CISC 322/326 Assignment 3
Enhancement Proposal of Apollo

April 11th, 2022

Group #7: ArchiTiger
Poppy Li 20181706 19xl12@queensu.ca
Xuan Xiong 20147035 18xx15@queensu.ca
Yuen Zhou 20186821 19yz57@queensu.ca
Yingjie Gong 20144264 18yg24@queensu.ca
Zhimu Wang 20190758 19zw28@queensu.ca
Baisheng Zhang 20094496 17bz15@queensu.ca

Table of Contents

1.0 Abstract **2**

2.0 Introduction and Overview **2**

3.0 Proposed Enhancement **2**
3.0.1 Description & Value and Benefit 2
3.0.2 Current States 2

4.0 Implementation 1 **3**
4.0.1 Conceptual Architecture for Implementation 3
4.0.2 Changes to support 4
4.0.3 High-level and low-level interactions 4
4.0.4 Impact on current directories/files 6

5.0 Implementation 2 **6**
5.0.1 Conceptual Architecture 7
5.0.2 Changes to support 7
5.0.3 High-level and low-level interactions 8
5.0.4 Impact on current directories/files 8

6.0 SAAM **9**
6.0.1 Major stakeholders 9
6.0.2 the most important NFRs for each stakeholder 9
6.0.3 impact of implementations of each identified NFR and stakeholder 10
6.0.4 compare to get the best one 11

7.0 Use Cases **12**
7.0.1 Use Case 1 Remote Control By Traffic Police 12
7.0.2 Use Case 2 13

8.0 Plan for testing **13**

9.0 Potential Risk

10.0 Concurrency **14**

11.0 Lesson Learnt **14**

12.0 Limitations **14**

13.0 Name Conventions **15**

14.0 Conclusions **15**

15.0 References **15**

1.0 Abstract

This report aims to propose emergency control as a possible enhancement for the Apollo autonomous driving platform. The following report demonstrates a new feature that our group designed for the Apollo platform called emergency control. This report will analyze two different implementations of the proposed feature to Apollo. We will go through the SAAM analysis method by defining the NFRs of the feature, comparing the two implementations concerning the NFRs and stakeholders, and selecting the first one to be the better one. This new feature will complement Apollo's functionality, making autonomous driving as close to a human driver as possible.

2.0 Introduction and Overview

A clear understanding of Apollo's architecture and functionality has been achieved through the past two assignments. The upcoming report demonstrates a new feature that our group designed for the Apollo platform called Emergency Control. We divide the report into four major parts.

In the first part, we began our design process by establishing two implementations to implement the enhancement. The first one needs to introduce a new module named "Remote Control," while the second one does not. We analyzed both the high-level and low-level systems the feature would impact the Apollo platform. In the second part, we performed a SAAM analysis on both implementations, which identified the stakeholders and the advantages and disadvantages of each implementation. The SAAM analysis revealed that the former is the better choice. In the third part, knowing the better option, we explain the chosen enhancement through two use cases and sequence diagrams. Finally, We identified various test cases to ensure our implementation is sound within the Apollo platform. The potential risks, lessons learned as well as limitations were discussed correspondingly.

3.0 Proposed Enhancement

3.0.1 Description & Value and Benefit

As a group, we realized that the Apollo platform does not have the function to respond to emergencies. For example, there is a scenario where the ambulance or fire truck needs to pass quickly or if a traffic light fails and the police need to direct traffic. In this circumstance, the Apollo self-driving car has to be capable of controlling the emergency. With the introduction of emergency control, Apollo autonomous cars can have the ability to avoid special vehicles. Within the feature, the traffic police can send direct commands using their portable devices to send specific commands such as Wait, change lane, and U-turn. Another way is Automatic avoidance, which received vague instructions from emergency vehicles, and told all surrounding Apollo to plan their route to avoid.

3.0.2 Current States

Since we added remote control as a feature in Apollo, the current conceptual architecture has changed slightly. These changes will affect the interaction and communication

between the Planning module, the HMI, and the Common module. The original Planning module will be gradually implemented under the supervision of the guardian and other modules after receiving and processing the information of each module. However, after the appearance of the Remote Control module, the Planning will be directly or indirectly affected or changed by external factors, and the new module is still under the security control of the system. Therefore, it belongs to a safe and reasonable expansion function module. Figure 1 in 4.0 shows the new conceptual architecture after adding the Remote Control.

4.0 Implementation 1

Remote Control is an effective way to realize the temporary control transfer between the Apollo AI and Traffic Police or some specific emergent vehicles. For example, traffic police can use their portable devices to send commands (e.g., Wait, change lane, U-turn) to Apollo cars through simple clicks or swipes. After receiving the commands, the Apollo system will pause the current driving plan and execute the commands sent by the traffic police first.

