A2 Concrete Architecture of Apollo

Cisc 322 Group 20

Youtube: https://youtu.be/vgMvPHBWTUY

Presenter: Yucan Li, Yuzhe He

Intro

Members:

Yucan Li 18yl259@queensu.ca

Yuzhe He 18yh46@queensu.ca

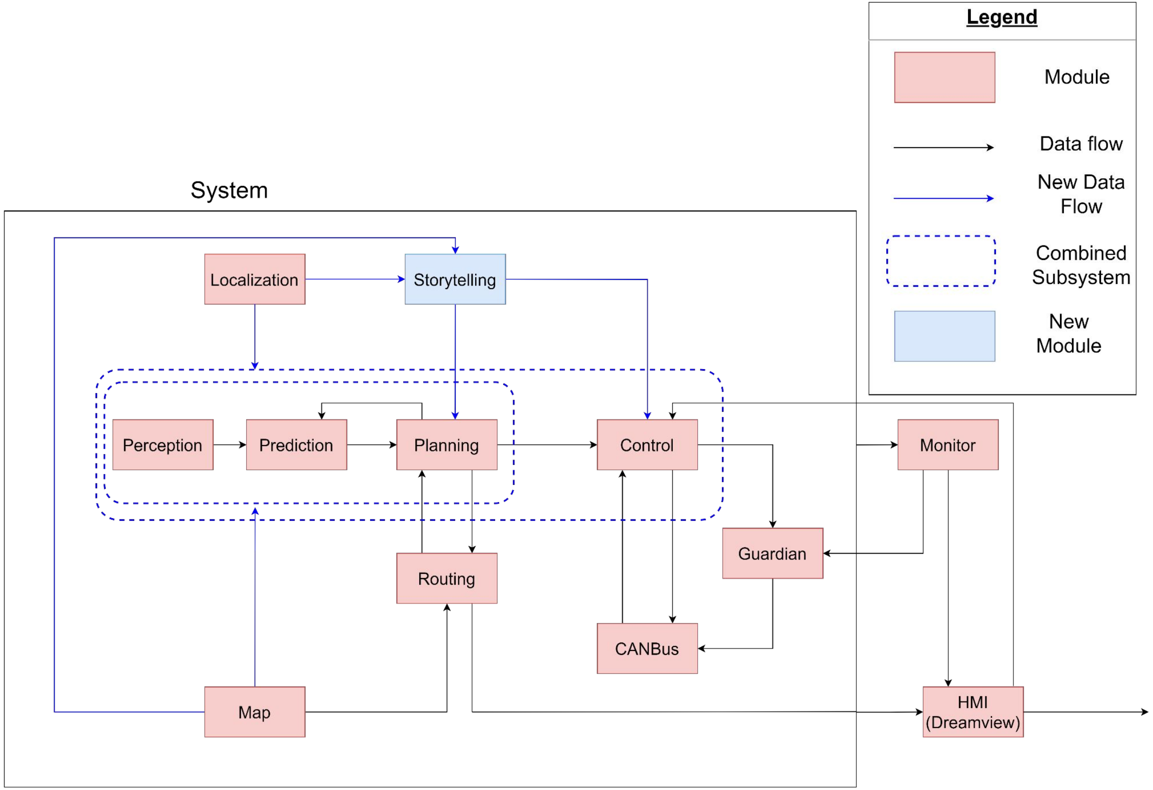
Xuchuan Mu 18xm24@queensu.ca

Yiming Zheng 19yz38@queensu.ca

Wenran Hou 18wh10@queensu.ca

Mukun Liu 19ml13@queensu.ca

Modified Conceptual Architecture



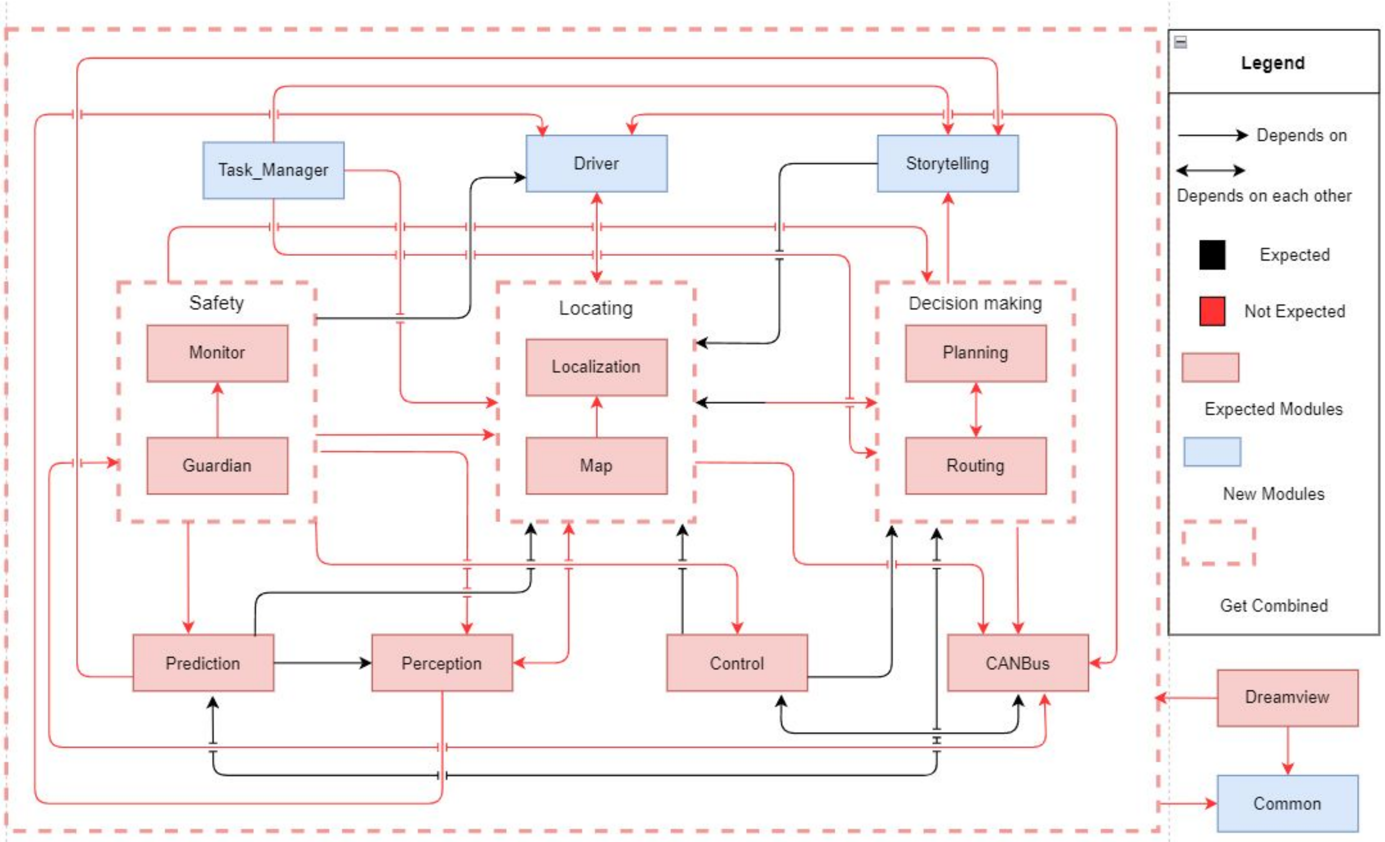
Updated component:

Storytelling

Data transmission from Map to Perception, Perdiction and Planning

Data transmission from Localization to Perception, Perdiction and Planning

Concrete Architecture



Task_Manager

Driver

Storytelling

Safety

Monitor

Guardian

Locating

Localization

Map

Decision making

Planning

Routing

Prediction

Perception

Control

CANBus

Dreamview

Common

Legend

Depends on

Depends on each other

■ Expected

🟥 Not Expected

Expected Modules

New Modules

Get Combined

Reflexion Analysis

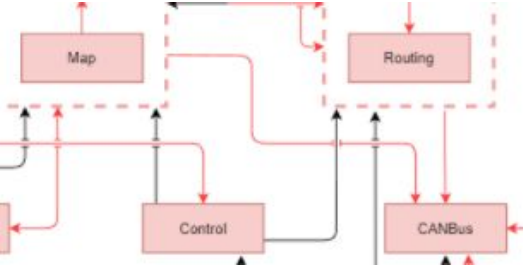
Divergence - Unexpected realtions:

Map -> CANBus: collect info like speed and acceleration ot produce high precision map

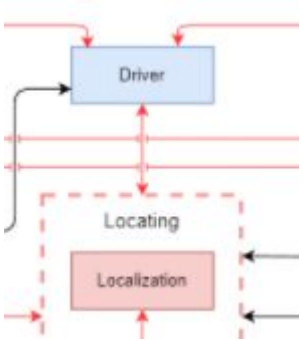
Localization -> Driver: The localization module, as a listener, relies primarily on the four parameters of the driver

Perception -> Localization: uses LocalizationEstimate to estimate if the vehicle has enough speed.

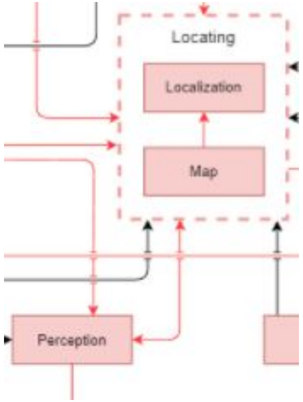
Monitor -> Localization: data aggregated in the summary_monitor and provide system status as output.



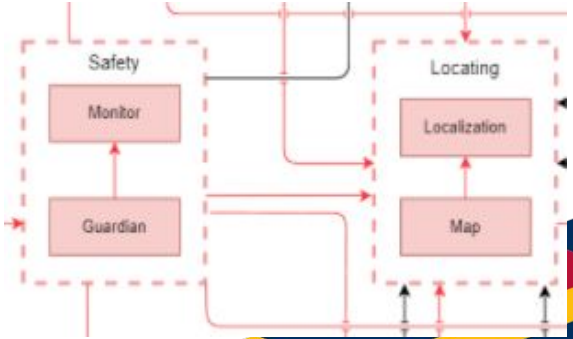
Map -> CANBus



Locolization -> Driver

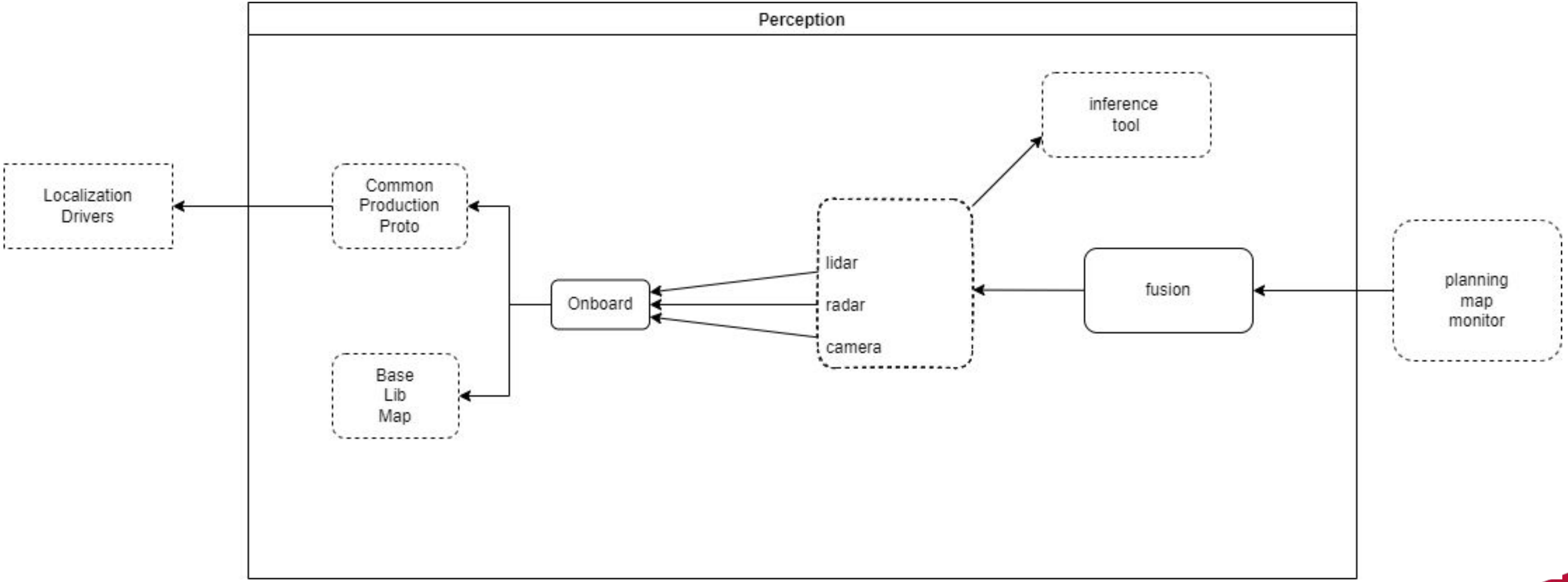


Perception -> Localization



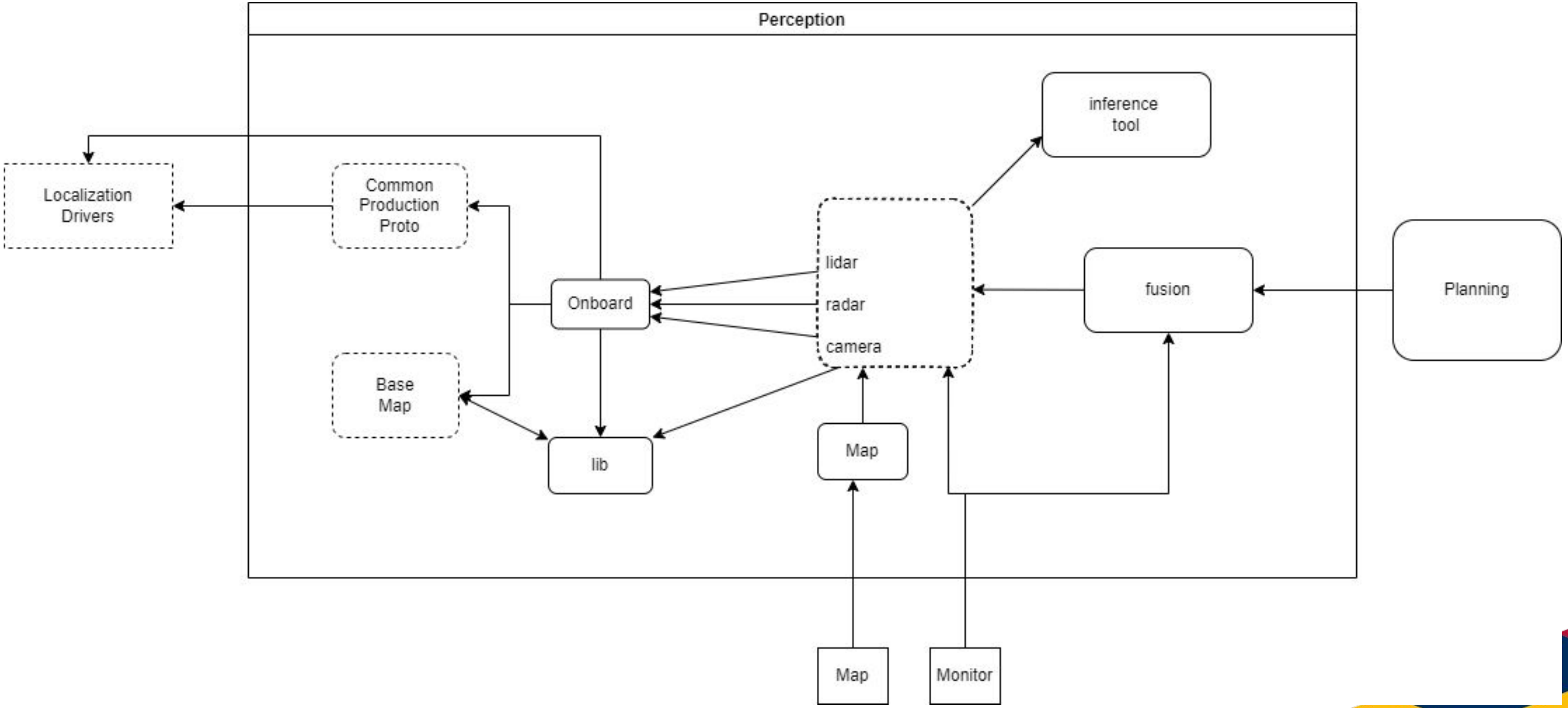
Monitor -> Localization

2nd level Subsystem: Perception



conceptual architecture of perception

2nd level Subsystem: Perception



concrete architecture of perception

Sequence Diagram

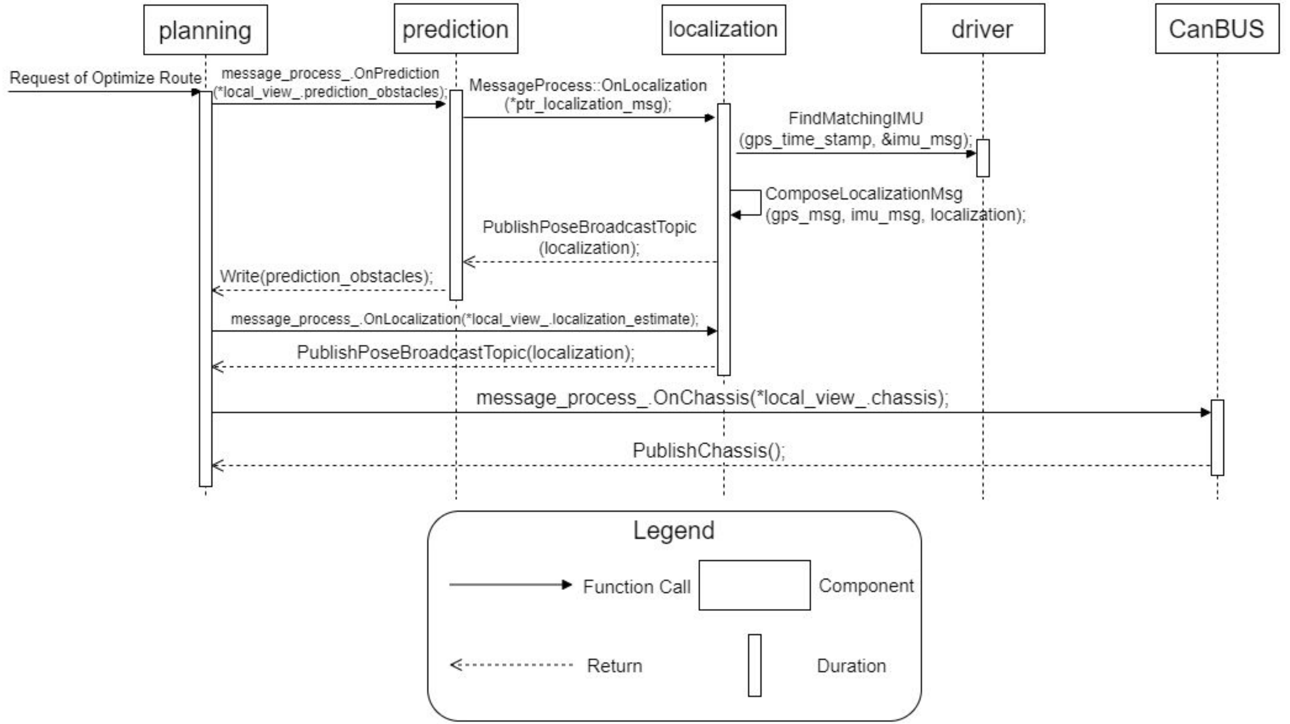

planning
prediction
localization
driver
CanBUS
Request of Optimize Route
message_process_.OnPrediction
(*local_view_.prediction_obstacles);
MessageProcess::OnLocalization
(*ptr_localization_msg);
FindMatchingIMU
(gps_time_stamp, &imu_msg);
ComposeLocalizationMsg
(gps_msg, imu_msg, localization);
PublishPoseBroadcastTopic
(localization);
Write(prediction_obstacles);
message_process_.OnLocalization(*local_view_.localization_estimate);
PublishPoseBroadcastTopic(localization);
message_process_.OnChassis(*local_view_.chassis);
PublishChassis();
Legend
Function Call
Component
Return
Duration

