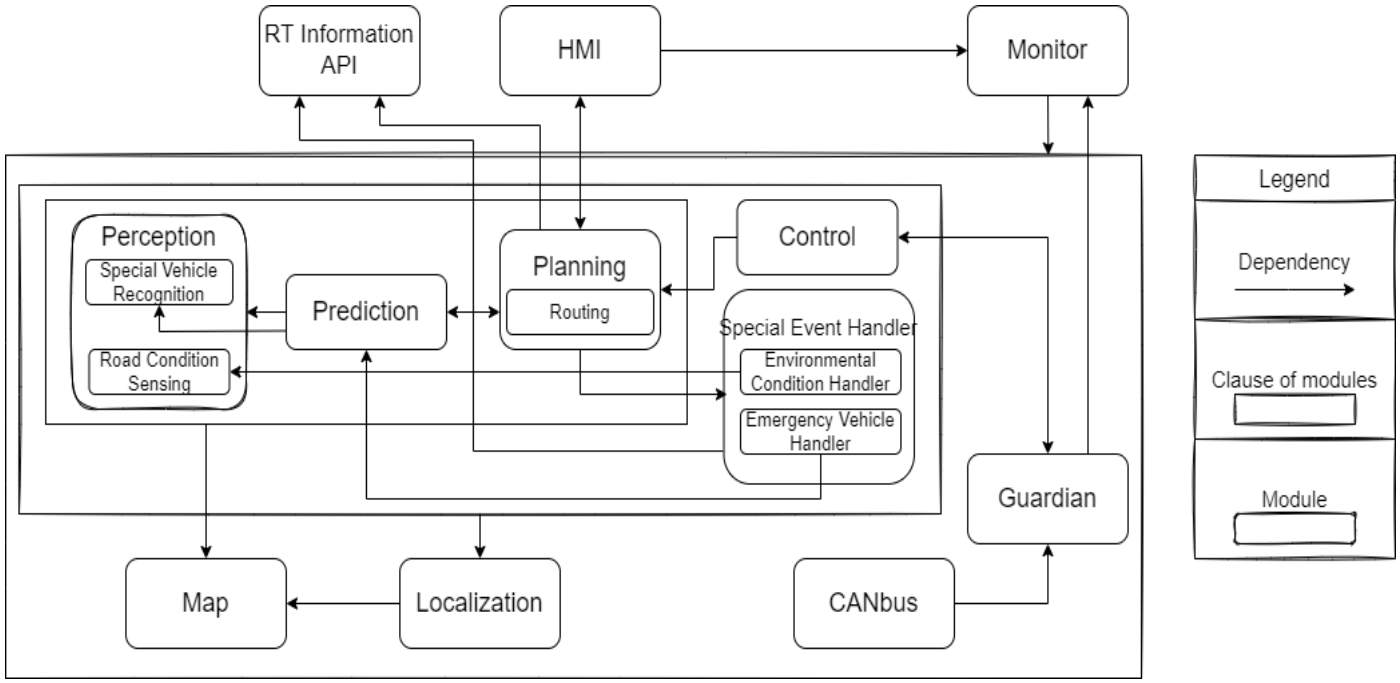

Figure 3. Implementation of features into conceptual architecture subsystems by the alternative approach

The above figure is our alternative implementation design for the extension. The architecture style does not change; it still has a pub-sub architecture style and a pipe and filter architecture. The difference between approach 2 and approach 1 is that we take out the environmental condition handler and the emergency vehicle handler from the planning module and merge them into a new top-level module called the special event handler. This modification led to some functional changes in these two submodules. The environmental condition handler and emergency vehicle handler will not generate trajectories themselves. The planning module will generate an appropriate trajectory based on the information for the special event handler.

Effects on Maintainability&Evolvability&Testability&Performance

The maintainability of the first approach will decrease a lot compared to the original architecture because we extend the scale of the perception module and the planning module. If an error occurs, the developer is required to maintain all the code inside the error module, and the workload of maintaining the system is possibly high. Compared to the first approach, the second approach is relatively easy to maintain because the special case handler module processes the special task separately (loose coupling than the other module, which means the

system can still run if the code fails in this module), so it is easier to locate the error and fix, and update the existing special event.

