Video Presentation Link

https://youtu.be/e5tFHqJj53k

Group Breakdown

Group Member	Contribution
Haadi Mufti (Group Leader)	Interactions, Lessons learned, Alternative styles
Emily Poon	Abstract, Architectural style and architecture diagram
Kevin Shroff (Presenter)	Component overview, Concurrency, Modifiability, Presentation
Oliver Cao	Introduction, Components overview
Gregory Secord	Use Case, Conclusion
Connor Colwill (Presenter)	Use Case, Slides, Presentation

Conceptual Architecture:
Apollo v7

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

Overview

- Introduction
- Derivation Process
- Alternative Styles
- Conceptual Architecture
- Subsystem overview
- Use Case
- Concurrency
- Limitations and Lessons Learned
- Conclusion

Introduction

- "High performance, flexible architecture for the development, testing, and deployment of autonomous vehicles"
- Developed by Baidu and Kinglong
- Apollo 1.0 released in 2017
- Apollo 7.0 introduces new deep learning modules for improved perception and prediction

Derivation Process

What does an autonomous vehicle system need?

- Multiple manipulated variables
- Reuse & Reintroduction of variables

Process Control Style:

- Autonomous vehicles require multiple inputs taken into consideration

Publish and Subscribe Style:

- Reuse of components and variables
- Numerous variables can be added to the system

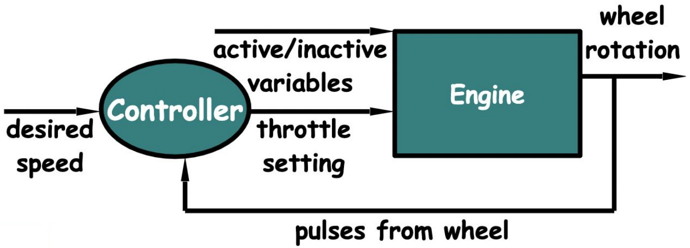


Fig. 1: Example Process Control from lecture slides

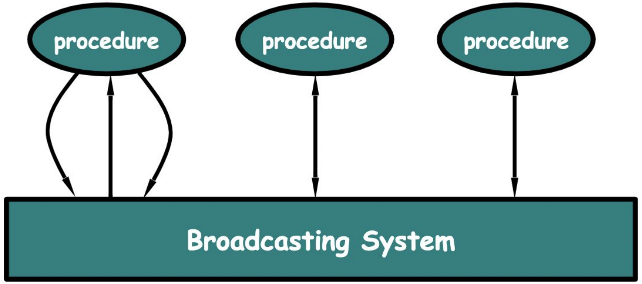
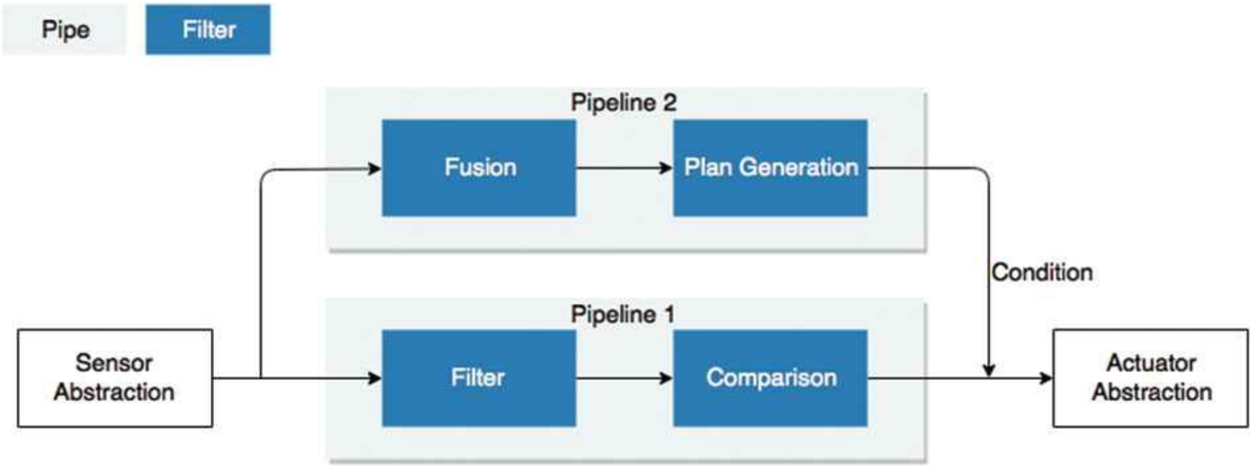


