Apollo Enhancement

Date: April 10, 2022

Authors: Apollogizers

Lavi Ionas: 17li3@queensu.ca
Xinyu Chen: 18xc26@queensu.ca
John Scott: 18jbs2@queensu.ca
Baorong Wei: 18bw16@queensu.ca
Zhihan Hu: 18zh22@queensu.ca
Anthony Galassi: anthony.galassi@queensu.ca

Abstract

Baidu's Apollo Auto is a mass scale open software project that is tackling vehicle autonomy. Their efforts have drastically changed the current state of Apollo's software. In the A1 report, the conceptual aspect of the architecture was examined. In the A2 report, the concrete architecture was laid out and articulated. In this report, we propose a potential enhancement for the Baidu Apollo Auto that would affect both the concrete and conceptual sectors of the architecture.

In this report, we proposed implementing a new module in the Apollo software called Communication. This module would take on the task of communicating between networks of autonomous vehicles in order to optimize inefficient and time-consuming tasks like parking. We focused on the motivation, implementation, and the SAAM analysis of the proposed enhancement. By tackling these three key rationalizations for the new module, we concretely realize the steps necessary in incorporating new features into already operational and live software. We propose two use cases that outline our vision for the interactions between the proposed module and the already existing system. We also take extensive measures in planning the methodology for testing the enhancements software to ensure its safety and consistency with approved regulations. Finally, we delve into the two different ways we can realize the enhancement by considering the stakeholders involved and the functional and non-functional requirements necessary.

Introduction and Overview

In this report, we propose a potential enhancement for Baidu Apollo Auto. We think communication of autonomous driving cars cannot only be used between moving cars but in a more static scenario - parking. The current Apollo module is able to park an autonomous car at a designated parking lot, but it does not support our day-to-day parking lots. With the module we propose, we aim to encourage autonomous cars to park at any parking lot efficiently by communicating with other cars empowered by V2X about free parking spots. Under the communication module, we also plan to cooperate with other taxi companies to free up parking places. This enhancement report is the last part of our Apollo Auto architecture analysis, and it will be on our final report with conceptual architecture and concrete architecture.

In proposing and analyzing the potential implementation of new functionality to the Apollo autonomous driving platform, we must cover several areas of interest. First, we formally propose our addition, and present in detail, in both text and diagrammatically, what this addition will add to the project, as well as why it would make a valuable addition. We then discuss the more logistical considerations of adding these features, by predicting the effect that they would have on the maintainability, testability, evolution, and performance of the system. Next, we consider the potential implementation of this enhancement, by regarding its effect on high and low-level conceptual architectures, the impacted modules, plans for testing the implementation, and potential risks that the implementation of this enhancement may introduce. Finally, we perform an SEI SAAM analysis of two different ways to bring about this enhancement, identifying more specifically the stakeholders involved, and the impact that these methods of implementing the enhancement would affect them.