4.0.1 Conceptual Architecture for Implementation

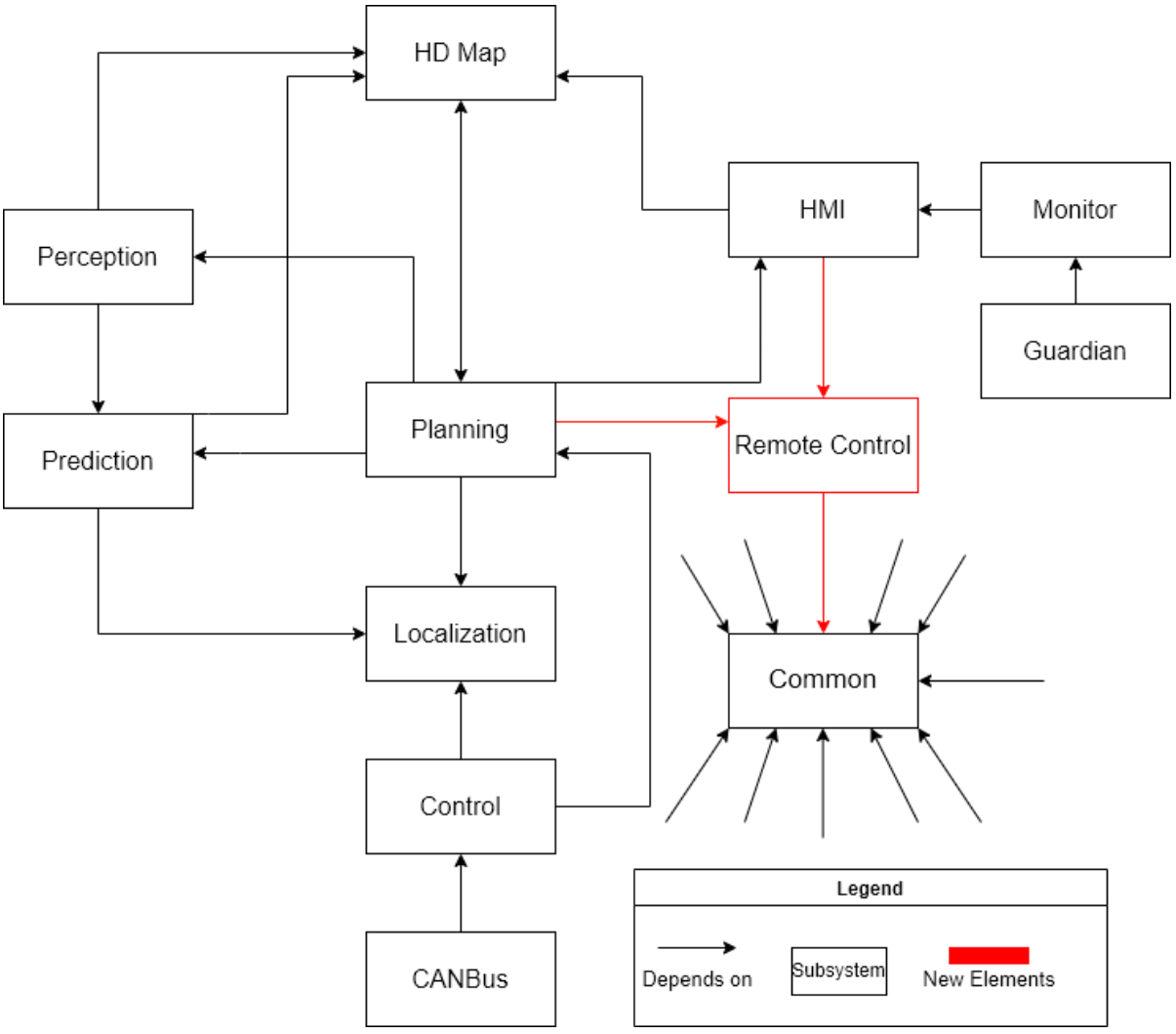

Figure 1. Updated Conceptual Architecture for Implementation 1

4.0.2 Changes to support

Remote Control: This is a new module we added to the current software architecture. It contains three sub-modules which are Receiver, Decoder, and Commander. The responsibility of this module is to receive signals sent from a terminal, decode them into commands and finally send the command to the Planning module.

Decode rule: This rule in the Common module will support the decoding of signals.

Adapter for Remote Control: This should be added to the Common module to support the new software and hardware related to Remote Control working well in the current system.

HMI Display: The HMI should notify the car owner now that the car is changing the driving plan due to the Intervene from traffic police (police car, ambulance…).

4.0.3 High-level and low-level interactions

High-level interactions:

As the graph of conceptual architecture shows above, there is a new dependency arrow starting from Planning to Remote control. This dependency is due to the Planning module requiring the remote control module to get commands sent from the terminals held by the third party like traffic police to plan for a new route. Moreover, the remote control module also notifies the Planning module when to continue the paused travel plan.

The interaction between the Common module and Remote Control module:

1. As an implementation based on the publish-subscribe architecture style, the realization of communication between each independent module is supported by the Common module. The Remote control module is a new component added to the current architecture, so there is no doubt it will also depend on the Common module to publish or subscribe to new messages.
2. The Common module provides the functionality support for the Remote Control module and makes sure it can be compatible with other current features.

The interaction between HMI and the Remote control module is relatively simple. HMI needs the commands decoded by the Remote control module and shows them on the screen to notify the car owner of what happened.

