Architecture Enhancement Report on Apollo

Minxuan Li 20144519 18ml51@queensu.ca
Li Bowen 20144417 18bl29@queensu.ca
Steven Wen 20144322 18yw85@queensu.ca
Yuehan Qi 20167259 18yq36@queensu.ca
Aolin Zhou 20058020 16az31@queensu.ca
Date 2022.04.08
Queen's University
CISC322
Instructor: Prof. Bram Adams

Table of Content

1.Abstract

In this paper, we proposed a new exciting enhancement of the summoning vehicle in Apollo. These features help to solve situations like saving drivers from entering a constricted space and being caught in the rain. To implement this enhancement, we noticed that it worked closely with almost every subsystem in Apollo, including Planning, Perception, Prediction, Control, CANbus, Localization, HD Map, Guardian, monitor, and HMI. As we added the Summon module, the Control, Guardian, and Monitor subsystems are impacted since these subsystems are the keys to communicating between driver and vehicle.

As we dive deeper, we discovered that some files and directories are impacted: lattice, planner, reference line, scenarios, and tasks. The developer may need to pay more attention to the lattice directory with its two crucial behavior and trajectory generation features. Some risks are associated with the Summon module, and the concerns are security and maintainability. We devised two ways of implementing this enhancement. One is to utilize enormous data from image recognizing systems, machine learning, and neural networks to build a system for autonomously summoning a vehicle to the destination. The other implementation is to control the driving condition throughout the preload process. The NFRs of Maintainability, Evolvability, Testability, and Performance are affected by realizing the enhancement. We proposed a plan for testing the Summon module in the system by giving simple examples of checking whether the car can stop at the destination accurately. Lastly, we identified two use cases and identified three significant stakeholders: Driver, Developers, and PMC.

2.Introduction and Overview

Apollo is an open-source, high-performance, flexible architecture developed by Baidu to provide a software platform for self-driving cars focusing on portability, flexibility, testability, reliability, maintainability, and manageability. This project was originally launched in 2017; the Baidu Apollo fleet of approximately 500 self-driving cars is known for its reliability and track record. In 2021, Baidu's Apollo Go provided 115,000 rides and Apollo L4 autonomous driving racked up more than 10 million test miles. As the driving technology matures, Baidu Apollo has become the leading autonomous driving technology for China's autonomous industry.

After analyzing the conceptual architecture and specific architecture of Apollo Autonomous Driving, we propose an improvement to Smart Summon. It relies heavily on the

autonomous driving capabilities of the Apollo architecture to independently perform safe movement tasks to target locations.

This report will first describe the enhancement and discuss the impact aspects of the overall system. Two use cases and sequence diagrams are also included in this paper. As our proposed enhancement may impact some NFRs, we found that maintainability, evolvability, testability, and performance may change. We also analyzed the non-functional requirements and stakeholders involved in the whole process. We not only suggested improvements but also provided potential risks and test scenarios for this improvement. All of these were done to propose improvements and accomplish better functionality.

3.Enhancement

3.1. Summon Enhancement

After a thoughtful discussion and consideration, our team came up with a facilitating enhancement in the Apollo autonomous driving system. The idea is to utilize a key fob or phone app to remotely control the vehicle to drive to the target location autonomously.

The driver needs to press a button on the key fob for three seconds to enter a summon mode; the vehicle will be ignited, and its hazard-lights flicker for a few seconds and turned off. During summon mode, for safety purposes, the driver takes complete control of the vehicle by its key or app, and the driver has three options to choose, forward, reverse, or shut down summon mode. The vehicle can only be moved when the driver is constantly holding a button of either forward or reverse, and it detects no obstacles are within the path or no obstacles are coming into the path. The vehicle moves to the target location based on the planned route in the forward option. The vehicle moves back to the original position in the reverse option based on the planned route. It is the driver's responsibility to sight the trajectory path of the vehicle and make sure everything is under control. Therefore, its general use case should be within a private parking area, garage, or driveway. Its moving speed is maintained at no more than 10 kilometers per hour for the sake of safety.

