April 11th, 2022

Apollo Proposed Feature Report

Authors

Alexander Nechyporenko (15ann2@queensu.ca - #20021759)
Kennan Bays (18khb5@queensu.ca - #20167866)
Joshua Kouri (18jak16@queensu.ca - #20158940)
Joshua Tremblay (20jkt1@queensu.ca - # 20247327)
Mason Choi (18gc23@queensu.ca - #20165470)

Table of Contents

1. Abstract

This report serves to present our proposed feature for the Apollo system along with an overview of its requirements, two proposed approaches and an explanation for why our team would pick one over the other, sequence diagrams relating to the feature, and an analysis of how certain modules would be affected by the implementation of the feature. Finally, we will conclude with a discussion of team issues, limitations, and lessons learned.

In our previous report, our team was able to obtain a concrete architecture for this system, allowing us to see where and how this feature would fit with what exists in Apollo so far. Our proposed feature would be a learning module that can record information about unexpected obstacles and classify them, allowing for the appropriate reactions to be identified. After implementing this feature, the system's architectural style would still remain as publish-subscribe, as the learning module has its own operations to carry out which would enable other operations in the process.

2. Introduction

In this report, our team will be discussing our proposed feature for Apollo: a learning module, which would deal with the issue of encountering unexpected obstacles. This module would record important information about these obstacles and classify them, allowing for the proper types of reactions to be identified. It would then output this data to the storytelling module, which would use it to create a new story to be sent to other modules in order to decide how the vehicle should behave in this situation.

This report begin with an overview of our proposed feature, along with a description of its many functional and non-functional requirements. The functional requirements of this feature would be to detect, record, and react to unexpected obstacles, and categorize these obstacles as either road-based (e.g., a pothole), obstacle-based (e.g., construction work ahead), or movement-based (e.g., a pedestrian). Then to accomplish these tasks, the Learning module is composed of sub-modules: Filter, Sender, and Library. It would also meet the non-functional requirements of maintainability, testability, evolvability, performance, and security.

Next, our proposed approaches are discussed. The main difference between the two approaches is that in one of them, the module would be located in between the Perception and Scenario Handler modules, while in the other, it would be located in the scenario handlers grouping alongside the Planning and Storytelling modules. This makes the decision between approaches come down to concurrency vs interactivity. By performing a SAAM analysis, our team ended up deciding that the first approach, locating our feature within the prediction module, would be the best method of implementing it. Both approaches were analyzed considering the NFRs previously mentioned and the stakeholders of the software (the driver, development team, Baidu, and Baidu shareholders), and our team found that the first approach is more practical, less costly to produce and maintain, and preserves security.

Two sequence diagrams illustrating this feature's functioning have also been created, as well as a description of how specific modules would be impacted by the implementation of this feature.

3. Proposed Feature

3.1. Overview

A new feature our group proposes to Apollo is the ability to learn from unexpected obstacles. These types of obstacles differ from regular obstacles as they occur as a result of specific events. With this new feature would come a new module which will be called the "Learning" module. This modules' purpose is to recognize these obstacles and record any important related information. After dictating which kind of obstacle is being dealt with, the storytelling module would receive the appropriate information and create a new story. As usual, it would send this story to other modules to determine what behaviour is necessary from the vehicle.

For instance, one of possible scenarios would be to record when a section of the road is covered in ice. When driving, people would slow down in order to not cause the vehicle to slip. The perception module would detect the ice on the road and signal the Learning module. Then, the Learning module would record that this is occurring in the winter season and recognize the obstacle from the traction on the tires. Determining that this falls in a road-based obstacle, the Learning module would send all related information to the storytelling module. A new story about a slippery surface scenario would be created and the vehicle should slow down in caution.

3.2. Functional Requirements

The main functionalities of this new feature are to detect, record and react to unexpected obstacles. The perception module outputs all obstacles in the nearby environment to the Learning module. The Learning module would contain three submodules; a Filter submodule, a Sender submodule, and a Library submodule.