Low-level interactions:

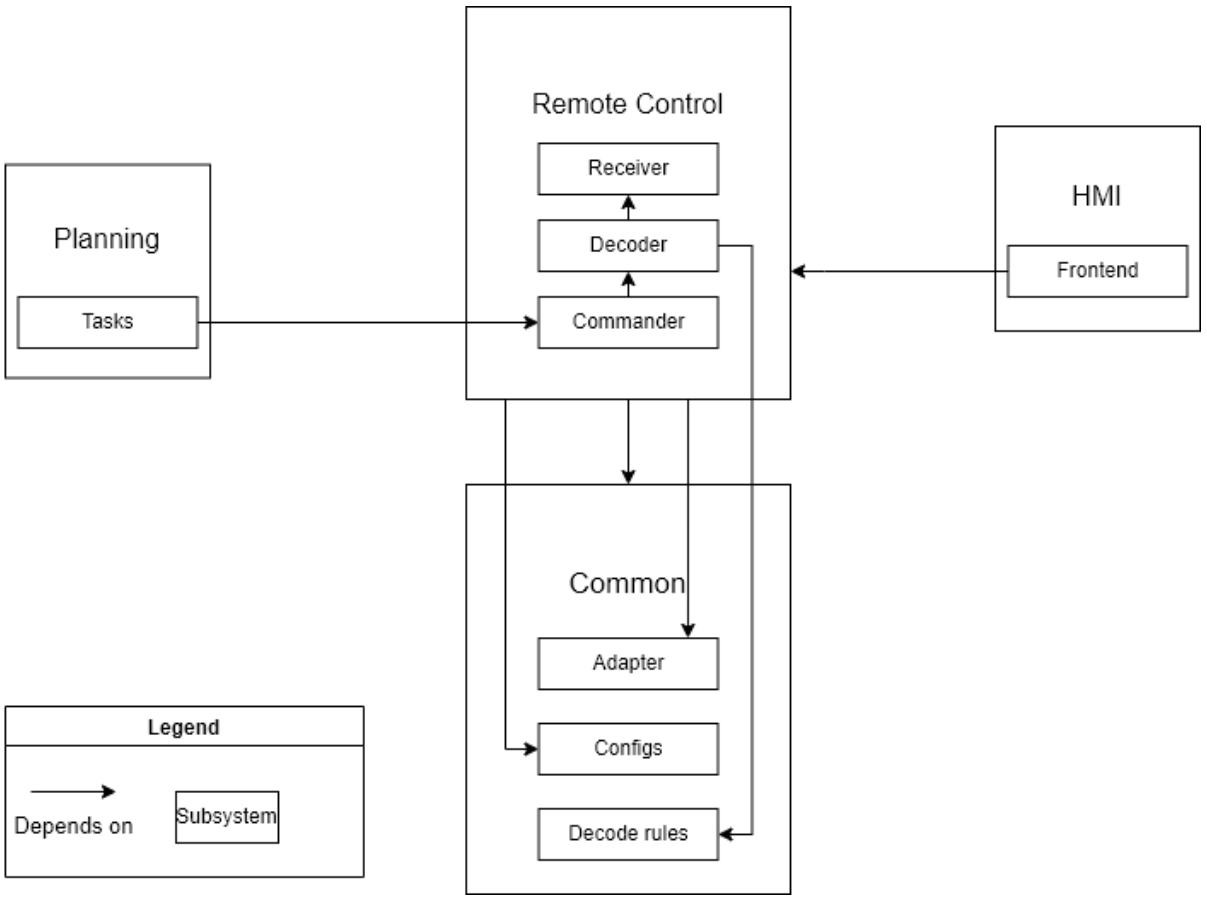

Figure 2. Low-Level Interactions of Implementation 1

Commander => Decoder => Receiver

The Receiver module receives the signals representing commands from the intervenors, and then the signals will be sent to the Decoder module. After being decoded by the Decoder, Commander would save the results.

Planning/Tasks => Remote control/Commander

The commands saved will be sent out to the Tasks module in Planning by the Commander module in Remote control.

HMI/frontend => Remote control

When an intervention happens, the components in the frontend module of HMI will require Remote control to provide detailed information to display on the screen.

Remote control/Decoder => Common/Decode rules

The Decoder module depends on the rules and math methods written in the Common module to decode signals and transfer them to readable commands, which the Planning module can process.

Remote control => Common/Adapters and Configs

A new component like the Remote Control module relies on the adapter and configs in the Common module to become compatible with current features in the software architecture.

4.0.4 Impact on current directories/files

New directories/files:

modules/remote_control/receiver

modules/remote_control/decoder

modules/remote_control/commander

modules/common/math/decode_rules

All these 4 directories above will contain new files to support related functions.

Modified directories/files:

modules/planning/tasks

modules/dreamview/frontend

Modifying some files in these two directories to apply the enhancement.

modules/common/adapters/adapter_gflags.h

modules/common/adapters/adapter_gflags.cc

modules/common/configs/config_gflags.h

modules/common/configs/config_gflags.cc

5.0 Implementation 2

The enhancement can also be achieved without introducing a new module in the architecture. Instead, new subsystems can be added to the original modules to adapt to the new change. Compared to Implementation 1, this option gives more independence to Apollo cars in that the actual control is happening within the car rather than external commands. When encountering emergencies, the newly introduced sub-components help Apollo cars to identify the emergency and act accordingly.

