Video Presentation Link

https://youtu.be/j23xGSHGmro

Group Breakdown

Group Member	Contribution
Haadia Mufti (Group Leader)	SAAM Analysis, Abstract
Emily Poon	Effects on High and Low Level Architecture, Presentation
Kevin Shroff (Presenter)	Plans for Testing, Lessons Learned
Oliver Cao	Potential Risks, Conclusion,
Gregory Secord	Use Case, Enhancements
Connor Colwill (Presenter)	Use Case, Introduction

Feature Proposal:

Apollo v7

Group 9: Haadia Mufti, Emily Poon, Kevin Shroff, Oliver Cao, Gregory Secord, Connor Colwill

Overview

- Introduction
- Enhancement
- SAAM Analysis
- Effects on High and Low Level Architecture
- Enhancement on Use Case
- Plans for Testing
- Potential Risks
- Lessons Learned
- Conclusion

Introduction

- Using conceptual and concrete architectures previously derived, implementations of new features to the system are discussed
- Comparison of proposed enhancements using SAAM analysis
- Impact of enhancements to the architecture
- Scenarios using enhancements using sequence diagrams
- Possible testing requirements for the enhancements

Enhancements

1. Notifications pop up at regular intervals through Dreamview

2. Putting a camera inside the car so eye-movement from the driver can be tracked

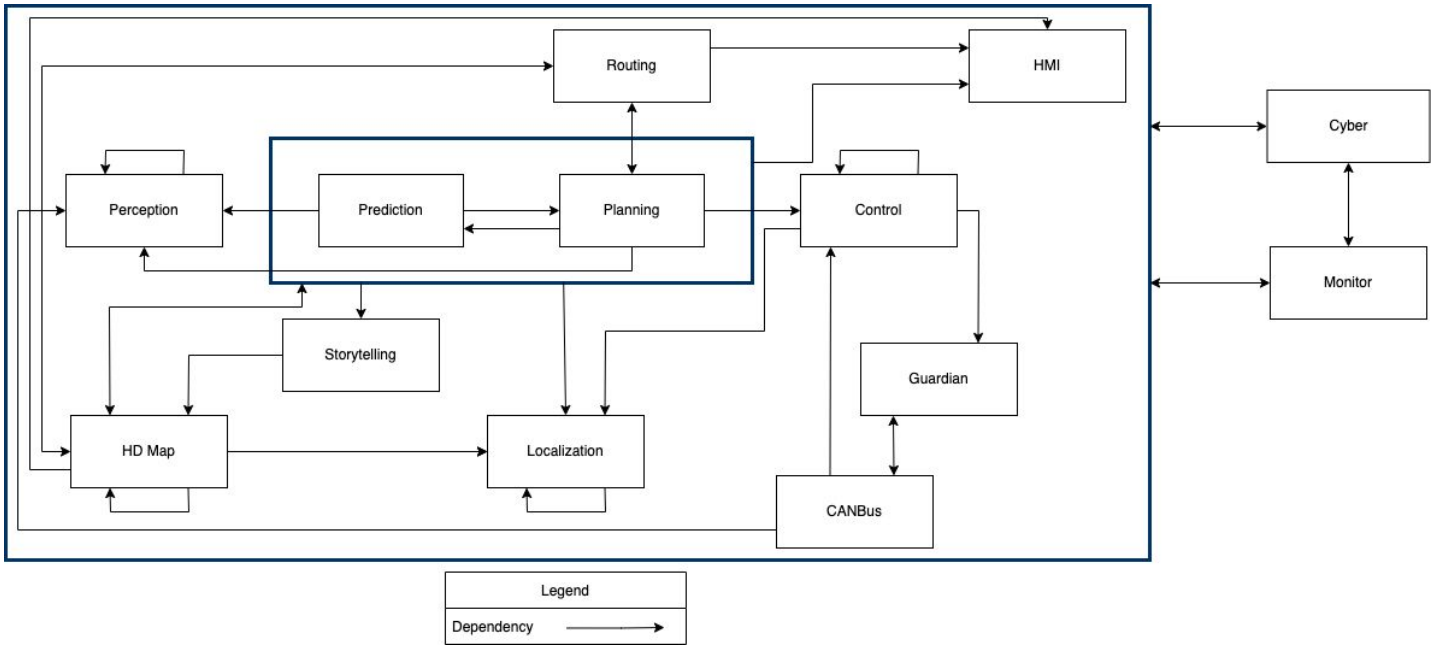
SAAM Analysis

- Notifications implementation is a much simpler way of implementing the enhancement
  - Easy to maintain and test
  - Not require many changes into the overall architecture of the system
- Camera implementation
  - Will require more hardware
  - Change in the conceptual architecture of Apollo