The evolvability of the first approach will increase when adding more features because the increment of the planning and perception module will add more dependencies inside the subsystem and may completely change the current architecture of the subsystem to achieve the required approach. The second approach is relatively easy to evolve compared to the first approach. Since we added all the special handlers to the specific module, all the other adding features do not need to concern too much about the special handler module and also adding the new special case is much easier.

The testability of the two implements is the same, before the system starts each test module tests each function works well, and then the system will build a simulated environment to test the module functionality, and if the program is able to pass the simulation, the test is successful.

The performance of the first approach is high because we do not change the current high-level architecture and just add the two modules in the submodule, the top-level sequence didn't change, and the planning module can process the trajectory based on the direct data from the perception. Compared to the first approach, the second approach performance is relatively lower, because the special handler is separated from the planning, if the process of the handler module cannot be done before the planning module start, will cause the planning module still use the previous data provided by handler module or planning module wait for the new data both situation will cause the performance drop.

SAAM Analysis

Stakeholders

Users

Users of the Apollo autonomous driving system are the main stakeholders of our proposed enhancement. The users benefited directly from our feature and enhancement's performance and accuracy. The users are mostly concerned with the accuracy of the system since their safety is directly dependent on these NFRs. Misjudgment of road conditions can again lead to serious consequences, for example, when the system fails to determine the icy road ahead and thus fails to slow down, it may result in a crash. Of course, lower response time means that the vehicle will have a longer time to deal with special situations, which will also improve safety to a certain extent.

Apollo System Developers

Developers and the working staff of the Apollo autonomous driving system are the second-largest stakeholders of the Apollo autonomous driving system. The developers are mostly concerned with testability and maintainability. Better testability and maintainability means less work, so the enhanced features will not affect the deadline of the deliverable date too much.

Emergency workers

One feature of our added feature is that the Apollo platform will detect the position and potential driving strategies of the surrounding special vehicles such as fire trucks and ambulances and then plan a vehicle's route based on the surrounding situation to avoid blocking their road. This enhancement will increase the working efficiency of the special

vehicles since the possibility for them to be interrupted halfway to their working place by the traffic problems will be lowered as all other surrounding vehicles have planned routes to avoid bothering their work. And the NFRs of them are accuracy and performance, which means that the system can handle the situation with less time consumed and higher correctness. Besides, the entire citizens will be potentially benefited since the working efficiency of the emergency workers increases. When they are the target of the emergency workers or they are involved in the work (accident, criminal cases), their problems and difficulties will be solved sooner and then become one of the stakeholders. In addition, emergency workers are also concerned about the security NFR. The real-time information of special vehicles is often non-public and needs to be kept confidential. Therefore, this data needs to be highly encrypted to prevent data leakage.

Investors

Investors and stockholders of the Baidu corporation would benefit directly from our enhancement feature as the stakeholder of the Apollo system. Since the enhanced system would be very competitive in the market with its performance, accuracy and reliability. Hence the market would recognize this enhanced system more than the other competitors. Therefore, the investors and stockholders of the Baidu corporation will profit directly from this recognition in the market. Investors do not have a specific special NFR; the only requirement for them is profit maximization.

Impacts of Suggest Ways(Implementation) on NFR and Stakeholder

The alternative implementation of our enhancement feature will slightly influence the performance of the system, specifically the response time of the system. However, our first implementation does not have this issue since the interactions between the planning module and other modules are different in these two implementations, and the alternative implementation takes a longer route for data to be transferred to the planning module. Hence the performance of this alternative implementation is slightly worse than the original implementation. Slower response times mean higher security risks, so users prefer to use the first approach. Another impact we have discovered is that the alternative implementation has a unique module called the special event handler, this module passes information to the planning module, and hence the system is going to be easier to maintain and more reliable because the environmental condition handler and emergency vehicle handler are separate from the planning module. We only need to fix the special event handler module when the proposed feature fails. However, splitting modules with similar functions tends to create a bloated system architecture.