Enhancement

Enhancement Motivation

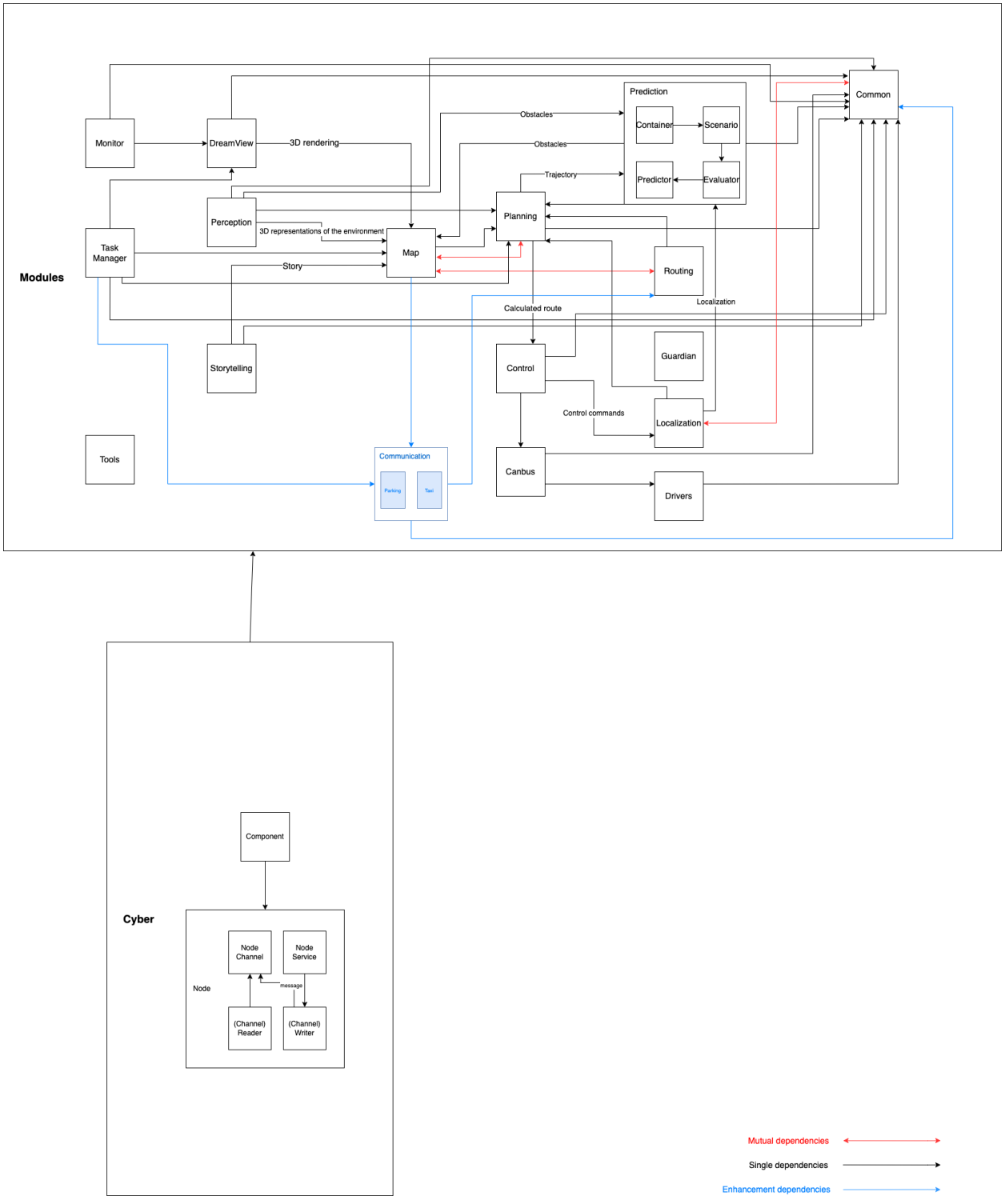
Our proposed enhancement is to implement a new module in the Apollo software called Communication, which would contain two sub-modules Parking and Taxi. The motivation behind this enhancement would be to add functionality for automatic parking and integration with a ride-faring service such as Taxi to utilize the cars more efficiently. Rather than have private cars, public or company-owned autonomous vehicles that can be used all the time are much more efficient resource-wise.

The sub-module Parking would get input from Task Manager indicating that parking is required. Then it would use the Planning and Map modules to determine an appropriate parking spot nearby based on availability. The Map module would check with the Communication module to ensure that there are available parking spots. The Communication module would then reserve a parking spot and send to the Planning module the approval status to move forward to go to the parking space. The vehicle will then find the spot and complete the parking sequence and update the Planning module that the spot is now taken.

The sub-module Taxi would get input from Task Manager with either a pick-up order or drop-off order. The Map module would supply the geographic information about the vehicle. The Communication module would then calculate the distance, time, and cost relative to the person who ordered the Taxi based on geographic information from the Map module. The Communication module would then input the vehicle and the person's geographic information into the Taxi network to be compared with others and find the most efficient vehicle for the task and route. Either the vehicle in question is chosen as the best option, in which case it updates Planning with the destination and will drive to the pickup location. Otherwise, if the vehicle is not chosen as the best option, it will update the Taxi network to indicate it is available for other ride-fares.

Currently, the system relative to this enhancement has a Taxi module, however, most people do not use Taxis anymore. It would make much more sense for our enhancement to be implemented so ride-faring services can utilize the vehicles, which would improve the efficiency and use of the system overall.