In reality, some situations may occur, such as a thunderstorm and the users just finished shopping at a supermarket. A vehicle summoning enhancement in Apollo could save the users from being caught in the rain. On the other hand, for instance, if the users' car is parked in a narrow space between two cars, this enhancement is also beneficial for saving the driver from squeezing into such a constricted area to get into the vehicle. Moreover, this enhancement could be partnered with home assistant AI-powered applications in-home

devices such as Google Assistant. As soon as the vehicle leaves or enters the garage, it automatically closes the garage door and shuts down or turns off the garage light, making everything more straightforward.

3.2.Interactions with components in the system

With this new enhancement added, there is not much difference in how the vehicle operates autonomously, except certain constraints are imposed, such that the driver is driving the vehicle using either the forward or reverse button on the phone application or key fob. There is also a constraint of speed limitation of 10 kilometers per hour to ensure safety. Upon pressing a key to enter the summon mode, the monitor module receives a signal from HMI, and the control module which subscribed to the monitor starts the engine. Many other monitor subscribers begin to operate normally to ensure the vehicle moves to the destination safely and reliably. These modules are planning, prediction, perception, HD Map, Localization, CANbus, and control. They are working essentially the same as autonomous driving on the road, except that the perception module needs to be more careful with pedestrians, and a speed limit of 10 kilometers per hour is in force.

In this case, the monitor modules have to constantly subscribe to the message of HMI, which is a phone app or key fob. When the forwarding button is pressed, the vehicle enters a summon mode and proceeds to the destination; however, the vehicle comes to a complete stop when the forwarding button is released. Therefore, a Guardian module may have to constantly subscribe to the CANbus to receive messages from the monitor module. Once detected, the forward message is no longer sent, so stop the car immediately.

4.Current state of system

As we added smart summon as a feature in the Apollo, there are changes to the current conceptual architecture. The changes will affect the communication between the new summon module and the control module. After discussion and SAAM analysis, the new subsystem is added to the conceptual architecture. The Figure 1 below shows the new conceptual architecture after the summon was added.