In conclusion, we think approach 1 is a better method to implement our proposed feature.

User cases

Use case 1: Apollo plans route and starts trip according to address provided by passenger and road congestion information

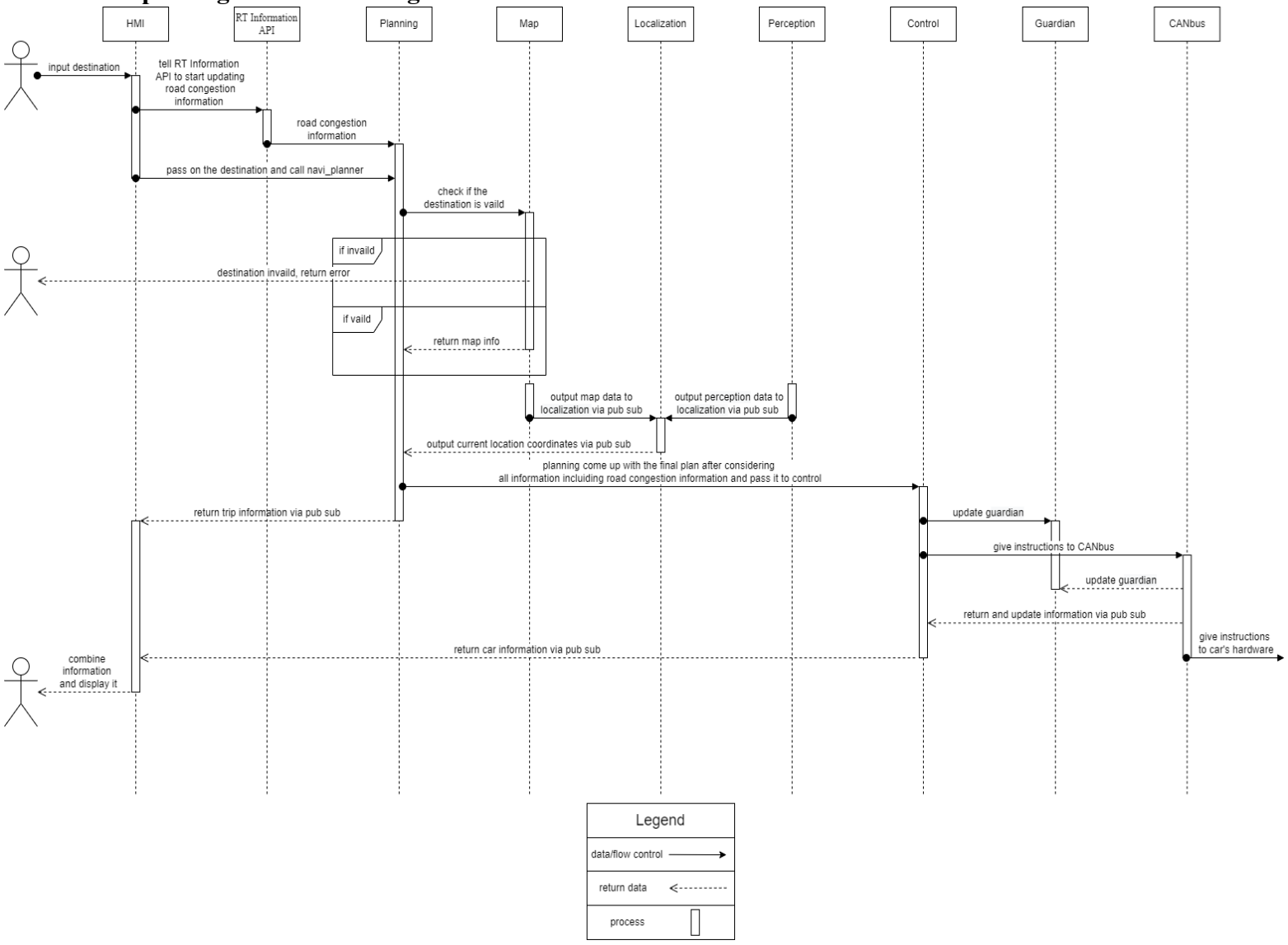

figure 4: sequence diagram of use case 1: Apollo gives way to an emergency vehicle.

