Group 8 - Artemis

Proposed Feature Enhancement

Video Link: https://www.youtube.com/watch?v=1q860ltBobo

Team Artemis

- **Josh Otten** - Use Cases, SAAM Analysis
- **Aleks Jugovic** - Alternatives, Current State, Use Cases, Presenter
- **Muyun Yang** - SAAM Analysis, Diagrams
- **Chong Guan** - Abstract, Introduction, Conclusion
- **Daniel Jang** - Proposed Enhancement, Alternatives, Motivation, Presenter
- **Wooseok Lee** - Team Leader, SAAM Analysis, Potential Risk, Test Plan, Lessons Learned

Agenda

1. Introduction
2. New Feature Overview
3. Alternatives Considered
4. SAAM Analysis
5. Conceptual Architecture Changes
6. Risks and Limitations
7. Testing Plan
8. Concurrency
9. Use Cases and Sequence Diagrams
10. Lessons Learned
11. Conclusion

Introduction

Apollo

- Proposing an enhancement of the Apollo architecture
- SAAM Analysis
- Effects of the enhancement on Apollo
- Test plan and risk analysis

New Feature Overview

Guardian

- Enhancing Guardian’s stopping scenarios
- Cache previous data from other subsystems
- Attempt to bring the vehicle to a stop in a safer way
- Stopping method based on the module failure

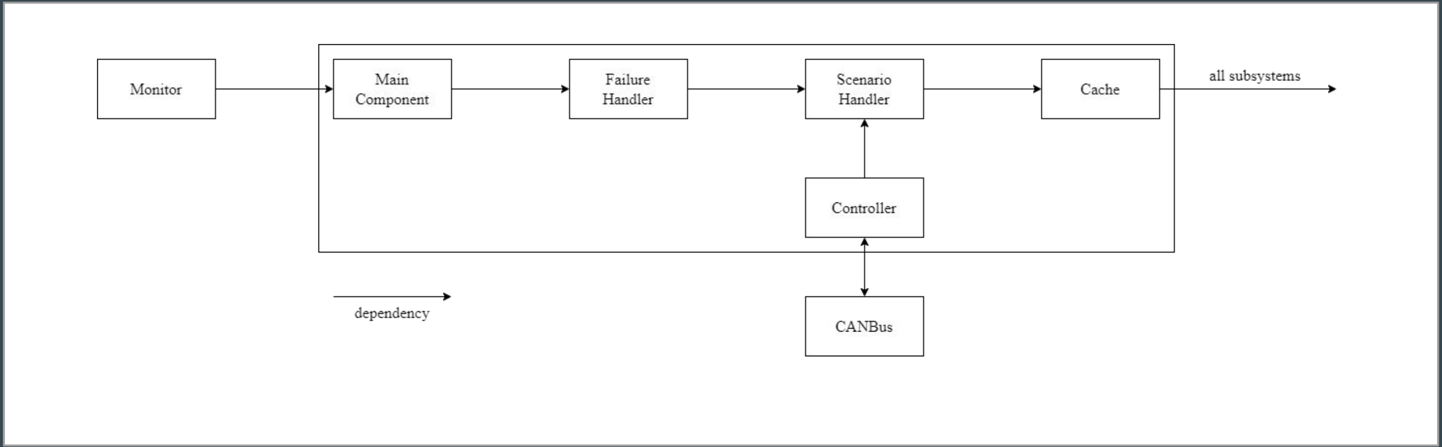
Motivation

- Guardian stops car when a module fails
- Safety concerns
- Stopping in traffic
- Room for improvement

Alternatives Considered

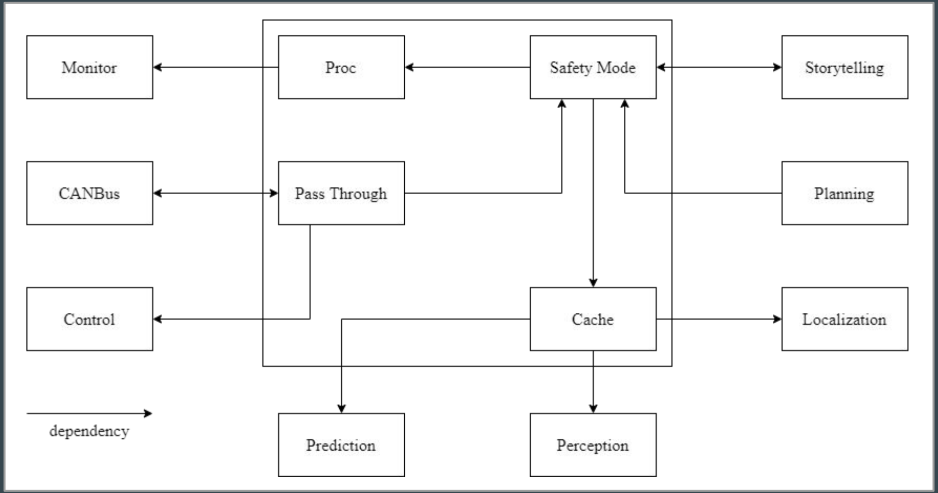
Approach A

- Object oriented architectural style
- Caching data from all subsystems in Apollo
- Different stopping scenarios based on failure reason