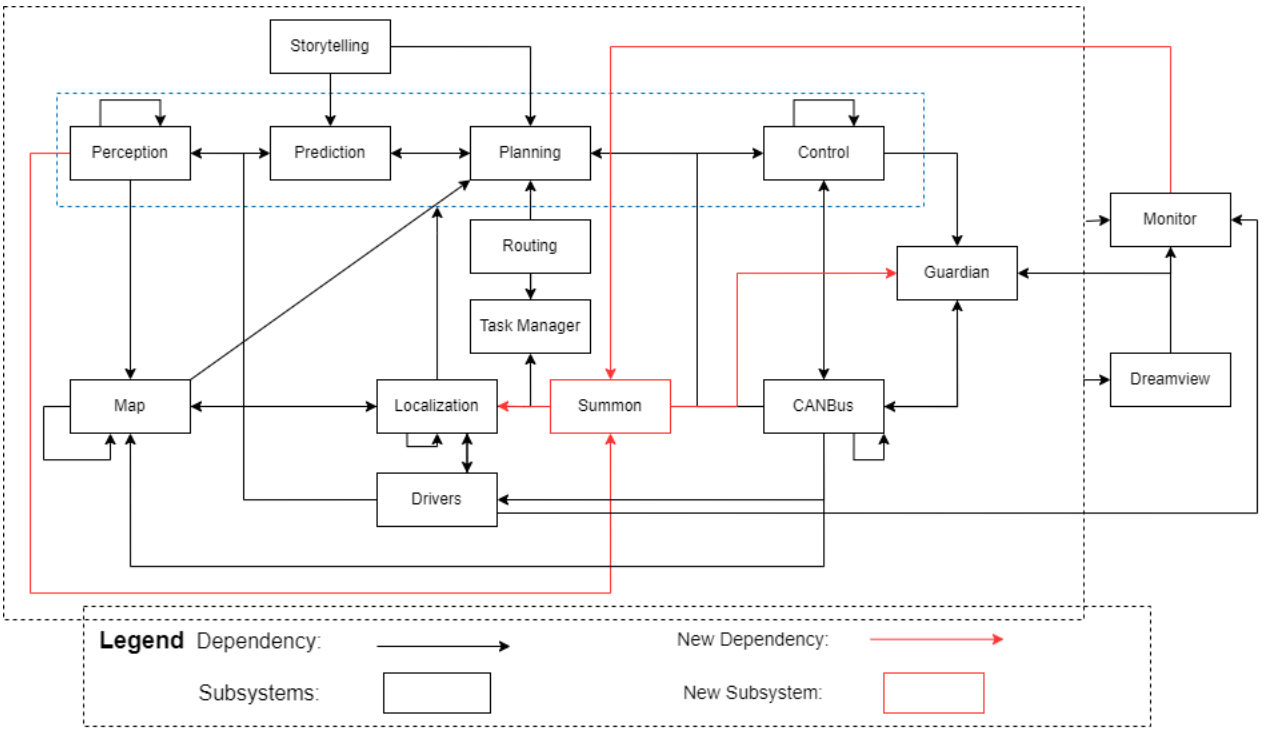

Figure 1 : Current state of the conceptual architecture with summon implemented

5.Use cases and sequence diagram

The first use case is when the driver is not in the car and wishes the car to come to the users' location. The summon module will work with the user's cell phone app when the phone's location is within 500 m of the car. After the pre-requirement is met, the GPS location will be sent to localization, and the GPS location will be constantly sent to the Localization module.

All the modules will be operated in the same manner as in the self-driving case, where the perception module will take the information from the map, driver, localization and its own camera, lidar and radar data and output the perceived obstacles to the prediction module. Where the data will be processed and then sent to the planning module and storytelling. The planning module will be subscribed to CANbus for the chassis information, localization for location estimate, map for map message, perception for stories from storytelling, routing response from routing and prediction for prediction obstacles. With all the input information, planning will output the ADC Trajectory to the control module, which has been subscribed by CANBus, where the vehicle will be controlled. The difference will be that the user will have a live camera feed of the vehicle since the user has to keep pressing the button to keep the summon working, which Guardian will subscribe to the monitor to receive system status if all the modules are working fine and if the user is pressing the button then the

Control can be sent to CANBus normally. Otherwise, Guardian will prevent Control signals from reaching CANBus and bring the car to a stop.

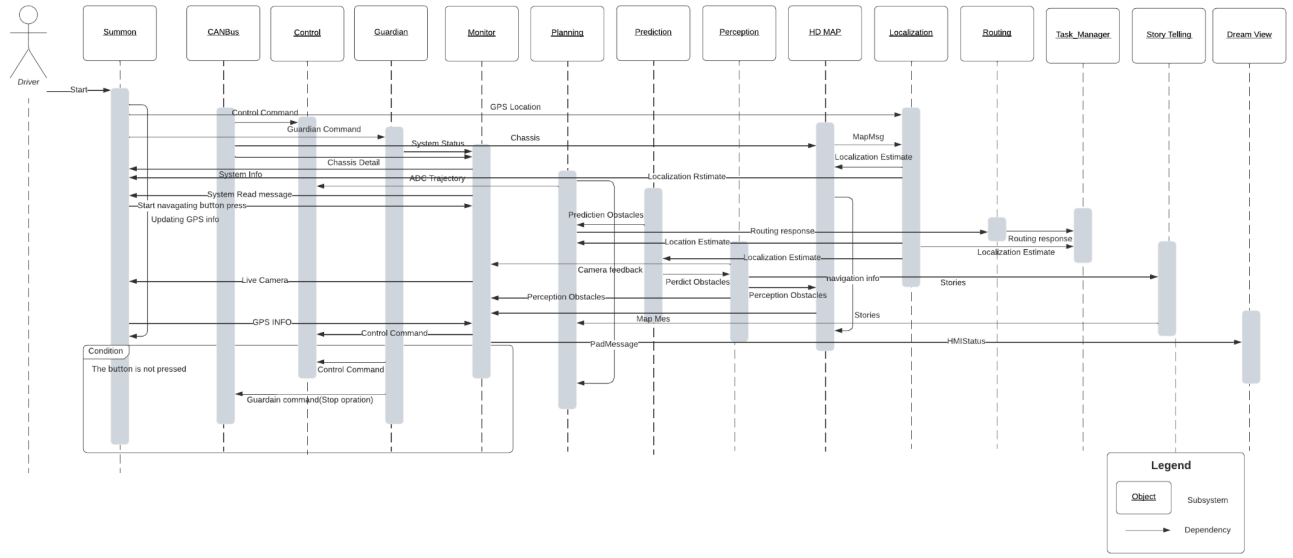

Figure 2: Use case for when summon base on the user's location

The second use case is the user providing a destination for the car. And the summon module will only work when the given location is within 500m of the vehicle. This case is similar to the first user case but the cell phone location won't be constantly updated to the Localization module. And the vehicle will stop when the car reaches the destination.