The Filter submodule will skim the data and flag any objects (things detected by the Perception module such as cars and road lines which are outputted), deemed "irregular" or "unexpected." This information is processed and sent to the Library submodule. The Library submodule records this information along with information about factors such as the time, season, and location this incident occurred at. The Filter submodule will request information from the library as well. If a registered obstacle does not reappear when it is expected to reappear, it may be removed from the database. Users would also have the ability to choose whether they would like to enable or disable certain classifications of objects according to their preferences. If a classification is disabled, the Filter would not record information relating to it. The Sender submodule, at certain time intervals and when approaching a significant object, will send all relevant information to the Scenario Handlers. Relevant information will be objects

which have been consistently encountered under conditions the vehicle is currently experiencing. These objects will be treated as second-class objects and will enter the decision process here, where the Storytelling module may or may not use them to determine the vehicle's trajectory. The Storytelling module will decide whether to exploit this information to change the vehicle's behaviour.

Additionally, the Learning module would introduce new classes of obstacles with their own unique triggers:

Road-Based: These are obstacles that occur as a result of tire traction with the road. Examples of such obstacles would include potholes, icy patches, and gravel patches.

Obstacle-Based: These types of obstacles would be viewed as perceived obstacles. Construction work and obstructively parked cars would fall into this category.

Movement-Based: These obstacles are noted by their changes in movement. They can be sudden or frequently occurring. Obstacles in this category include people and poorly driven cars.

3.3. Non-Functional Requirements

In addition to the functional requirements, this report will present some non-functional requirements that can be applied to this new feature. Potential risks tied to these requirements are also mentioned.

Maintainability: This feature would need the same maintainability as existing obstacle detection and handling. If either scenario is failing, the result would put users at a safety risk. Further, the downtime and fixing time of the feature should be noticeably short to prevent as much risk to the users as possible.

Evolvability: Our feature would evolve well as new classification and obstacles could be implemented whenever needed. It would be expected that, over time, they could change or additional ones would be implemented.

Performance: This feature should perform to the same standards as the existing obstacle detection and handling. Fast response times from all modules affected allow the vehicle to react to any type of obstacle quickly. If response times are too long, the vehicle would not be able to react to the obstacle before potentially causing a collision.

Security: Each new obstacle scenario should have methods to prevent security attacks. Each obstacle needs to be notified by its correct trigger and have the proper vehicle reactions. Additionally, the ability to enable and disable certain obstacles can create some risks. This feature should be protected as well and analyzed to ensure it is not of harm to users.

Testability: Each obstacle type should be simulated and tested to ensure that each scenario is functioning as intended. The perception module should provide every type of unexpected

obstacle to the Learning module. The storytelling module should be able to create stories from each unexpected obstacle as well for the vehicle to behave properly.

4. Proposed Approaches

Our team produced two approaches with which could be implemented to create our proposed feature. The main difference between approaches is where the Learning module would be located. The effects that these approaches would have on the system's NFRs will be described in our SAAM analysis. In both methods, the user can enable/disable specific learned behaviours, such as whether to adapt to ice patches or not.

1. In our first approach, the Learning module would be located in between the Perception and the Prediction module. An extension of the Perception module's object recognition capabilities may be required to fully exploit this module. The Learning module will receive all outputs from the Perception module. The sender sub-module would still transmit all information needed to the Storytelling module to allow changes to the vehicle's behaviour. Obstacles, such as potholes, may appear suddenly and force the Planning module to create a sudden turn to the right to avoid it. The Learning module allows repeat events to be dealt with in ways that promote smoother driving experiences. An example of this would be the vehicle encountering thick construction at the end of a sharp turn. The vehicle may reduce speed by 10km/h the next time it encounters this sharp turn to avoid a sudden stop.
2. In our second potential approach, the Learning module could be located within the Scenario Handlers group, alongside the Planning and Storytelling modules. It would still be sending output to the Storytelling module and receiving information from the Perception module, but in addition it could have the potential to be more active as a scenario handler as it is no longer just a component of the Prediction module. For example, the Learning module and the Storytelling module could pass information back and forth and the Storytelling module could query calculations from the module based on their collective information.

4.1. Concurrency

Most of Apollo's modules run concurrently while the centralized system of Cyber RT manages them, and our proposed feature would be no different. This is crucial for the functionality of the system as the Learning module should be able to work in parallel with all the others, without the concern of a Learning module failure severely impeding other modules and resulting in catastrophic failure of the entire system. If the Learning module is impeded, the Storytelling module will not be able to use its output, but it can continue to do the rest of its tasks.