In user case 1, the user first enters the destination, then HMI tells RT Information API to subscribe information in the cloud, keeps updating the road congestion information and pass it to the planning module to help finding the best route.The HMI also gives the destination to the planning module by calling the navi_planner function in the planner folder in the planning module. Then the planning module calls functions in HD map to get map information, if the map module finds that the destination address is invalid, it quits and informs the user that the address is invaild. If the address is valid, the map module returns the map data to the planning module. After that, the localization module gathers map data and perception data from the map module and the perception module to come up with the current location coordinate and gives it to the planning module. After receiving data from the three modules, the planning module now has enough information to find out the best route to the destination. So it publishes the optimized route. Then the control module gets the route and outputs direction

instructions. These instructions are then further processed by the CANbus which translates it to outputs that the mechanical components can understand, and send to the car. Lastly, CANbus gathers car information and sends it back to the control module for analysis. HMI meanwhile, gets updated information from many modules like trip plan from planning and car status from control, and outputs it to the user.

Use case 2: Apollo encounters special situation such as dangerous weather and emergency vehicles while driving

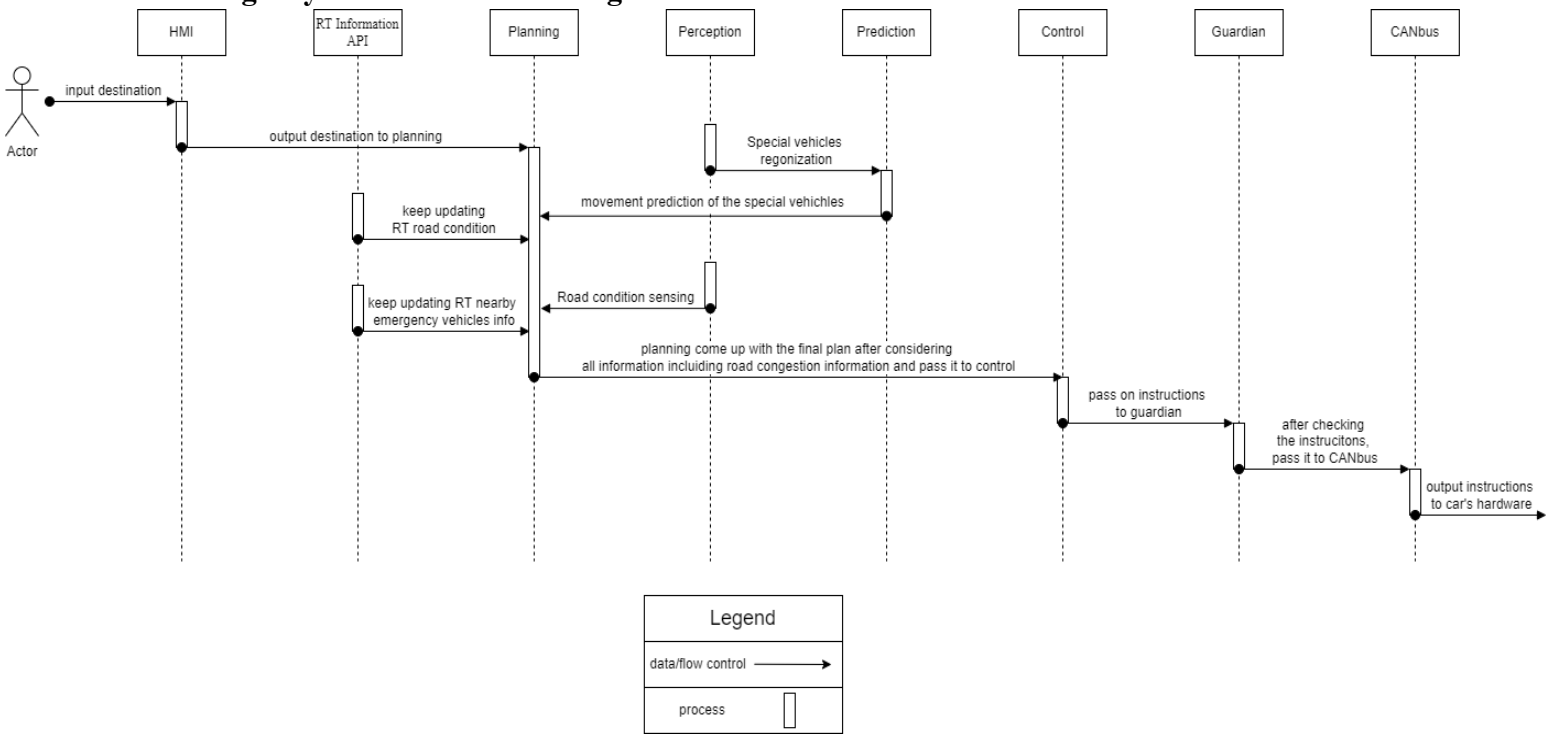

figure 5: sequence diagram of use case 2: Apollo handles special situation

In user case 2, after the user input his/her destination, RT information API and the perception module keeps updating information of emergency vehicles and road conditions. For emergency vehicles, the perception module senses the speed, location and vector of the emergency vehicle and passes the information to the prediction module. The prediction module then makes a prediction of the emergency vehicle and returns the information to planning. After the planning module receives information of those special conditions, it calculates the optimized plan and passes it to the control module. Then the control module gives the final instruction to CANbus through the guardian module.