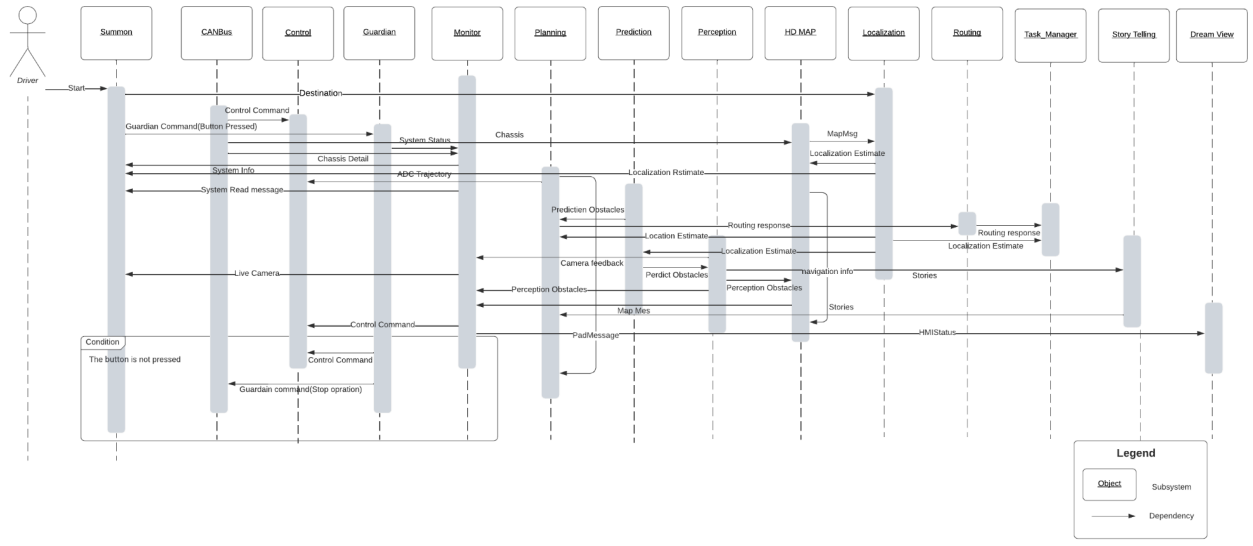

Figure 3:Use case for when summon base on the given location

6.Realizing the enhancement

- **Implementation 1:**

When the system receives the destination (GPS on the user's mobile phone or a specific location within 500m of the car), it plans the route. Then the system uses all the modules together to complete autonomous driving to the destination without driver assistance. This is our first implementation. In order to realize this implementation, the self-driving car developers use vast amounts of data from image recognition systems, machine learning, and neural networks to build systems that can drive autonomously. The neural network identifies the patterns in the data supplied to the machine learning algorithm. These data include a home-based car camera image that determines the neuron network to recognize trees, sucking, pedestrians, road signs, and other parts of a particular operating environment. The implementation can be realized in auto-driving cars that drive to the destination without driver assistance.

- **Implementation 2:**

The driver controls the driving condition of the vehicle throughout the preload process. The control of the driver app allows the driver to control the driving state of the car remotely. Remote start technology is complex from a mechanical point of view but straightforward from a user's point of view. Normally, the users need to press the button once or twice with the key fob or press the "Start" command in the smartphone app. When the driver presses the button on the remote start fob, the driver signals the vehicle's ignition system to start the vehicle. This signal activates the engine and activates the heating, cooling, or other functions previously turned on by the driver. This is the method to realize the implementation that drivers can remotely control the cars.

7.Impacted Subsystems

The Summon module works closely with many established modules. First, when receiving the GPS signals or destination entered by the user, the localization module will update the information to the task manager and then to planning and control. After calculating the route, the vehicle will automatically drive to the driver’s location utilizing all modules. The difference is that the driver will not be behind the wheels in the summon period as he will in a regular automatic driving experience. Instead, he will send commands to the car by pressing the buttons in the app through his phone.