5. SAAM Analysis

In Table 1, it is discussed how the two previously discussed approaches compare when discussing the proposal's Non-Functional Requirements:

Table 1: Comparing how the different approaches succeed to relation to the Non-Functional Requirements

Attribute	**Approach #1**	**Approach #2**
Maintainability	By isolating the module as in Approach #1, maintaining the module would be easier. Its interactions are more stable and thus modifying/adapting the Learning module, which will likely be done frequently due to the module's adaptive nature, would be easier.	Due to the Learning module here being a separate subsystem within Scenario Handlers, it would be more difficult to maintain. This is because the Learning module would act as both an intermediary between Perception and Storytelling modules, along with being a separate module that does additional computational work. As a consequence, this approach would require more maintenance from a testing standpoint, along with including more interactions and calls between the systems.
Evolvability	This approach would handle evolution relatively well. The module is designed to support a variety of object classifications and is isolated from other modules. Additional support will be introduced over the years and reworking the module is easy due to its isolation. Bloat might be introduced due to its support for individual classifications of objects.	This approach handles evolvability slightly worse due to its interdependence with other modules. Changes will have to occur in larger batches as the testing must be more thorough. Tweaks would require too much testing to happen frequently.
Performance	The concurrency this approach provides allows it to be quicker than the opposing approach. By acting concurrently to the calculations of known object's trajectories, it can reduce throughput time for perceptions to decisions.	The lack of concurrency may cause a decrease in performance. However, this trade-off does come with the additional functionality of responsive learning.
Security	The module interacts with fewer objects, which should reduce security risk.	The interactions between modules may introduce vulnerabilities. However, the responsive aspect of this approach will dramatically increase the possible responses of the vehicle and help promote drivers' safety across many complex situations.

Testability	The isolated nature of this approach allows for much more efficient and successful testing.	Harder to test, due to also needing to trace interactions with external modules, such as Storytelling.

In Table 2, the relationships between stakeholders and the NFRs is outlined. This will broaden the scope on what the more desired NFRs are for the project based on the analysis of the stakeholder needs.

Table 2: Comparing the relevance of different Non-Functional Requirements for different interested parties

Attribute	**Apollo Driver**	**Development Team**	**Baidu (Parent Company)**	**Baidu Shareholder**
Maintainability	Safety concerns for the driver. If a system failure occurs due to this feature, it can have potentially devastating consequences	Developers would need the same maintainability of this module as existing modules in order to maintain consistency between modules.	For the company, the code base will have fewer issues, therefore costing less money to maintain the codebase. It would also avert any bad PR due to system failures.	The shareholder wants to maximize profits, therefore less expensive maintenance and avoiding bad PR is crucial.
Evolvability	A more dynamic Learning module will result in a better driving experience	For developers, being able to add new features to Learning modules seamlessly will make the job easier. This will also allow the developers to focus on other important tasks and not get bogged down in expanding the Learning module.	By offering more dynamic services from Apollo's Learning system, the product can be seen as a safer option to its competitors, along with added convenience of a better driving experience. This would draw more customers to use Apollo.	If Apollo's product is seen as safer and more convenient to its competitors, this will draw more customers to Baidu and Apollo. This will positively affect company revenue and stock performance.
Performance	Performance at this level should not be seen by the driver if performance lags exist due to the Learning module that can potentially cause safety issues for the driver. If	Performance would need to be on par with other components from a consistency perspective. This will make the job simpler for the developer.	The company only cares about the performance not being detrimental to the safety of the car. Otherwise, performance issues in the Learning module	Same as the company, shareholders want to avoid any safety concerns, especially when associated with self driving cars. It would be very bad