In user case 2, after the user input his/her destination, RT information API and the perception module keeps updating information of emergency vehicles and road conditions. For emergency vehicles, the perception module senses the speed, location and vector of the emergency vehicle and passes the information to the prediction module. The prediction module then makes a prediction of the emergency vehicle and returns the information to planning. After the planning module receives information of those special conditions, it calculates the optimized plan and passes it to the control module. Then the control module gives the final instruction to CANbus through the guardian module.

Testing Plans

Firstly, we will test whether the newly added module can function well, which means, the road condition can process the sensor data to give the correct condition result. And giving an image of the special vehicle to the module, the special vehicle recognition can recognize it as a special vehicle. The emergency vehicle handler and environment condition handler can output the correct trajectory to let the control module to execute.

After that, we will measure whether the module's newly added communication function works well or not. The API information can be taken anytime when the planning module needs it, and that information should be up-to-date in order to handle the special situation, specifically the emergency vehicle path. The road condition sensing is able to publish the road information to the broadcast system and the environmental condition handler subscribes to information that can be correctly received. Special vehicle recognition module messages passed to prediction and prediction data messages passed to special vehicle handlers can also function well.

In the enhanced version of our architecture, we have to make sure that there are no negative impacts on other previously existing modules. In simple words, we need to examine whether the old module still works well after adding new features. For instance, the perception module is equipped with two new submodules, RCS and SVR, and we need to test whether it is still capable of performing its original functionalities like perceiving traffic lights and traffic obstacles. Also, we can test the planning module, with EVH and ECH mounted, to see whether it will still pick the least time-consuming path while making sure there will not be any problems with emergency vehicles or road conditions due to weather and traffic jams.