- **Perception**

The camera in the front of the vehicle monitors the vehicle's road condition in the forward. While the summon module is instantiated, the live videos received from the camera in the perception module will be sent as a copy to the app on the user's phone to ensure the vehicle's safety.

- **Localization**

The summon module will interact with the localization module by constantly sending GPS signals or one destination to localization. It locates the user's location or the entered destination for further route planning and determines the vehicle's location in the global map for relative localization.

- **Guardian**

The guardian will update its conditional statements given the new scenario where the user will not be in the vehicle, rather, in a reasonable distance away from the car. And it will subscribe to the summon module when the user stops pressing the button then the guardian will stop the operation of the vehicle.

- **Monitor**

The monitor will send data to the summon module, which equivalently means sending information to the user's app. It is equivalent to the dreamview module except the user-interface now is compatible with the phone and will be displayed on the app. Under this circumstance, the monitor will send a copy of all the signals it sends to HMI to the app.

8.SAAM Analysis

8.1.Identify stakeholders

- Driver

The drivers who take the auto-drive cars are the main stakeholders in our implementation. They are mainly focused on the system's performance since only applicable implementation can help the driver summon the car. Meanwhile, the driver needs to be concerned with the functionality of how to control the car remotely in an efficient way. Since the drivers have direct interaction with the implementation, they are considered the most significant stakeholder.

- Developers

The developers are only an important part of stakeholders. The developers need to consider more than drivers since they investigate the success of the implementation. Firstly, the developers need to know how implementation evolves from the old version and what

areas are improved. Moreover, the developers need to pay attention to testability because they test the implementation to promise safety for the drivers. Finally, the developers are concerned with maintainability due to the fact that they need to fix further issues in the future.

- PMC (Project Management Committee)

The Project Management Committee oversees the Apollo community as the core management team. The PMC needs to be concerned with the success of the implementation as well, but they do not need to care about every detail about the implementation.

8.2.Effects of the enhancement on NFRs of the System

NFRs	Effect (Implementation 1)	Effect (Implementation 2)
Maintainability	Always need to update the route planning algorithms which is a massive workload.	Need to update the app which is used to control the cars in certain periods.
Evolvability	The system evolves a huge step since the implementation is the enhancement of previous system	The system evolves a huge step since the implementation is the enhancement of previous system
Testability	The system is easy to test.	The system is easy to test.
Performance of System	The system must plan the route in a very short period since it is unacceptable for people to wait a long time.	The system must connect the car in a wide range to make sure the auto-driving can reach a long distance place.

8.3.Comparison of Implementations

With the established stakeholders and related important NFRs, we can compare the relative advantage out of the two implementations and determine a better choice. As a result, we think implementation two will be a better approach as it satisfies NFRs and stakeholders in a better way. More importantly, as we think safety should be our priority among all the NFRs related to the enhancement, Implementation 2 has a better safety guarantee with consistent control from the driver through the app. Therefore, we think implementation two will be a better choice, and it will be the implementation considered in the remaining paper.

9.Impacted directory and file

According to our proposed enhancement for summoning a car, the implementation of this powerful feature will affect various directories and files, such as the controller file in the control and the strategy file in the routing subsystem. The most significant impacted ones are the following:

- lattice
- planner
- reference line
- scenarios
- tasks

These directories and files are significantly affected due to the summon implementation of new code within the planning system. The planning subsystem's input is localization, perception, prediction, HD map, routing, and task manager, producing a collision-free and comfortable trajectory for the control module to execute. The new enhancement means that we need to create new functionalities and then connect them with the existing functionalities of the planning module. The lattice contains behavior and trajectory generation files when we want to control the car's route when we summon it. At the same time, when we summon the car, unexpected situations and rules to follow will happen to tasks that were not available in the previous version, which will be affected by the new enhancement.

10.Potential risks

- **Security**

First and foremost, safety issues concerning vehicles and pedestrians will affect the most important issues for autonomous driving. Summoning a vehicle can be a good solution to the problem of drivers operating vehicles in tight spaces and saving time in limited time, but it also has significant safety risks. The importance of the driver's blind spot is greatly increased when maneuvering the vehicle in cramped conditions. The area immediately in front of, beside or behind the vehicle rarely poses a threat at highway speeds, but suddenly becomes home to pedestrians, pets, grocery carts and debris, often with disastrous consequences.