	performance is good, the driver will not see it, if it is bad then there is a serious problem. Hence, only a certain threshold needs to be met.		will not be seen from a customer perspective.	reputationally for Baidu if they release a car with poor performance that affects safety. This would negatively impact the stock price.
Security	The customer will want a secure car. If the Learning module releases data to the public or to bad actors, it can have potentially devastating consequences for the driver. Such as stored data on routes taken often that should be kept secret.	Security is a fundamental aspect of the Apollo architecture and needs to be upheld, otherwise the developers are not doing an adequate job.	If data is released from the Learning module to the public, it would have an extremely negative impact on PR. This would result in a lowered bottom line from the Apollo product due to perceptibility.	If customers view Apollo as not being secure with data, then customer perception would impact the usage of the Apollo product. This could also impact Baidu's reputation as a whole, which can have severe consequences on stock prices.
Testability	If done properly, the driver will not see a difference. Hence, a certain threshold of testing needs to be met.	Proper testing will make it much easier for the developer to find issues in the code, which will allow them to better maintain the Learning module. If issues arise with the Learning module, they will be able to find the source quickly and fix it.	If done properly, then the Apollo product works as intended. If not, then issues with maintaining the product can arise which can impact the feature's utility. If the utility becomes low, then Apollo as a company loses money on building it.	If it works as intended, then the bottom line will not be affected for the company. If done poorly, then resources will be wasted on implementing the Learning module without a solid ROI (return on investment)

Approach #2 is focused on security. The dynamic Learning module approach allows for it to interact with other decision making modules and will produce a safer driving experience. However, by all other metrics, maintainability, testing, performance, and evolvability, Approach #1 is preferable. Meanwhile, the concurrency introduced in Approach #1 and the performance gap may lead to Approach #1 making quicker decisions, which usually end up being safer decisions since no one wants to be lagging on the road. Overall, the only way Approach #2 is preferable, is if the security gap was large enough to outweigh all of the advantages of Approach #1. Ultimately, while it is still valid to prefer Approach #2, our group believes the advantages outweigh the security hindrance. Additionally, the speed with which patches can be developed in

Approach #1 which could solve any discovered security lapse is much higher than that of Approach #2, meaning that Approach #1 feels more effective to meet the desired NFRs.

Approach #1 is also more cost-effective. It reduced testing time and improved maintainability and evolvability will accelerate development. While this is great for the company, since costs are always a factor, it is also important for third-party developers. A large aspect of the Apollo design is its ability to incorporate features with ease, in which evolvability plays a large role in. This approach could allow for third-party developers, who are not yet established and thus have a higher concern for cost, to produce extensions for cheaper amounts.

6. Sequence Diagrams

For the following paragraph, refer to Figure 1. During the driving process, the Perception module repeatedly gathers information about the vehicle's surroundings and states. If an unexpected obstacle is detected, the details are sent to the Filter submodule, which then reports the details to the Library submodule. The Library then requests the vehicle's current location as well as the current time. It then checks if this object has been recorded before; if it has not, the Library generates a new record with the conditions and object details. If the obstacle has been encountered before, the Library appends the new details to the existing record.