5.0.1 Conceptual Architecture

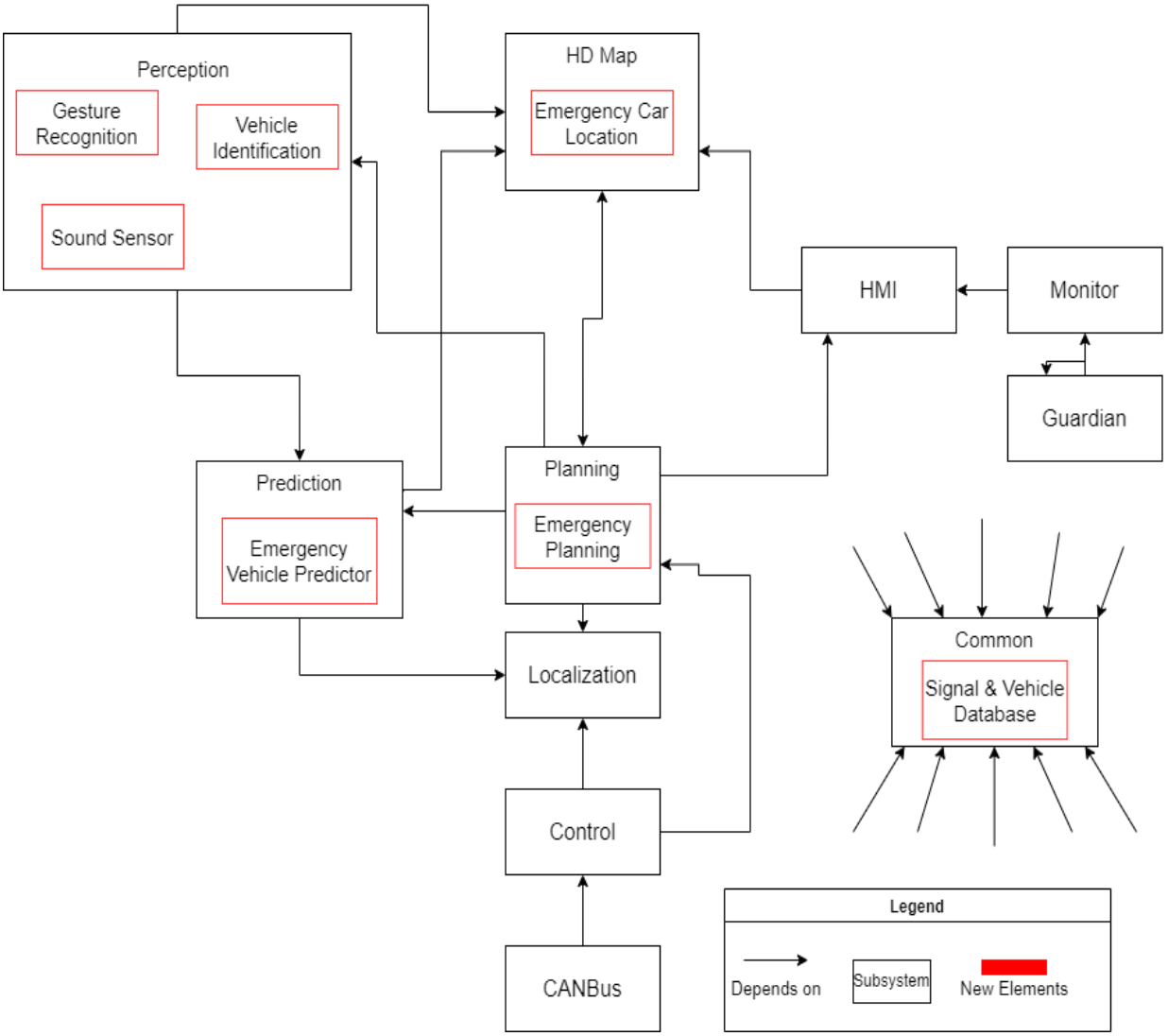

Figure 3. Updated Conceptual Architecture for Implementation 2

5.0.2 Changes to support

Gesture Recognition: This component is introduced under the Perception module to enable Apollo cars to recognize traffic police's hand signals in real-time during emergencies, such as letting an ambulance go through a traffic jam.

Vehicle Identification: It is added to allow Apollo cars to identify what type of emergency vehicles are around, whether an ambulance, a police car, or a fire truck. Vehicle identification is essential as different vehicles may lead to different planning strategies.

Sound Sensor: It is responsible for identifying surrounding sirens coming from any emergency vehicles and notifying the inner system of emergency.

Emergency Vehicle Predictor: This component is added inside the Prediction module to predict emergency vehicles' locations with the help of HDMap and Localization modules.

Emergency Planning: This component is introduced in the Planning module to refine the planning strategy further, particularly for emergencies.

Signals and Vehicles Database: In the Common module, a new database has been created with images and gesture signals of various emergency vehicles to support the perception module for more accurate identification of emergency vehicles and signals.

Emergency Car Location: This component is integrated into the HD Map module to show all the emergency vehicle's locations on the interface.