Enhancement Implementation



Enhancement Effect on Architecture

In order to facilitate this new addition, some major though noninvasive additions and alterations must be made to the existing conceptual architecture. Although the Apollo platform is already equipped to use the V2X communication protocol, our addition necessitates changes that will handle the receipt of this information from other vehicles, the processing of that information into actionable plans and locations for the car, the collection of further information, and finally the transmission of this new information to other vehicles. Firstly, for the parking functionality, we would need to draw information from the Localization, HD Map, and Planning, so that we may analyze what we need to receive from other vehicles via V2X. For instance, if we know from these modules that we are in a certain parking garage, and are planning on parking, then we know to pull information about this specific parking garage from other vehicles. We would then also know from the HD Map which parking spaces are preferable, by proximity to destination, cost of parking, security, etc. In a similar way, we can combine information from V2X about parking lot availability, and information from the HD Map about proximity to the destination to determine which parking lot is most favorable. Once information about the optimal parking space is found, we can use the Perception and HD Map modules in collaboration to detect which parking spots are taken, by knowing the physical location of the parking spots, as well as detected cars in these parking spots. This information can be used for both the parking of our own car and the transmission of this information to cars which will attempt to park in the future. While the car is collecting this information and trying to find a suitable space, the routing and planning modules will be informed of the systematic scan of the parking area that the car will undergo, and will control the car accordingly. For instance, when the parking module determines through map data and received V2X communication about parking availability which parking lot is most favorable, it will send the position of this parking area to the routing module, which will, in turn, navigate the car. Once it has reached the parking area, the parking module will often loop through an entire parking lot, from more favorable spaces to less favorable positions, until it finds a space. During this time, it will feed start and destination positions to the routing module, which will handle the routing of this loop around the parking lot, and will pass this information to lower-level modules which will execute this route. Finally, when a parking spot is found, the Storytelling module will take over to handle the actual act of parking the car in the space.

Concerning the Taxi service that we would also add functionality for, similar modules would have to be utilized. Although the actual function of the Taxi service is very different, the basic principle of receiving V2X information and planning optimal routes and behaviors is much the same. After a ride is requested, different autonomous taxis could use the V2X protocol to communicate their positions to the passenger, their remaining gas, the remaining time they have for pickups, etc in order to determine

which taxi is most able to take the job. Once one taxi has taken a job, it can also use the V2X protocol to communicate with other vehicles to determine traffic conditions along it's route, and possibly reroute if another vehicle tells it that there is significant traffic on the initial route. The calculation of the most appropriate taxi would be informed by data from V2X, HD Map, and Localization modules. Localization would be used to communicate the location of the taxi to other taxis, and the HD Map would be used to determine which taxi could arrive at the pickup point fastest, a calculation that could be informed by traffic data also observed from V2X transmissions. While en-route, the Perception module would be used to take note of traffic or obstructions on different streets, which would then be communicated to other cars in the V2X network, in turn informing their HD Map modules, if present.

Overall, the addition of our enhancement will have little effect on the existing architecture and will be able to be dropped in-place on top of existing components with minimal disruption. It will change the architecture somewhat conceptually, however, as it will both add more brand new modules, as well as use existing modules in a way that is functionally the same but for a different goal. High-level aspects such as the Implicit Invocation architectural style will not change, as this new functionality fits neatly into this style, but the application of this platform to automatic taxi service will likely have knock-on effects on the trajectory of the project as a whole, as it will alter and add to the major goals of what the software is trying to accomplish, something that is very influential on both high and low levels of conceptual architecture.

Plans for Testing

We could split testing into two major parts: tests in the lab and tests on public roads. The test specialists mentioned below should pass technical and driving tests since they have a direct impact on whether an autonomous car should be certified or not.

As for testing in the lab, test specialists could simulate virtual environments with a variety of traffic situations to test how Apollo is going to react. Behaviors like planning, perception, and interactions between modules should be observed from monitors, and the specialists will determine whether it is effective or not. We should take both software and hardware into consideration. Quality Assurance specialists should ensure the functionality of the code throughout different stages of testing. Simulation of hardware components from sensors to car wheels should be tested under different weather, different parking lots, different lighting, and other various factors to ensure its versatility and functionality.

As for testing in public, we plan to place two test specialists in each car [ref2]. The test specialist who is in the driver's seat will ensure the safety of routing, and if there is an expected behavior, they should switch the driving mode from autonomous to

manual. The other test specialist is responsible to observe the monitor and collect data for later analysis. The efficiency of parking can be measured as follows:

Test specialists and Data Scientists should compare the accident rates between autonomous mode and manual mode. If the result shows that autonomous driving cars have much lower accident rates than manual driving cars in the same scenario, we can mark our module passes the test.