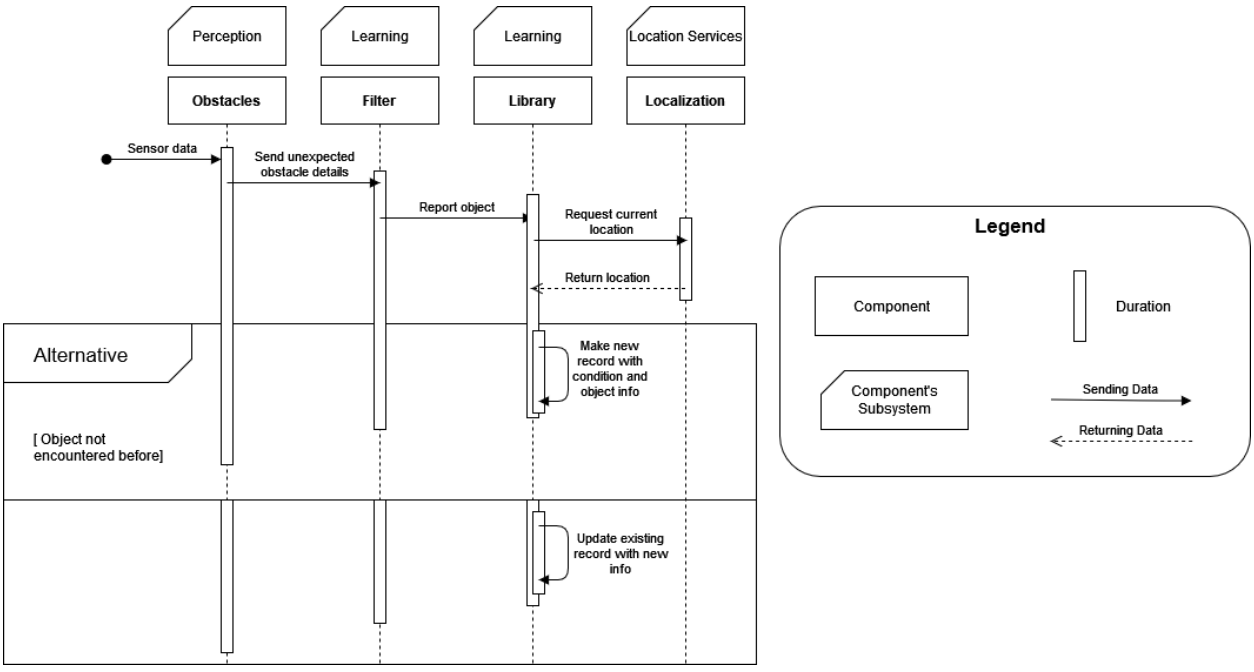

Figure 1: Sequence diagram showing the use-case for when an unexpected obstacle is encountered

For the following paragraph, refer to Figure 2. When the route changes, either due to an unexpected detour or a user event, the Sender submodule will be activated. It first queries the Routing module for the current route, then requests all potential obstacle records on this route

from the Library submodule. Sender then queries the location of the vehicle from the Localization module, then filters for all approaching records within range of the vehicle. These records are then sent to the Scenario Handler modules for future consideration once the obstacle is approaching.

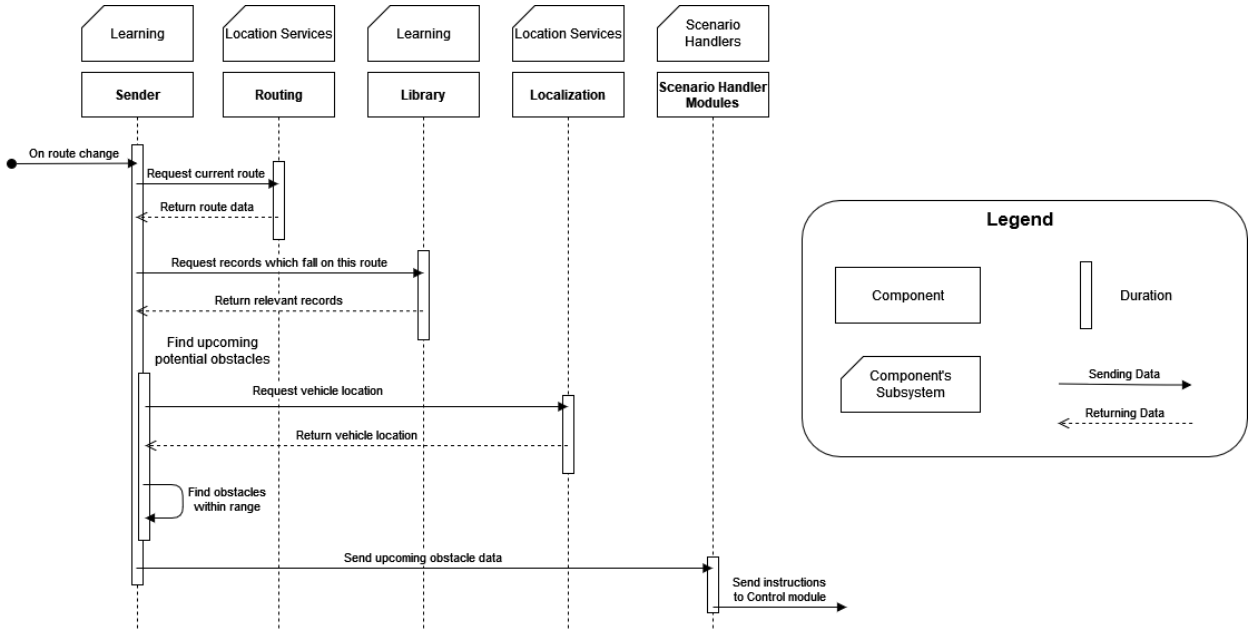

Figure 2: Sequence diagram showing the use-case for reacting to a previously identified obstacle

7. Affected Modules

The following affected modules and directories were determined by using Approach A. The main modules affect are those along the decision making route.