5.0.3 High-level and low-level interactions

High-level Interactions:

All the newly introduced components indicated above work closely to make the proposed enhancement possible. With everything else staying the same, our newly created emergency vehicle predictor enables the Prediction module to retrieve real-time locations of emergency vehicles and compute the relative distances between Apollo cars and them. It is achieved by using information and data from HD Map and Localization. The Prediction module gets that information through the Common module, which offers internal communications. The Perception module subscribes to the Prediction module for nearby emergency vehicles information. That is why there is a dependency from the Perception module on the Prediction module. Thanks to the added database in the Common module, the Perception module can now analyze hand signals from traffic police when every traffic participant needs to obey the command of the traffic police. In addition, it also helps situations when fire trucks and ambulances ask to make way. It returns what type the nearby emergency vehicle is based on appearance and sound sensor. The Planning module subscribes to both Perception and Prediction modules for emergency vehicle type, traffic police hand signals, and relative distance. It generates a suitable plan and sends it to the Control module with those parameters. Overall, the architecture style remains a publish-subscribe style as exchanging messages between publishers and subscribers is essential in autonomous driving.

Low-level Interactions:

For scenarios where the traffic lights do not work well, or some workarounds are required, the traffic police are present and guide the whole traffic. In this case, the gesture recognition in the Perception module will identify what signal the police are giving and make the car respond. The new database helps the recognition process in the Common module. This database contains all hand signals for more accurate recognition. The emergency planning component in the Planning module subscribes to the gesture recognition event for necessary information and computes the corresponding plan.

Another scenario is that sometimes, for example, emergency vehicles want to find a way in a traffic jam. The emergency vehicle predictor gets the locations of emergency vehicles with the help of HD Map and Localization. Subsequently, the vehicle identification component and sound sensor in the Perception module receive the output from the emergency vehicle predictor and act based on it. These two components help analyze the emergency vehicle and determine if it is on a mission based on whether it turns on the siren. Similar to the first scenario, the emergency planning component in the Planning module subscribes to all of them to make reasonable plans. The control module subscribes to the emergency planning module to get the chosen plan and make the car move autonomously.

5.0.4 Impact on current directories/files

Due to the newly introduced subsystems, they will be added to several old modules:

modules/perception/gesture_recognition

modules/perception/vehicle_identification

modules/perception/sound_sensor

modules/prediction/emergency_vehicle_predictor

modules/hdmap/emergency_car_location

modules/planning/emergency_planning

modules/common/signal_vehicle_database

The adapter and configuration files should also be updated to adapt the new architecture:

modules/common/adapters/adapter_gflags.h

modules/common/adapters/adapter_gflags.cc

modules/common/configs/config_gflags.h

modules/common/configs/config_gflags.cc

6.0 SAAM

6.0.1 Major stakeholders

The major stakeholders for these two different enhancements include users, developers, government agencies like traffic police or ambulance, and the public.

Table 1. Major Stakeholders and Corresponding NFRs

Skateholders	NFRs
Users	Usability, Safety, Privacy
Developers	Testability, Maintainability, Scalability
Governemnt agencies	Accessibility, Privacy, Security
Public	Safety, Privacy

6.0.2 the most important NFRs for each stakeholder

Users:

Usability: The easiness for the user to implement the system themself

Safety: The software needs to ensure the safety of the user in a certain way

Privacy: The ability to protect user privacy and prevent illegal surveillance

Developers:

Testability: the ability allow developers to run an experiment to test a hypothesis, theory or even functionality

Maintainability: Ability to run all components for automated testing and overnight

testing

Scalability: the highest workloads will still meet the performance requirements, reducing the workload of developers

Government agencies:

Accessibility: the ability to ensure that government agencies like the traffic police can send commands to the Apollo car

Privacy: The ability to protect government agencies' privacy and prevent illegal surveillance

Security: The software needs to protect the accuracy and security of data to avoid malicious acquisition of private information

Public:

Safety: The ability to ensure the safety of pedestrians and does not cause damage to the public property

Privacy: The ability to ensure the security of data collected on road conditions and public information

6.0.3 impact of implementations of each identified NFR and stakeholder

For users, its NFRs include usability, safety, and privacy. The first is usability, which users most directly feel in terms of driving experience, road planning, and human-machine interaction. The word-of-mouth effect can be easily formed among consumers. As for safety, the two enhancements mentioned above can provide help when the safety of other people's lives and property are in danger, but also when they encounter driving problems or traffic safety accidents, they can have certain emergency handling capabilities or anticipate the possibility of accidents in advance to minimize the risk of travel. At the same time, the most important thing is that customers are most concerned about personal safety and privacy. The system helps drivers get a good driving experience simultaneously and needs to learn its driving habits, etc. Still, it needs to grasp the degree of learning and observation, which involves social, moral, and ethical issues. The system will continue to combine the facts to update the system permissions and leave the user the ability to customize permissions.

There are three NFRs for developers: testability, maintainability, and scalability, and for testability, many interfaces are reserved to facilitate extensive testing during later upgrades. At the same time, each module also has independent operation capability, so it is easy to conduct a series of joint operation experiments or individual module performance tests according to the developer's needs. For maintainability, which is quite understandable, similar to the modular design of hardware, our system is also designed in a modular way, extremely easy to understand, and very friendly to post-maintenance. With the corresponding software, the user can implement the monitoring of individual modules. Finally, scalability, as the previous testability said, a large number of interfaces reserved not only to facilitate testing but also to ensure the safety and reliability of the

case, some functional expansion, and reserved interfaces also indicate a full grasp of the system load and confidence.