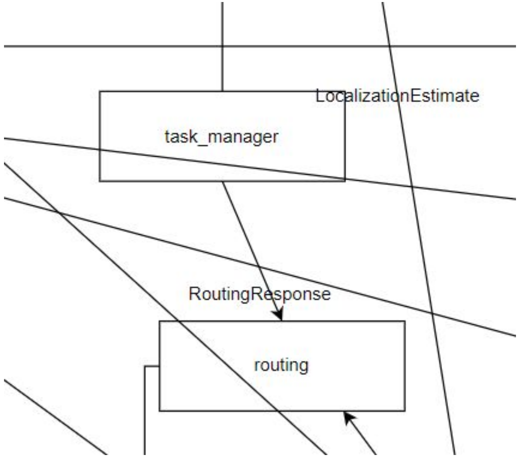
Figure 3: Sequence diagram illustrating the process of optimizing route locally.

Derivation Process

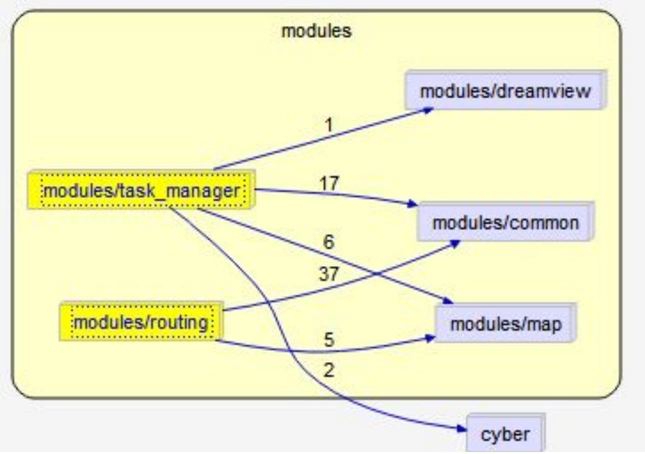
(1) started to investigate the dependencies between these submodule

(2) submodules in Understand have no dependencies like professor shown in his graph

(3) submodules might communicate with each other indirectly through common and cyber



prof's graph



understand graph

Concurrency and team issue

Concurrency happend during data transmission:

(1) routing modules request origin and destination infomation from both localization and map modules. (2) Planning module work concurrently with perception and prediction module

(3) Cyber RT provides a coroutine scheduling algorithm ("CRoutine") that help the oprimization of the system. [2]

Team issue or Division of Responsibility:

During team members' coordination in prediction and planning module, defect in data synchronization may be arised due to mistakes in cooperation.

Considered Alternatives

An alternative architecture style could be:

Process Control style

Monitor -> supervising process

Guardian -> controller (manipulate porcess)

Control -> collect data and send command (controller)

CANbus -> implement command from Control and sendback feedback(process)

Learned Lesson and limination

Liminations

- Understand 5.1: professional and efficient software, but high learnability and it crashed often
- Missing files from “proto” folders.

Learned Lesson

- Using Understand to analyze relationship between modules.
- Through analyzing source code, we develop better understanding to C++.
- Understand a well-round system could have multiple architecture style.

Conclusion

- Pub-sub architecture style.
- We find that the common module provides universal functions to all of the modules. Modules could communicate in the adapter inside the common module

Reference

[1]*apollo developers's centre*. Apollo. (n.d.). Retrieved February 19, 2022, from https://apollo.auto/developer/index_cn.html#/

[2]https://github.com/ApolloAuto/apollo/blob/master/docs/cyber/CyberRT_Terms.md

Thanks for watching