7.1. Perception, Prediction, and Planning

The Perception module creates objects which represent important environmental information. Apollo is still early into development and the Perception module may not have the complete range of functionality to ensure that all obstacles can be detected. Obstacle detection for entities such as ice patches or potholes may not be fully supported yet. Thus, the Perception module at *https://github.com/ApolloAuto/apollo/tree/master/modules/perception* may require further development to fully exploit the potential of the Learning module. This would also require an expansion of the classification range for objects outputted by the Perception module. The Learning module must be subscribed to the Perception module.

The Prediction and Planning modules will be required to implement any additional functionality introduced to the Perception module. If, for example, an “unavoidable obstacle”

object is introduced, the remaining modules must have functionality introduced to be able to support the functionality of additional classifications.

7.2. Storytelling

The Storytelling module is designed to handle complicated situations. Thus, the Storytelling module, not Planning, will exploit the information generated by the Learning Module. The Storytelling module must be subscribed to the Learning module and will now have additional complexities introduced into its calculations. Additionally, an object generated by the Learning module must be treated differently from an observed object. This would be a complicated series of reworks which may fundamentally reshape the Storytelling module. To ease this redevelopment, the Learning module will only send the Storytelling relevant information and will denote this information with a level of priority to treat that object with. Storytelling will treat objects differently depending on whether they originate from the Prediction module or the Learning module. Additionally, a Learning module object which can be cleanly mapped to a Prediction module object will be ignored.

7.3. Monitor

The Learning module will only register objects the user wishes for the system to attempt to learn from. Thus, the monitor must provide an interface to enable/disable learning for specific objects. An example of this would be disabling ice patch detection in the summer. The Monitor module would have a two-way dependency introduced between it and the Learning module.

7.4 Location Services

Location Services would be subscribed to by the Learning module. During its run, the Learning module would receive updates from the Location module which allow the system to pinpoint where the objects of note are located.

8. Team Issues

Since our team chose Approach #1, the Learning module is its own unique subsystem, meaning it will require the most work to implement and will therefore be within the critical path of implementing this feature. The optimal team for this task will be one with experience in AI systems due to their relevance to the Learning module. Additionally, the team may want to work with or take on developers from the Storytelling module, due to the Learning module’s integral interactivity with it. The Prediction and Planning subsystems will both require updating as well, but these changes are relatively small in comparison.

9. Limitations and Lessons Learned

Our group encountered several limitations throughout this project which impacted our ability to accurate create a proposal for an Apollo extension:

- Our team could not find any public reference architecture which detailed how our proposed system would work. The field of self-driving cars is still very new and this feature would be implemented late into development. This made understanding if our team were moving in the right direction difficult and forced us to improvise to deliver on this feature.
- Options for implementation are very numerous. The approaches proposed are still rather general and can only specify certain aspects of the architecture. This is because fully fleshing out this component would be extremely difficult to do in advance. It would also be difficult to assess the completeness, safety, and effectiveness of the implemented solution. Within our group, we had multiple discussions on the optimal placement.
- Conducting the SAAM analysis was difficult due to the non-quantitative difference between them. Our team had to compare NFRs and assess which of these non-comparable attributes were preferable.

Our team was not confined by limitations. While facing these limitations and other challenges, we learned some valuable lessons, such as:

- We discussed how implementing different approaches impacted future development. Our team learned how to evaluate the effectiveness of different approaches long-term. Prioritising modularity and independence is something our team has little experience with. Our team developed experience with prioritising development which was flexible and would stand the test of time.
- Some challenges have no clear solution. Whether to prioritize security over other features is a tough decision with no clear answer. Our team learned how to compare non-comparable traits to arrive at the best outcome.

10. Conclusion

To summarize, our proposed feature would deal with the issue of encountering unexpected obstacles by introducing a Learning module to deal with them. The functional and non-functional requirements of the feature have been described. Two possible approaches were generated for implementing this feature (locating it between the perception and prediction modules, or locating it with the scenario handler modules), and after performing a thorough SAAM analysis, we have concluded that the first approach would be the best way for this feature to be implemented. In addition, two sequence diagrams have been created, along with a description of impacted modules. Overall, we think that this would be a beneficial feature for Apollo to have.