Once the program is implemented, it will be tested on the Apollo cloud service platform for the ability to respond to and handle different usage scenarios first. If it passes the test, a dedicated test team will conduct realistic scenario road tests. If the software fails the test, the test team will check the information stored in the BlackBox, analyze it and find the component(s) responsible for the failure and then tell the corresponding team for modification. The test team will also upload the failure analysis to the cloud, providing lessons for subsequent systems to avoid making the same mistakes again. After passing the test on the Apollo cloud service, we planned the following practical test example cases to check whether the usability, reliability and other requirements are fulfilled in the enhancement:

1. Use the Apollo autonomous system to drive on several icy roads of different icy extents and check whether it will automatically reduce the vehicle speed to a level that corresponds to how icy the road is.
2. Use the Apollo autonomous to plan a route that goes through the center of the city where it will potentially encounter traffic issues and then see whether Apollo has planned a route to avoid all these traffic jams. Also, we can compare the time consumed using normal gravitation with the route planned by Apollo. This will reflect the usability of one of our enhancements.
3. Use the Apollo autonomous system to drive in traffic lines that have lots of hospitals or police offices on the side and observe whether the Apollo vehicle appears to be in front of the special vehicles directly or on the right side of it when the special vehicle is about to turn right (vehicles with our enhanced Apollo system should not appear in the front of the emergency vehicles directly). If not, it means that our enhancement does have reliability in the aspect of avoiding special vehicles.

Before putting this enhanced version of the Apollo system on the market, we will test it strictly in all aspects to guarantee passenger safety. We will test all the source code both separately and entirely, and keep editing it if there is any possibility for a potential security or driving planning issue to occur.

Limitation and Risks

While writing this report, our team met certain difficult limitations, but the experiences we gained from the previous two assignments helped us through most of the limitations. Hence we actually have fewer limitations than in the previous two assignments. The first limitation we encountered was the time management of this report; the busy time schedule for every one of the team really troubled us. However, we still managed to have everyone doing their work on time in the last week of the semester. The second limitation we encountered was looking for the affected source codes in the proposed enhancement feature. It was difficult to do so since Apollo is such a huge project, we were able to locate the top-level modules affected by our enhancement feature, but at low-level files, it was a hard task to do indeed. The third limitation we have is on a broader level; when we were trying to come up with an enhancement feature of the Apollo program, we found that almost every aspect of Apollo has been considered by the devs of the program, a previous couple of features we came up with eventually turned out to be already in the Apollo system. This limitation was a bit hard for us to overcome since we are trying to make a program better that is taken care of by thousands of programmers daily.

Risks include data safety since our enhancement requires the ambulance and fire trucks to upload their location info to the internet; hence, the system can track their locations and avoid them if needed. The location info should be classified data for the Apollo system to use, and this information should not go public. Because this information will link to the autonomous system vehicles, some criminals may steal the information from the vehicles with the Apollo system, which may cause security issues and concerns around sensitive data safety. It's the first risk our enhancement feature may encounter. The second risk we have encountered is the maintainability of the enhancement feature since this feature is mostly dependent on the internet around it. If a vehicle with the enhanced Apollo system goes into a region without any internet connection, then the maintainability of the enhancement feature would go wrong completely. This is a huge but understandable risk since there’s no autonomous driving system that can go offline so far. The third risk we have discovered with our enhancement feature is the compatibility issue. If the input and output data of our modified modules are not the same format, there will be a compatibility issue, but this issue is within the acceptance level. However, this risk won’t affect the usability of stakeholders as long as the input and output data are in the same format.

Lessons Learned and Team Issues

After adding the enhancement feature to the Apollo program and analyzing it with the SAAM method, we have gained some valuable experience on how to interpret and analyze huge programs like the Apollo system. Based on our conceptual architecture, we built the enhancement and added some supportive functionalities. In this process, we better understand the immense difficulty of implementing small features into large codebases such as Apollo autonomous. According to the course material of week 2, we have to make sure that our enhancement meets both functional and non-functional requirements. For the functional

requirements, we have to specify the function of the enhanced part of the system and figure out the connections between the newly equipped modules and submodules. On the other hand, for the non-functional requirements, we have to consider the quality, managerial requirements and the range of conditions in which our enhanced features can be operated. The only step that we lack here is the testing, and unfortunately, this step is not practical so far since everything we have enhanced so far is conceptual. For the criterias above, we have introduced them in this report in detail since we have learned in class about how to build a decent enhancement that is usable and reachable.