For government agencies, the first thing that must be done is that such state agencies as government public security should have absolute control over the ride and have the ability to issue command messages to Apollo cars. There also needs to be geographical monitoring, and certain detection features should be deactivated or removed immediately when entering government units or sensitive areas. If the request exceeds the functional scope of Apollo cars, the car will be directed to leave the area. Furthermore, compared to the privacy of general users, the privacy management of government agencies needs to be more strictly controlled and monitored. A separate closed-loop database can be created for easy management if necessary.

For the public, the Apollo system ensures pedestrian safety. It minimizes damage to public property, but with its detection module, it can also proactively file a series of accidents to determine the innocence of everyone involved. It also ensures the security of the road and public information data collected and can immediately contact the relevant authorities for reporting an emergency.

6.0.4 compare to get the best one

We select implementation 1 as the proposed enhancement for these ample reasons, then implementation 2 as the alternative approach. Therefore, for the remainder of the paper, we will take implementation 1 as consideration.

Compared to implementation 2, implementation 1 is more conducive to higher control, for example, in the event of criminal cases or dangerous gas leaks around sensitive areas by the state agency Apollo system management rights, to carry out the driving path update, to avoid possible risks, or to make way for the treatment unit, the maximum possible protection of people's lives and property Safety. At the same time, in-state agencies such as governmental areas and the higher-level units should control and close the detection and analysis functions related to Apollo cars to avoid the leakage of privacy and secrets.

At the same time, compared with implementation 2, implementation 1 only adds one module, which is more conducive to later maintenance and repair. In addition, it is more convenient and quicker than other methods because it uses a modular approach to add new functions.

On the other hand, implementation 1 provides better safety. Every time a police car, fire engine, and ambulance are dispatched, we can say that it is a particular and urgent situation, and in this case, any mistake is not allowed. If the Apollo car only uses the interaction between its modules, errors will likely occur. For example, a police car or an ambulance being wrongly judged will cause sudden avoidance of the lane, which may affect the user's safety. However, this kind of problem can be avoided very well by government agency personnel to control the way of Apollo cars.

Finally, implementation 2 has higher requirements for Apollo's perception ability. When the ambulance or fire engine is dispatched, their speed is relatively fast. If the Apollo car cannot sense and recognize them in time, it may delay the task of the ambulance or fire engine. Although there is a way to solve this situation that cannot be sensed and recognized in time, it needs to spend much money to improve the perception ability of the

Perception module, and the hardware requirements will be much higher. For implementation 1, the Apollo car Perception module only needs to sense nearby vehicles and make changes.

In conclusion, implementation 1 guarantees higher security and can also be managed by government agencies, thus protecting the privacy of users and regions. Not only that, implementation 1 does not need to increase the spending of its sensor in the Perception module, which indirectly saves much money. Eventually, compared with implementation 2, implementation 1 only adds one module, which is beneficial to the later maintenance work. For these combined reasons, we choose implementation 1 as the proposed enhancement.