If the collected data shows that autonomous driving cars increase the capacity of parking lots while not creating traffic jams, we should mark our module as efficient.

Potential Risks

Currently, Apollo can only park at designated parking lots. With our improvement, Apollo could potentially park in an ordinary parking lot. However, we cannot eliminate the possibility when most of the parked cars do not have the V2X module. In this case, our communication module cannot receive enough information to collect the possible parking space. Our car could cruise around to explore every single one of its parking choices before it decides where to stop, which could lead to traffic in the parking lot. If the parking space does not follow layouts that the system is familiar with, our Apollo could get lost because it does not know what to do. This confusion may lead to car crashes in serious cases or may lead to traffic jams making it impossible for ordinary vehicles to park or leave.

Similar but worse traffic jam issues could arise if the module acts unexpectedly. If the AI decides to leave the parking lot and slowly drives around before it needs to be put into use, gridlock could happen [ref]. Such a decision could be made when the AI thinks cruising around is more cost-effective than parking at a place or driving home. Although our module allows the car to be used as a Taxi if the number of vacant autonomous driving cars surpasses the number of passengers, cars driving around aimlessly can still cause serious traffic jams.

We may have a difficult time finding the person or organization that is liable for the above accidents. For example, in scenarios where car crashes happen in the parking lot could be difficult to identify the accountability. One may argue that developers should be accountable for the accident, but developers are not the ones who decide if the car is ready to be put into service. However, Apollo may hold developers accountable because they are the ones who failed to detect and debug the error. If the vehicle is used as a Taxi, traffic accident liability is very likely to be pushed between Apollo and Taxi. If drivers want to report the inconvenience caused by slowly driving vacant cars, they may need to go through a complicated process because the issue may be associated with two companies across different departments. The

complaint is very likely to not be addressed due to the complicated and prolonged progress.

Although Apollo has its security system to prevent accidents from happening, we cannot eliminate the possibility of malicious attacks. When the car is used as Taxi, hackers who are hired by offenders may attack the system to modify the passenger's destination without them knowing. While human abduction may be the worst-case scenario, we must acknowledge this possible vulnerability. Other malicious attacks, including bumper stickers or car crashes, etc., have a very small chance of occurring even if we take the most stringent safety measures.

Enhancement Use Cases

Please refer to the use case section of the report for the sequence diagrams that represent the interactions between the new Communication module and other subsystems in the software. Both diagrams display the steps the software and the autonomous car needs to take in order to accomplish the previously laid out tasks.

The first task (*Use Case 1)* of finding a suitable parking space requires the Communications module to interact with a designated Parking network. This network would interact with other automated cars and manage the availability of parking spaces in order to mitigate overlap. This would mean that if one car reserves a certain spot in the network, other autonomous cars would not be able to park there.

The second use case (*Use Case 2*) depicts the interactions between the Communication module and a designated Taxi network. This means that if a passenger requests a pickup, the system undergoes a calculative process to determine which car in the network is optimal (with regards to distance, time, and cost). The best option is then chosen, and that automated Taxi picks up the passenger. Similarly, the system calculates the optimal order for dropping off multiple passengers sharing a Taxi.

Please note that due to the complexity of said sequence diagrams, they tend to be very large and long (unsuitable for a standard document). Hence, they shrink and become practically unreadable. To remedy this inconvenience, high-resolution images will be posted on the group website for viewing / downloading purposes.

Enhancement Maintainability and Evolution

The implementation of the Communication module in the system has minimal invasion in both the Conceptual and Concrete fronts of the architecture. The system has dependencies on other subsystems (Map, Task Manager), but does not inherently change the main structure of the software. Tasks that were executed in particular ways will still follow the same structural algorithm. The change in the data flow is only felt in

the Planning module. The Planning module is dependent on the Communication module to make better choices with the integrated parking and taxi protocols. The system is still capable of making decisions without interacting with the Communication module, but it is within our belief that the module would enhance the said decisions. The ability to communicate between autonomous cars is critical, and saving time finding a parking lot or choosing which passenger to pick up would better improve the software's potential. The main goal of the new module is to enhance the system, not slow it down. This can be done by reducing the amount of overlap autonomous cars would go through if there was no direct communication between them.