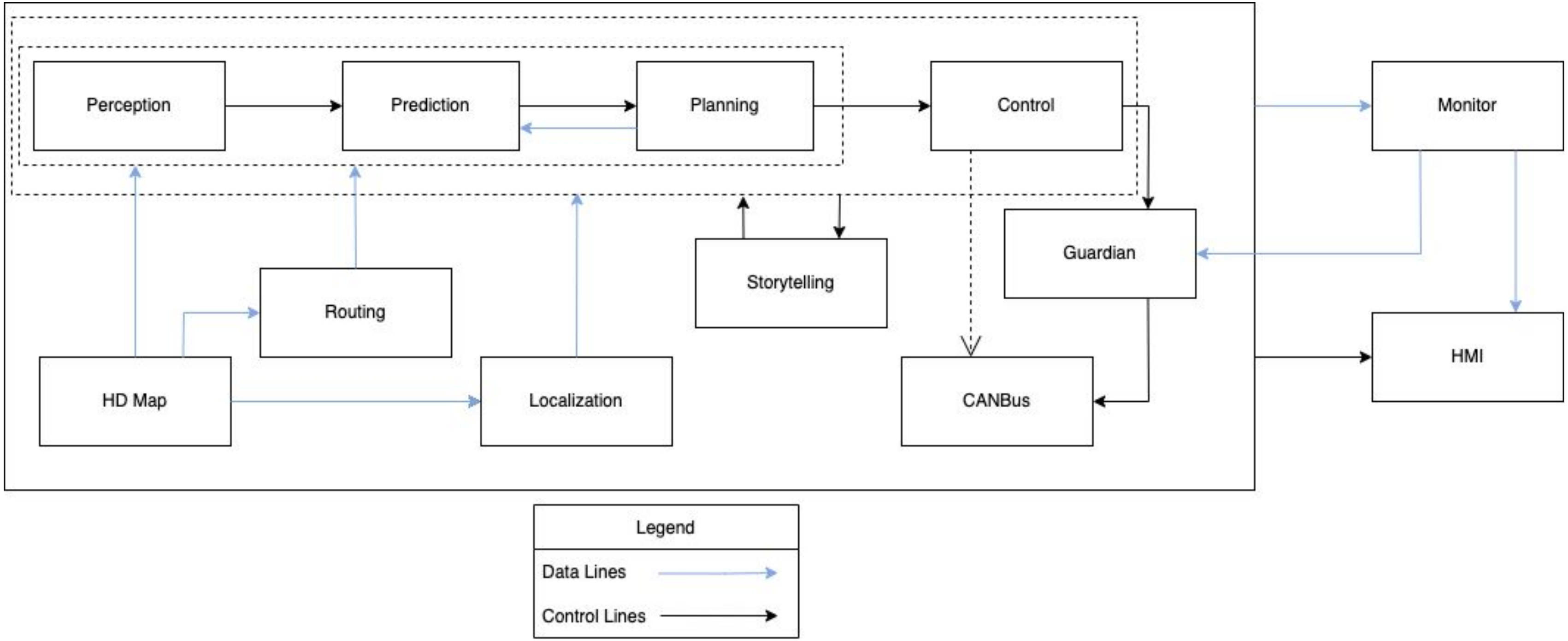
Fig. 2: Example Publish & Subscribe from lecture slides

Alternative Style: Pipe & Filter



- The steps in computations can represent different pipes
- Allows for a series of computations to be performed on sensor data

Conceptual Architecture



Subsystem Overview: Perception, Prediction

Perception:

- Interprets data gathered from sensors into concepts of surroundings
- Cameras, radars, LiDARs
- Obstacle Detection

Prediction:

- Predicting obstacle behaviour
- 4 main functions: Container, Scenario, Evaluator, Predictor
- Predicts and generates future trajectory path of objects

Subsystem Overview: Routing, Planning

Routing:

- Uses map data and routing requests
- Generates high level navigation information

Planning:

- Utilizes localization, perception, HD map, routing
- Computes a collision free comfortable trajectory

Subsystem Overview: Control, CANBus

Control:

- Uses trajectory and vehicle status to create a driving experience
- Passes instructions such as steering and brakes to CANBus

CANBus:

- Passing control commands to the hardware, and chassis information to the software
- Major components: Vehicle, CAN client

Subsystem Overview: HD-Map, Localization

HD-Map:

- Query engine to provide road information
- A module of maps

Localization:

- Computing and determining location
- Uses satellite information from GPS and LiDAR

Subsystem Overview: HMI, Monitor

HMI:

- Web application viewed and operated by user
- A UI for the current output of the main modules
- Allows for debugging, starting the car, hardware status, etc.

Monitor:

- Surveillance system of all modules operating in the system
- Watch the status of the system for hardware or module failure
- Passes data to the HMI

Subsystem Overview: Guardian, Storytelling

Guardian:

- Safety decisions
- Blocks control signals from being sent to CANBus in event of error

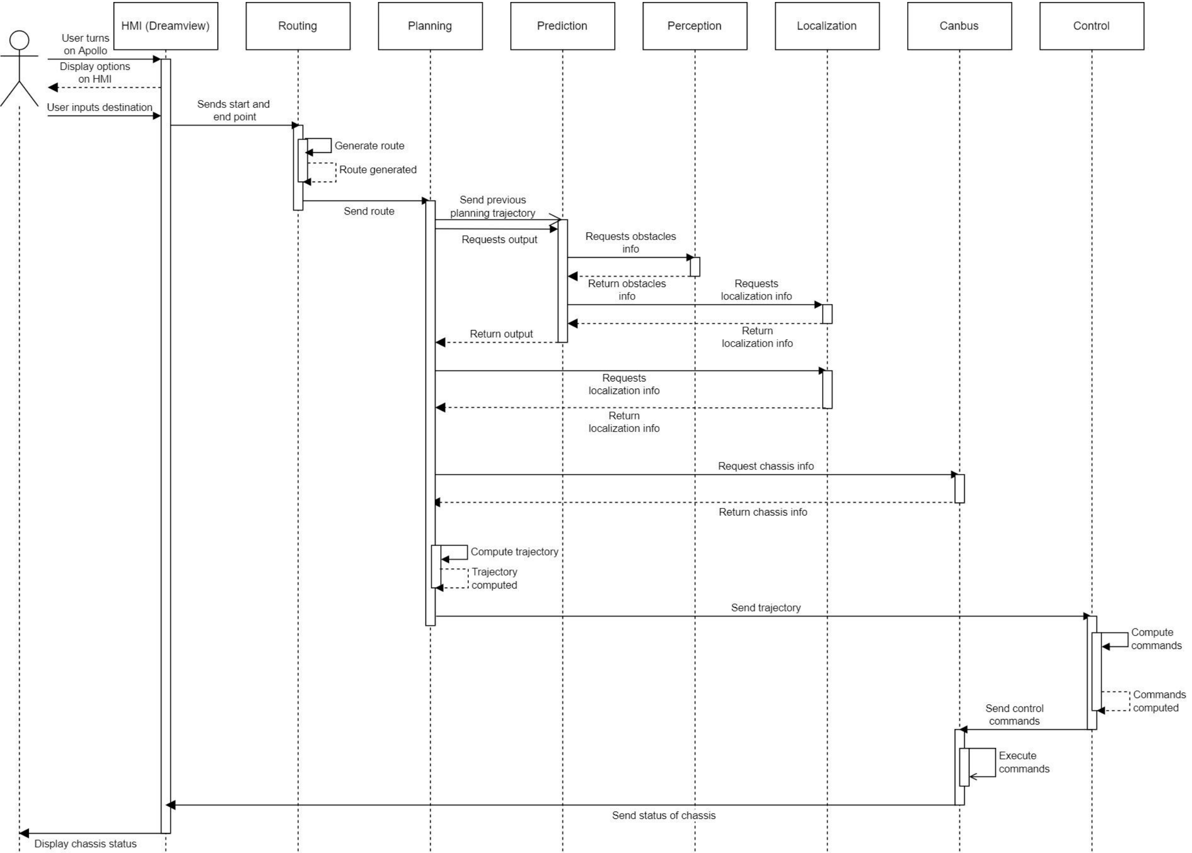
Storytelling:

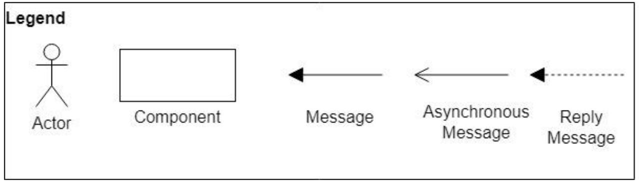
- Used to coordinate synchronous and cross-module actions from multiple modules
- Allows for tackling of complex driving scenarios
- Allows for driving scenario fine-tuning

Interactions

- Perception sends obstacle data to Prediction
- Localization receives information about roads from the HD-Map
  - This then passes to Prediction and Planning
- Planning and Prediction take data from each other to generate safe trajectory
- Storytelling takes localization and HD-map input and creates scenarios to send to Planning
- Control sends driving instructions to CANBus
- HMI and Monitor provide external support
  - They take data from all components to display
- Guardian receives a warning signal from Monitor to stop execution

Use Case


HMI (Dreamview)
Routing
Planning
Prediction
Perception
Localization
Canbus
Control
User turns on Apollo
Display options on HMI
User inputs destination
Sends start and end point
Generate route
Route generated
Send route
Send previous planning trajectory
Requests output
Requests obstacles info
Return obstacles info
Requests localization info
Return localization info
Return output
Requests localization info
Return localization info
Request chassis info
Return chassis info
Compute trajectory
Trajectory computed
Send trajectory
Compute commands
Commands computed
Send control commands
Execute commands
Send status of chassis
Display chassis status


Legend
Actor
Component
Message
Asynchronous Message
Reply Message

Concurrency

- Concurrency is heavily present
- Apollo requires real-time input:
  - LiDARs, GPS, Cameras
- Components interact with each other to perceive, predict, plan, localize, and control the vehicle

Limitations and Lessons Learned

Limitations:

- Information from Apollo did not always specify version
- Hard to find other architectural styles

Lessons Learned:

- Autonomous driving components were consistent across sources
- Apollo’s GitHub had extensive documentation
- How to work as a group remotely

Conclusion

- Apollo uses a process control style
- Subsystems include:
  - Perception, prediction routing, planning, control, CANBus, HD-Map, Localization, HMI, Monitor, Guardian, Storytelling
- There is high interactivity and concurrency in the system

References

- [1] Apollo Module Breakdown: https://github.com/ApolloAuto/apollo/tree/master/modules
- [2] Reference Architecture: https://onq.queensu.ca/d2l/le/content/642417/viewContent/3814366/View
- [3] Apollo Software Architecture: https://github.com/ApolloAuto/apollo/blob/master/docs/specs/Apollo_5.5_Software_Architecture.md
- [4] Autonomous Vehicles Software Architecture: https://encyclopedia.pub/9379
- [5] Previous year projects: https://research.cs.queensu.ca/home/ahmed/home/teaching/CISC322/F18/index.html
- [6] External Information About Apollo: https://www.mo4tech.com/analysis-of-baidu-apollo-autonomous-driving-platform.html
- [7] Alternative architecture https://www.atlantis-press.com/journals/jase/125934832/view#sec-s4