7.0 Use Cases

7.0.1 Use Case 1 Remote Control By Traffic Police

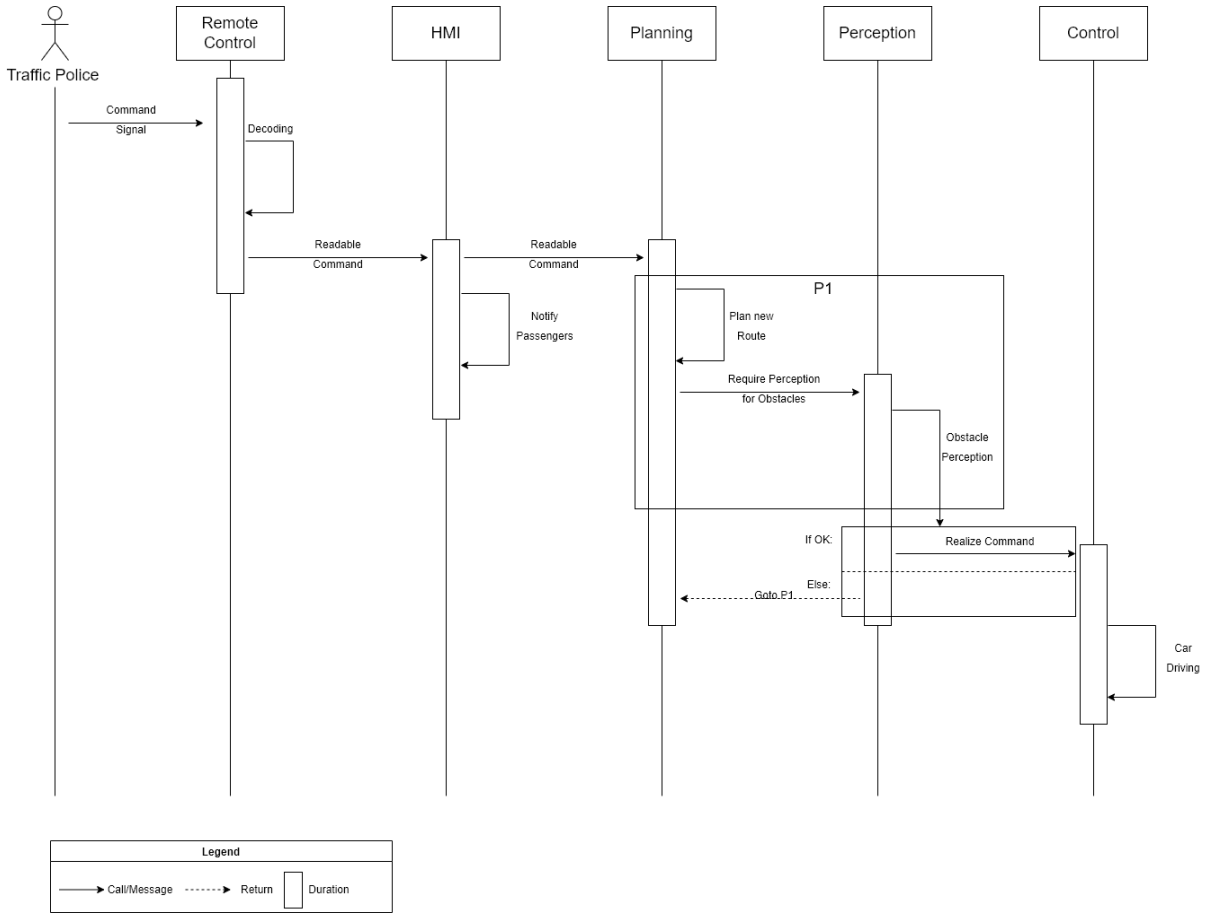

Figure 4. Sequence Diagram for Use Case 1:Remote Control By Traffic Police

This Use case describes interactions between the components in our architecture for implementation 1 when traffic police take over the Apollo. The Remote Control module first receives the order from the traffic police, which is in the form of a remote signal, then the decoder inside decodes the signal into a readable command. HMI will display the command on the one hand. On the other hand, it will be sent to the Planning module for regulating the

exact route to realize the command from the traffic police. After the cooperation between the Planning and Perception modules, the Control module will execute the command in a safe route.

7.0.2 Use Case 2

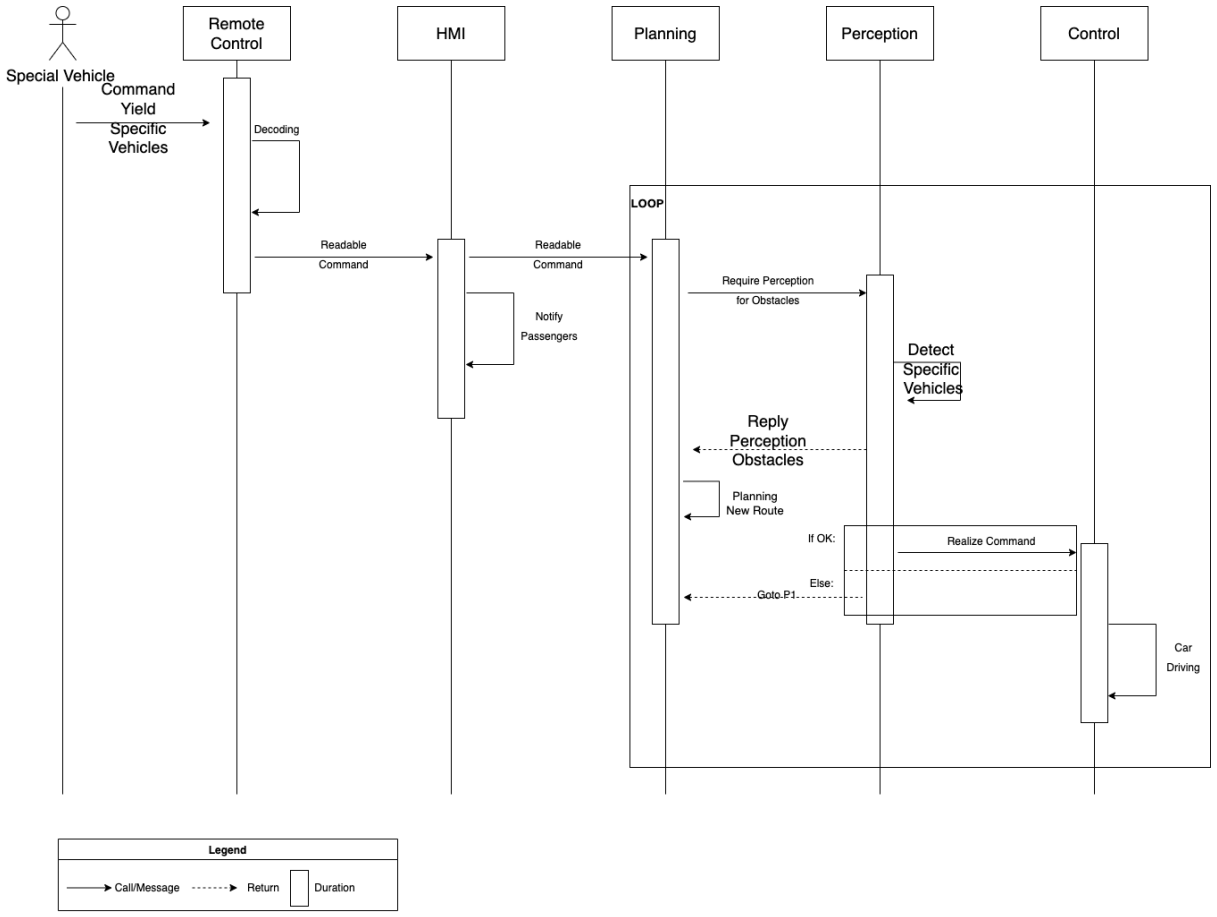

Figure 5. Sequence Diagram for Use Case 2: Remote control through special vehicle

This use case describes that when a special vehicle is in charge of Apollo, the interaction between modules in the architecture is used to implement 1. The remote control module will first receive an instruction from the special vehicle, which is used to tell Apollo that the car will evade. This instruction is based on a remote control signal. The internal decoder then decodes the signal into readable instructions. The HMI module will receive the instruction and notify the passengers simultaneously. On the other hand, the HMI will also send this instruction to the Planning module to adjust the exact route. The Planning, Perception, and Control modules cooperate, thus choosing a safe route to avoid.