The structure's maintainability would remain partially the same. Additional protocols for the Communication module would need to be incorporated into the central maintenance system. The software would need to ensure that the Communication module is running as intended, without deficiencies. These issues, if gone unnoticed, would cause inefficiencies within the system, possibly creating overlaps in parking spaces or multiple autonomous cars would attempt to pick up the same person.

In the evolution of the system, the software would update with a new and improved version of the Communication module. These updates would possibly include bug fixes, additional features, and entire new sub-modules for completing different tasks that require car-to-car communication. These changes would be first thoroughly tested. A pre-released version of the update would be uploaded to a network of controlled automated vehicles to ensure safety and consistency. After these vehicles pass regulatory standards, only then can the update be mass released.

SAAM Analysis

Overview

Our enhancement, the Communication Module, can let Apollo Cars share data with one another or with other devices such that two cars would not compete for the same parking space, and let these already parked cars locate specific parking spaces for the coming Apollo car by using their cameras or other functionalities or the surveillance cameras in the parking lot. This enhancement could be of great help, especially in big cities where parking is a problem, and this could be time-saving as cars could go directly towards the located parking space instead of looking back and forth for a parking space, which could be quite beneficial to all car owners in this parking lot. Also, the Apollo Car can turn into Ride-hailing Service since autonomous vehicles do not require to be parked for a long time. This helps free up parking spaces and reduce traffic flow in big cities.

Stakeholders

- Ride-hailing Service Providers
- Ride-hailing Service Users
- Apollo Developers
- Apollo Users
- Parking Lot Owners
- Parking Lot Users

Scenarios

- The Apollo Car sets to locate a parking spot
- The already parked Apollo Cars help other Apollo Cars locate a parking spot
- The Apollo Car updates the status of the current parking space after parking is completed
- The Apollo Car sets to turn to Ride-hailing Service instead of parking
- The Apollo Car gets data from Ride-hailing Service Providers, including the passenger’s location and destination

Non-functional Requirements

- The Apollo Car can easily and quickly locate a specific parking space before reaching the parking lot.
- The Apollo Car can easily and quickly find the closest passenger to the vehicle and plan the best route to the passenger's location.
- The Apollo Car can quickly help locate a parking space and minimize energy consumption at the same time.
- The Apollo Car can operate safely with the added enhancement.

Two Different Ways For This Enhancement

- The Chosen Implementation:
  Communication between Apollo Cars to calculate the scenarios when the situation is settled
- The Alternative Implementation:
  Communication between Apollo Cars to update the process along the way