SAAM Analysis Conclusion

- After performing the SAAM analysis, we decided to go with the camera implementation method
  - Better result than the notifications
  - Make roads safer

Concrete Architecture



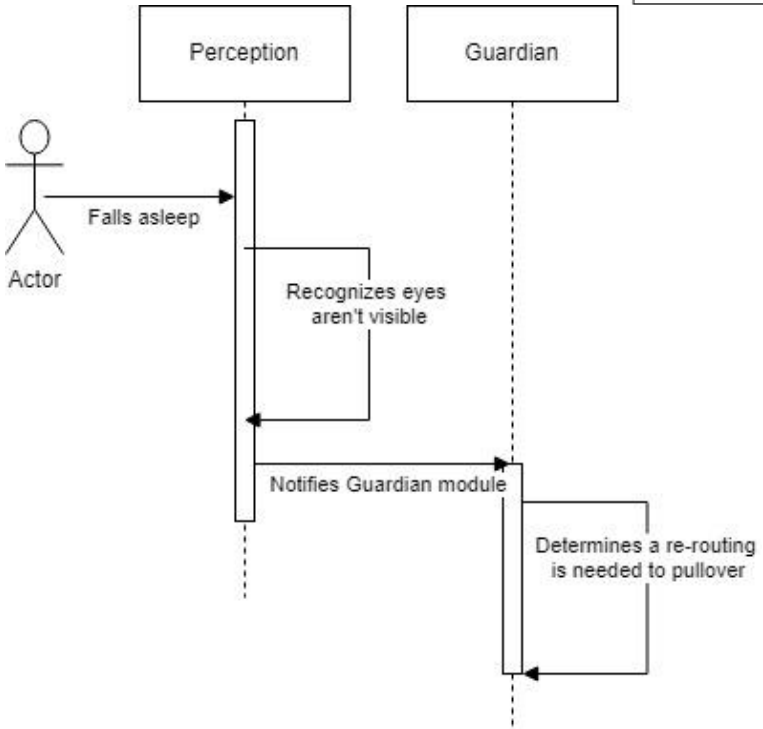
Effects on High Level Architecture

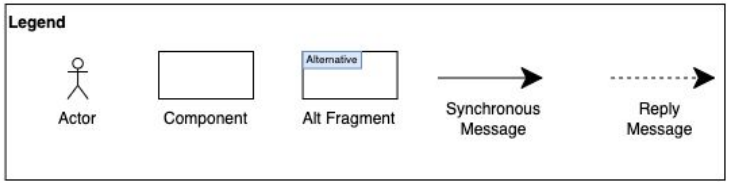
- Still Publish-subscribe architecture style
- Improvements within each subsystem and some new dependencies between subsystems
  - For the enhancement of driver notifications, there would be a new dependency from HMI/Dreamview to Guardian

Effects on Low Level Architecture

- Greatly affected
- Following directories and files would be impacted for the first enhancement (notifications):
  - perception/onboard
  - dreamview/backend
  - dreamview/frontend
  - guardian/conf

System Diagram





Testing - Implementation 1: Notifications

Use Case 1

- There must be a prioritization system to handle multiple notifications simultaneously
  - Prioritize based on severity: Pre-collision Warning >Driver attention > Infotainment.
- The MID must be tested and the priority system must be verified
  - Can be done by creating mock notifications, with pre-assigned priority levels
  - Verify that the notification stack prioritizes notifications with greater priorities

Testing - Implementation 1: Notifications

Use Case 2

- Dismissal of a Driver Attention notification should correctly remove the notification from the notification stack.
  - Autonomous operations should continue unimpeded by the notification
  - Notification should reappear at a predefined regular interval (can be dismissed again)
- Notification dismissal must be tested:
  1) Begin autonomous operation
  2) Observe Driver Attention Notification
  3) Dismiss Notification
     a) Ensure Autonomous control continues
  4) Verify Notification reappears

Testing - Implementation 2 - Camera

Use Case 1

- Use Driver Attention System to determine where driver is focused.
  - Must have high accuracy and reliability to be effective.