8.0 Plan for testing

To design practical tests for our proposal and its interaction with the Apollo architecture, we require a set of tests satisfying usability, safety, testability, maintainability, scalability, and privacy. We will start with two tests to test the two enhancements proposed in this paper more concretely and comprehensively. The first will test the car for usability, safety,

and privacy while the tester is in the remote control state. Under developer tuning tests, the second group will focus on the car's testability, maintainability, and scalability. Some simple examples of some test cases are

1. Test whether the software and hardware in the Apollo car are always available under different road conditions and speeds. Besides, whether there are faults that lead to dangerous situations;
2. After completing the test, check whether the car can automatically delete the sensitive information recorded before;
3. Detect the new hardware and software loaded on the original hardware, check the compatibility and test results;
4. Detect whether the maintenance time is faster than other car companies of the same level when there is a problem with a simulated module.
   These tests must consider all systems and subsystems that are significantly affected, and we will decide whether to test the entire system or one of the subsystems as needed. Each test we create will be saved to the corresponding suite or storage area so that future developers can observe and learn from it.

9.0 Potential Risk

The implementation of our feature might come with potential security risks. First, the emergencies we discussed, including the dispatch of fire engines or the dispatch ambulances, do not allow for any error. Since we are giving total control to the traffic police, this behavior is unknown. Since there might be misoperation or improper planning, this would increase crashing and take unthinkable consequences. Second, the traffic police first have to send commands. Then the system will make adjustments, pause the driving plan and execute the commands, resulting in slower response times.

10. 0 Concurrency

Apollo has a scheduling system that focuses on real-time performance, allowing Apollo to be concurrent and multi-threaded, such that it prevents sudden remote control signals from having an impact on system processes;

Apollo's mode of action comes from multiple interacting processes with concurrency between processes. Such a feature improves the performance of the perception module and allows Apollo to handle the detection of obstacles well;

The concurrency consists of 5 nodes (LiDAR, radar, fusion, traffic light preprocessing, and traffic light processes). Each node can be considered as a thread. These threads are connected to each other to influence each other. Even after the intervention of the remote control module, there is still an efficient and rapid response and feedback.

11.0 Lesson Learnt

Within completing this project, we were able to look at the entire Apollo autonomous driving system from a more objective perspective. As a result, we offered our insights into improving the whole system. We also learned the importance of privacy and security when perfecting a technique with user information. In addition, when we used SAAM, we also gained a deeper

understanding of this method of architectural analysis, and we discovered that this analysis method is beneficial to assess the risks inherent in the architecture.

12.0 Limitations

In our report, our enhancements are mainly focused on: 1) Creating a new type of remote control module for the Apollo autopilot system and 2) Updating the emergency program that comes with the Apollo autopilot system itself. However, these two updates also bring some limitations and problems. For the first update, the appearance of the remote control module will reduce the user's sense of security for apollo autonomous driving. However, the remote control power we propose is all in the government, which requires the appearance of the entire module to have the highest safety and security specifications (Baidu, 2022). Privacy to protect the whole system from being invaded by the government and the security of user information. However, a security system of this scale is challenging to achieve. For the second update, we require that the emergency module of the Apollo autopilot system can respond correctly to traffic police gestures, emergency vehicle signals, sirens, etc. This requires adding many sensors and requiring the emergency module to have a certain degree of learning. The ability to make relatively correct responses to some ambiguous situations requires a very complex program to achieve.

13.0 Name Conventions

SAAM (Software Architecture Analysis Method) is a method for Analyzing the Properties of Software Architectures

NFRs (Nonfunctional Requirements) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the system's design across the different backlogs.

14.0 Conclusions

This report proposes two aspects that should be enhanced for Apollo autonomous driving. The first is the remote control. When an emergency occurs, relevant agencies such as the government will have the ability to control the Apollo system remotely. This essentially solves the problem that when Apollo autonomous driving encounters a situation that the computer cannot judge, it allows relevant departments to propose a quick solution. In addition, it can also play an essential role in chasing criminals and other related government activities. Then we proposed to upgrade the emergency module, which is also very important. This upgrade mainly solves some small situations. The computer can quickly respond like a natural person, such as the command of the traffic police, the whistle of the emergency vehicle, etc. This improvement will allow Apollo's autonomous driving to respond quickly and correctly. After proposing these two enhancements, we use SAAM to assess the risks inherent in the architecture. Furthermore, give some insights into its privacy and security.

15.0 References

Baidu. (2022, February 7). Building a self-driving car that People Can Trust. MIT Technology Review. Retrieved April 11, 2022, from https://www.technologyreview.com/2020/12/16/1014672/building-a-self-driving-car-that-people-can-trust/