- **Maintainability**

Maintainability is the ease with which a system can be supported, changed, enhanced, and reorganized over time. A new enhancement proposed in this thesis affects various subsystems, resulting in a high degree of difficulty in maintainability. Moreover, each rule will lead to differences due to different users, many new possibilities need to be taken into account, and the frequency of modifications may be high, which requires a high level of maintainability of the module.

11.Plan for Testing

Throughout all phases of development, including post-development testing, the test plan will be regularly tested. We must have well-defined tests before we begin testing. Completeness, importance and stability ranking and stability, and verifiability should all be considered in these tests.

To better and more comprehensively test the enhancement proposed in this paper, we will start with two implementations. The first group will test the accuracy and safety of the car when the driver is in remote control, and the second group will focus on whether the car's route planning is efficient and safe without human control.

A few simple examples of some test cases are:

- Compare the car's actual route and the planned route, and check whether it can accurately follow the planned route.
- Check whether the car can stop at the destination accurately.
- Check whether the car can give way to other cars when it meets.
- When the driver is in control of the route, put obstacles on the planned route and check whether the car can detect the obstacle, then alert the driver when the driver overlooks the obstacles

These tests have to take into account all the systems and the planning subsystem that received significant impact, and we need to test the system as a whole. Each test that we created would be added to a regression test suite so that future developers may ensure that their changes do not affect pre-existing functionality. Using the concepts of extreme programming, in addition to testing, would require us to simplify our code at every phase, keeping the entire process orderly and allowing us to prioritize the most critical features at each step.

12.Conclusion

In general, this proposed enhancement of Summoning the vehicle can undoubtedly provide convenience for drivers. The realized way of the Summon module depends heavily on the data of image processing systems, machine learning, and neural networks. Thus, almost every subsystem is involved in building this new feature. On the other hand, there are several risks associated with it, and the main concerns are safety and maintainability. Some Control, Guardian, and Monitor subsystems are impacted with added new features. Thus the NFRs of maintainability and performance may be degraded. We figured out that the second implementation could be a better choice due to safety concerns. The priority is to ensure the driver and passenger's safety, and it would be more reliable and accurate in terms of the second implementation. Even though it comes with the cost of compromised performance and maintainability, the Summon module is an exciting and convenient feature for many users in some situations. Chances are these affected subsystems can have a way to improve in the future by developers.

13.Naming convention

PMC: Project Management Committee

HD Map: High Definition Map

NFRs: Non Functional Requirements

L4: Level 4 (High Driving Automation), vehicles can operate in self-driving mode

14.Lession Learned

With the completion of this project, we better understood the tremendous difficulty of finding viable and effective enhancements in a mature project like Apollo Auto Driving that has been updated many times. We found that differences in implementation details significantly impacted the ability to meet various non-functional requirements. Also, using SAAM analysis helps identify the proposed implementation's strengths and weaknesses and use this information to select a superior performance.

15.Reference

1) Loveday, S. (n.d.). *Testing Tesla Autopilot Summon Mode To See If It Will Detect A Small Child*. Insideevs.

https://insideevs.com/news/328942/testing-tesla-autopilot-summon-mode-to-see-if-it-will-detect-a-small-child-video/

2) *Apollo/Modules/Planning*. (2020, November 9). Github. https://github.com/ApolloAuto/apollo/tree/master/modules/planning/lattice/trajectory_generation
3) Plungis, J. (2019, October 8). *Tesla's Smart Summon Performance Doesn't Match Marketing Hype*. Consumer Reports. https://www.consumerreports.org/automotive-technology/teslas-smart-summon-performance-doesnt-match-marketing-hype/
4) Lutkevich, B. (n.d.). *Self-Driving Car (Autonomous Car or Driverless Car)*. TechTarget. https://www.techtarget.com/searchenterpriseai/definition/driverless-car
5) Choksey, J. S. (2021, October 18). *What Is Remote Engine Start in a Car?* J.D.Power. https://www.jdpower.com/cars/shopping-guides/what-is-remote-engine-start-in-a-car