- Driver Attention system (eye-tracking specifically) must be tested to ensure high accuracy.
  - Accuracy and certainty of >90% is recommended

Testing - Implementation 1: Notifications

Use Case 2

- If Camera is obstructed, the predefined behaviour is to:
  1) Prohibit Autonomous operation, until obstruction is removed
  2) Raise Notification via the MID

- Behaviour needs to be tested to verify correct safety procedures:
  1) Notification is correctly raised through MID when:
     a) Sensor issue is corrected
     b) Car is started
     c) Autonomous Operation is requested
  2) Autonomous Operation is disabled and prohibited until sensor is unobstructed

Potential Risks - Implementation 1: Notifications

1. Drivers may become dependent on notification system, in order to prevent distracted driving
2. Drivers may be unable to respond to potential collision notifications, if the driver is not attentive
3. Software may introduce another point of failure (glitch/bug), potentially pulling over vehicle, when it could be a danger
4. Drivers may develop a habit of dismissing notifications on a repeating time interval, continuing to drive while distracted.

Potential Risks - Implementation 2: Camera

- Drivers may become more dependent on autonomous driving and monitoring system.

- Camera System is another point of failure
  - Driver may shut off monitoring system
  - Driver might block line of sight to monitoring system

- Driver Monitoring System raises ethical concerns
  - Potential invasion of privacy
  - Security data risk
  - Unethical collection of data

Lessons Learned

- Apollo Autonomous System is a complete system
  - Difficult to expand software to bring genuine value
  - Already achieved L4 autonomy

- Improvements are similar to those implemented by competitors (Tesla’s Autopilot and Cadillac’s Supercruise)
  - Both include eye-tracking and notification systems

Conclusion

- Driver-notification and driver-monitoring system introduce another layer of safety.
- Notification System is simple and cost effective however the Driver monitoring system is safer and more effective.
- New dependencies from HMI/Dreamview to guardian module, but high level architecture is still publish-subscribe style
- Low level architecture adopts changes in Perception and HMI/Dreamview subsystems (camera for eye-detection)

Conclusion

- Diagrams demonstrate interaction between perception and guardian modules in the event of driver becoming insensible.
- Testing procedures were discussed for the notification (prioritization) and camera systems.
- Potential risks of notification systems and driver-alert systems were explored.
- Group reflection highlighted difficulties in developing improvements for a completed system.

References

[1] Apollo Module Breakdown:
https://github.com/ApolloAuto/apollo/tree/master/modules

[2] PubSub Dependency graph:
https://onq.queensu.ca/d2l/le/content/642417/viewContent/3865686/View

[3] Previous year projects:
https://research.cs.queensu.ca/home/ahmed/home/teaching/CISC322/F18/index.html

[4] Apollo Understand Diagram:
https://docs.google.com/document/d/1qcHmRh1gAGTZMorCl1ILjomHqvamk6pjCgH2GdOogFs/edit?usp=sharing

[5] Apollo Prediction Subsystem Documentation
https://github.com/ApolloAuto/apollo/tree/master/modules/prediction

[6] Apollo Planning Subsystem Documentation
https://github.com/ApolloAuto/apollo/tree/master/modules/planning

[7] Apollo Control Subsystem Documentation
https://github.com/ApolloAuto/apollo/tree/master/modules/control

[8] Apollo Relative Map Documentation
https://github.com/ApolloAuto/apollo/tree/master/modules/map/relative_map

[9] Apollo Dreamview Subsystem Documentation
https://github.com/ApolloAuto/apollo/tree/master/modules/dreamview