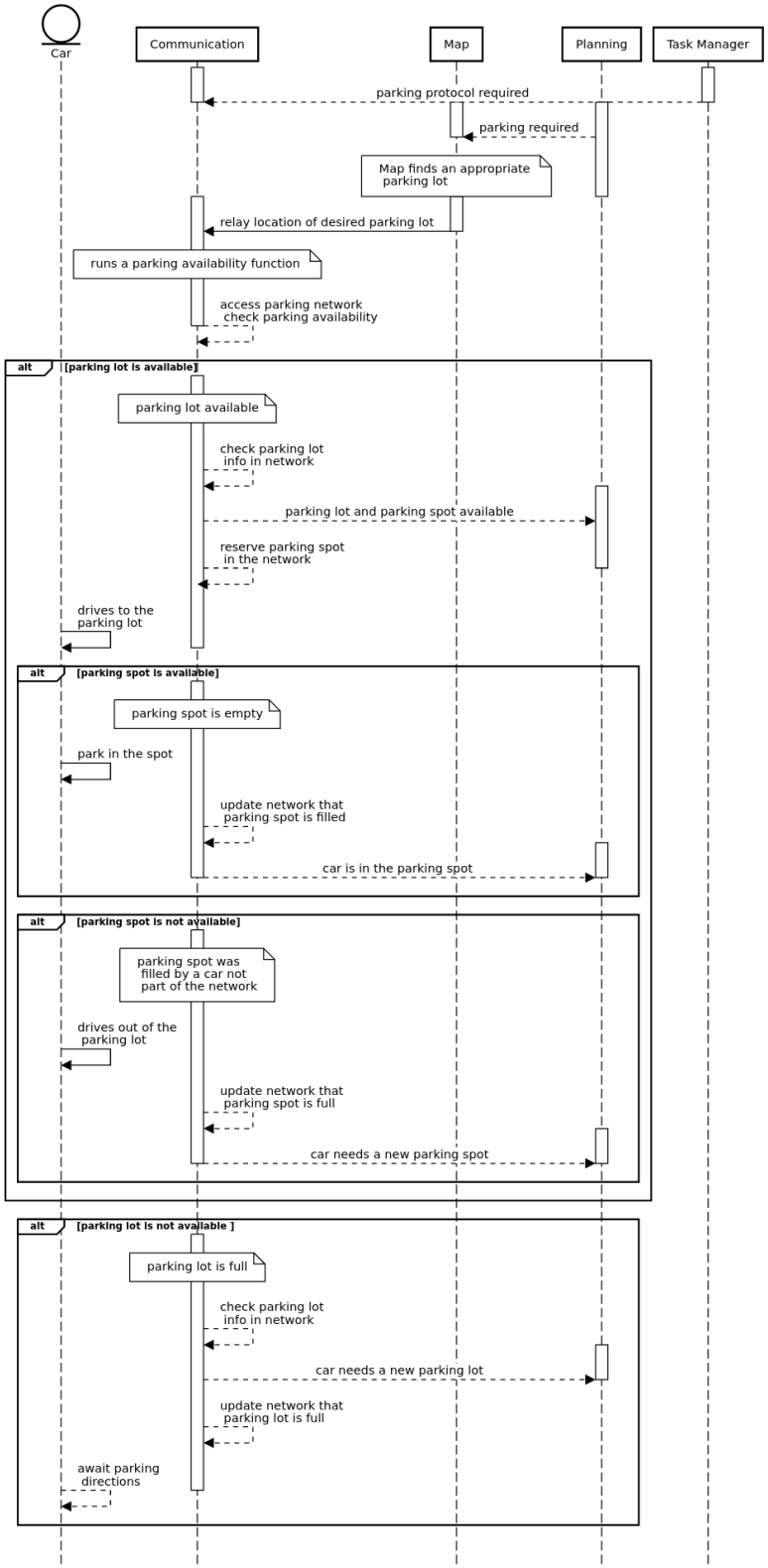
Comparison

Scenarios	The Chosen Implementation	The Alternative Implementation
Locate a Parking Spot	Locate the spot before entering the parking lot. Pros: Saves memory & calculation process as it only considers parking before entering the parking lot Cons: May not find the optimal position	Update the located spot all along the way. Pros: Can find the optimal position as it gets updates of the parking lot along the way Cons: Use more memory & require continuous calculation of the parking lot & more difficult for developers
Help Locate a Parking Spot	Use cameras to help locate a parking spot before a car is entering Pros: More energy-saving & use less memory Cons: May not help find the optimal position	Continuously use cameras to help locate a parking spot Pros: Can help find the optimal position Cons: Less energy-saving as the cameras and other functionalities require to be on continuously & more memory-consuming
Turn to Ride-hailing Service	Lock to a specific client before starting Pros: More memory-saving as it only needs to focus on a single client & safer as all the requirements are settled before leaving Cons: May not find the optimal/closest passenger	Update the client along its way Pros: Can find the most suitable passenger as it constantly gets updates of new clients Cons: Use more memory & may be less safe as it may suddenly turn to a closer client