In the aspect of the team issues, our team mainly had the problem of time management of the project due to the massive amount of tasks and exams to do in the last week of the semester. Thankfully, the good prof granted us a few extra days to work on the report. In terms of other team issues and concurrency, our team had some issues with the discussions of the enhancement feature; almost every one of us has different opinions over what enhancement feature we should add to the Apollo system. However, we eventually convinced each other by combining many of the most approved features into one enhancement and adding it to the Apollo system. Besides some inconsistent discussion and time management of the project, our team cooperated well in the progress of our enhancement feature report of the Apollo autonomous driving platform.

Conclusion

As proposed, our enhancement feature to the Apollo autonomous driving system includes the function of detection and response to special road conditions, specifically the special vehicle avoidance, special pavement conditions handling, and congested road avoidance. After the enhancement feature we proposed to the Apollo system, the vehicle with such an enhanced Apollo system should be able to detect the possibility of lane change from the adjacent lane in front of the car and actively initiate and execute the lane change control request to give way to the car behind and it will pre-judge whether the car can drive through the irregular road surface to start the intelligent suspension, safety brake, safety steering, and other ways in advance. Also, the vehicle can get the real-time congestion of the road from the cloud and then calculate the optimal route. Based on our enhancement feature, the 3 NFRs we have concluded are accuracy, performance, and reliability. The stakeholders, including users, investors, program developers, and emergency service workers, will benefit directly from the NFRs we have improved and concluded based on the implementations. From this experience of adding enhancement features to the Apollo system, our team has better knowledge and understanding of how huge projects like Apollo works in general, these precious experiences on how to interpret this kind of project will benefit us in the future as well as expand our horizon on the software architecture.

Glossary

Conceptual Architecture: the developers' view of the software architecture
Concrete Architecture: Implementation of the specific architecture with the decomposition into specific components and identification of actual relationships
Apollo Program: The open-source autonomous driving platform developed by Baidu.
API: Application programming interface, which is a software intermediary that allows two applications to talk to each other.

Modules: Important subsystems within the software, each subsystem has one or more specific functions. Modules play a key part in software architecture.
GPS: Global position system that is used to track location and generate navigation.
LiDAR: Light Detection And Ranging Sensor, determines the distance by pointing a laser at an object and measuring the time it takes for the reflected light to return to the receiver.
Pipe and filter: an architectural style, made up of a number of pipes (connections) and filters (module).
Pub-Sub: an architectural style, which relies on a broadcaster that can receive the messages from the publishers, and send the messages to the subscriber who is waiting for a specific message. subscribers don't know who sends the messages but only receive them.
RTK: Real-time Kinematic, which is stored in the perception module.
HMI: Human-Machine Interface is a user interface that allows users to interact with the machine.
Subsystem: It is the sub-component of its higher-level system.
SAAM: Software Architecture Analysis Method, a four-step method for analyzing software architectures.
NFR: Non-Functional-Requirements, for instance performance, accuracy usability, etc for a system or software implementation.

References

[1] Wang, Y., Jiang, S., Lin, W., Cao, Y., Lin, L., Hu, J., ... & Luo, Q. (2020). A learning-based tune-free control framework for large scale autonomous driving system deployment. arXiv preprint arXiv:2011.04250.

[2] Behere, S., & Törngren, M. (2016). A functional reference architecture for autonomous driving. Information and Software Technology, 73, 136–150. doi:10.1016/j.infsof.2015.12.008

[3] GitHub. 2022. GitHub - ApolloAuto/apollo: An open autonomous driving platform. [online] Available at: <https://github.com/ApolloAuto/apollo> [Accessed 20 February 2022].

[4] Leigh, B. and Duwe, R., 2019. Software Architecture of Autonomous Vehicles. ATZ Electronics worldwide, 14(9), pp.48-51.

[5] GitHub. 2022. apollo/README.md at master · ApolloAuto/apollo. [online] Available at: <https://github.com/ApolloAuto/apollo/blob/master/modules/perception/README.md> [Accessed 20 February 2022].