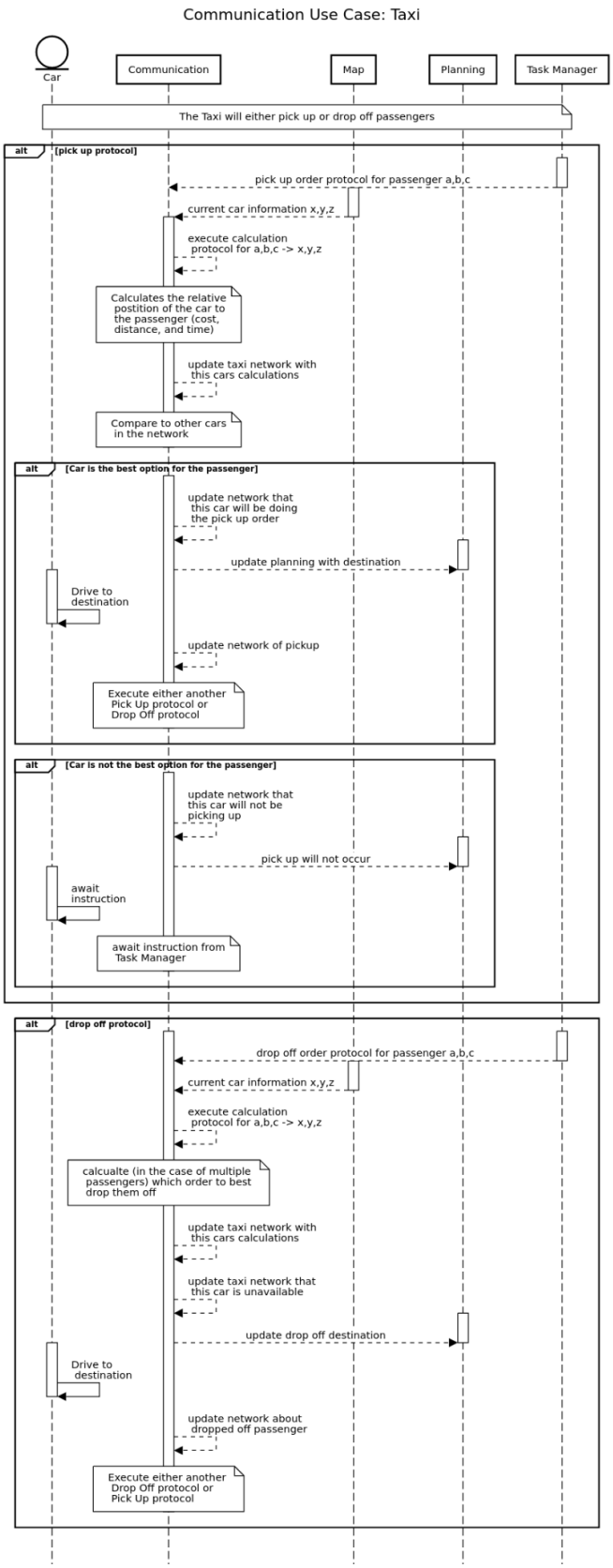
Evaluation

We evaluated our implementations based on performance, safety, and modifiability. We use the chosen implementation instead of the alternative one as the former combines our requirements: locate a specific parking space quickly and easily; help other cars locate parking spaces; more energy-saving and memory-saving; find a suitable passenger. Even though the latter can find the optimal solution more accurately, it continuously does this process, which would require more energy and memory, and its attention towards other functionalities while driving might result in less safety. Also the latter is more difficult for Apollo developers.

Use Cases



Communication Module Use Case 1: Parking



Communication Module Use Case 2: Taxi

Conclusions

Within this report, we discussed our thoughts on the enhancement, the Communication Module, and its functionalities. We clearly stated the motivation and implementation of this enhancement, by using diagrams and tables, and we pointed out the effect of the enhancement on software architecture. We also discussed the testing of our enhancement, both in labs and on roads, and thought of the potential risks. We used 2 Use Case Diagrams to represent the interactions between the Communication Module and other subsystems. We also did a SAAM Analysis to compare the implementation methods based on the stakeholders and the non-functional requirements, and we showed our evaluation of this analysis.

Lessons Learned

This report allowed us to realize that even large software systems such as Apollo can be enhanced through innovative technology and ideas. Considering how well the Apollo software is built, it was difficult to brainstorm some ideas for improvement. However, once we established a gap in the software's offerings to modern technological ideas, we moved quickly to establish our ideas further. We improved upon our last report in terms of organization as we broke up the work in pairs so we did not overwhelm ourselves in terms of workload. By dividing the parts into pairs, we allowed the ideas to stay grounded and concurrent throughout the conceptual and concrete architecture of our software enhancements. Our communication improved greatly from the last report as a team and we had more consistent communication with our teaching assistant, which helped us identify our enhancement sooner. Overall, this report has helped us to see how a real-world idea would be implemented into existing software.

References

[ref] Millard-Ball, A. (2019). The autonomous vehicle parking problem. *Transport Policy*, *75*, 99–108. https://doi.org/10.1016/j.tranpol.2019.01.003

[ref2] Tsai, J. Y., Cranor, L. F., & Craver, S. (2006). Vicarious infringement creates a privacy ceiling. *Proceedings of the ACM Workshop on Digital Rights Management*, 9–18. https://doi.org/10.1145/1179509.1179512