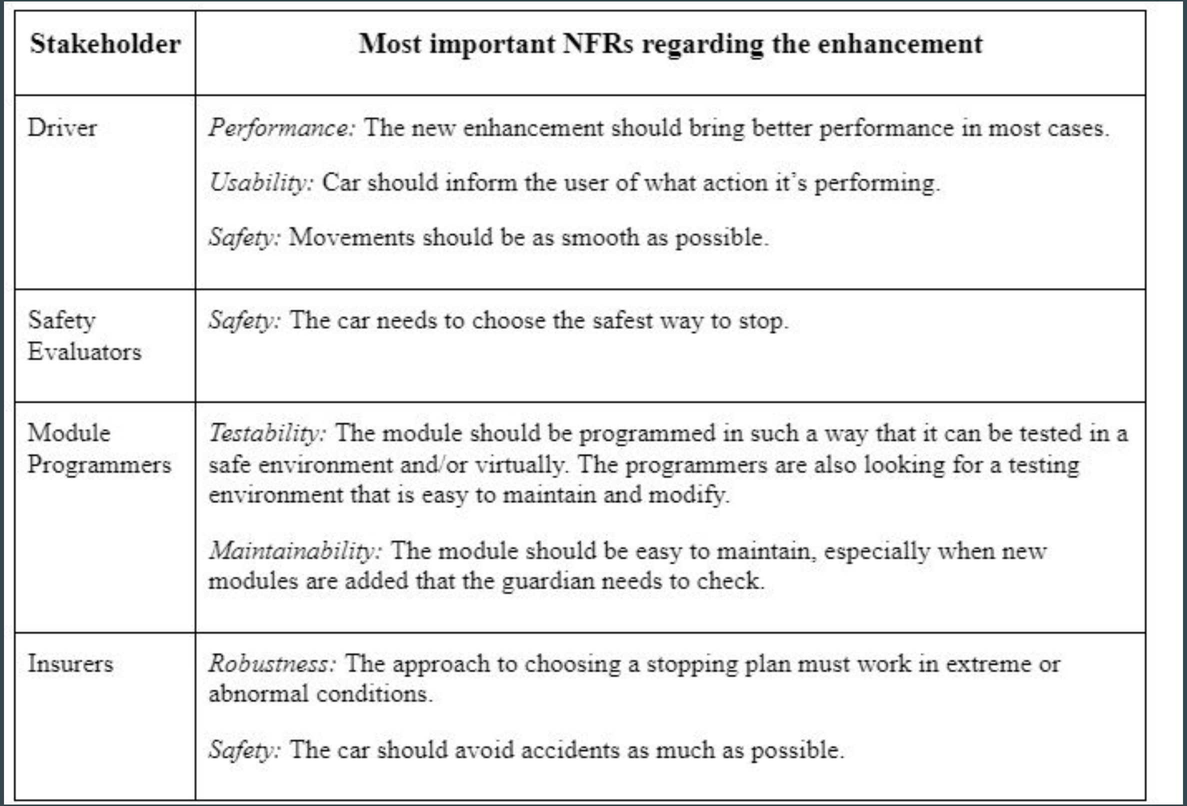
Approach B

- Publish-Subscribe architectural style
- Caching data from Localization, Perception, and Prediction
- Stopping scenario based on risk level of failure
- Attempt to create a new trajectory to bring the car to a stop



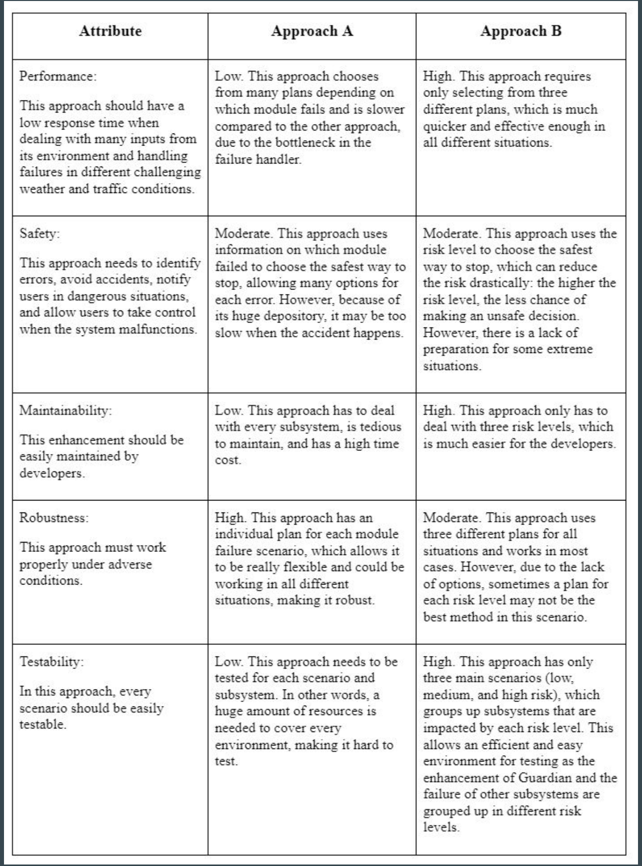
SAAM Analysis

SAAM Analysis - Stakeholders



Stakeholder	Most important NFRs regarding the enhancement
Driver	*Performance:* The new enhancement should bring better performance in most cases. *Usability:* Car should inform the user of what action it's performing. *Safety:* Movements should be as smooth as possible.
Safety Evaluators	*Safety:* The car needs to choose the safest way to stop.
Module Programmers	*Testability:* The module should be programmed in such a way that it can be tested in a safe environment and/or virtually. The programmers are also looking for a testing environment that is easy to maintain and modify. *Maintainability:* The module should be easy to maintain, especially when new modules are added that the guardian needs to check.
Insurers	*Robustness:* The approach to choosing a stopping plan must work in extreme or abnormal conditions. *Safety:* The car should avoid accidents as much as possible.

SAAM Analysis - Attributes

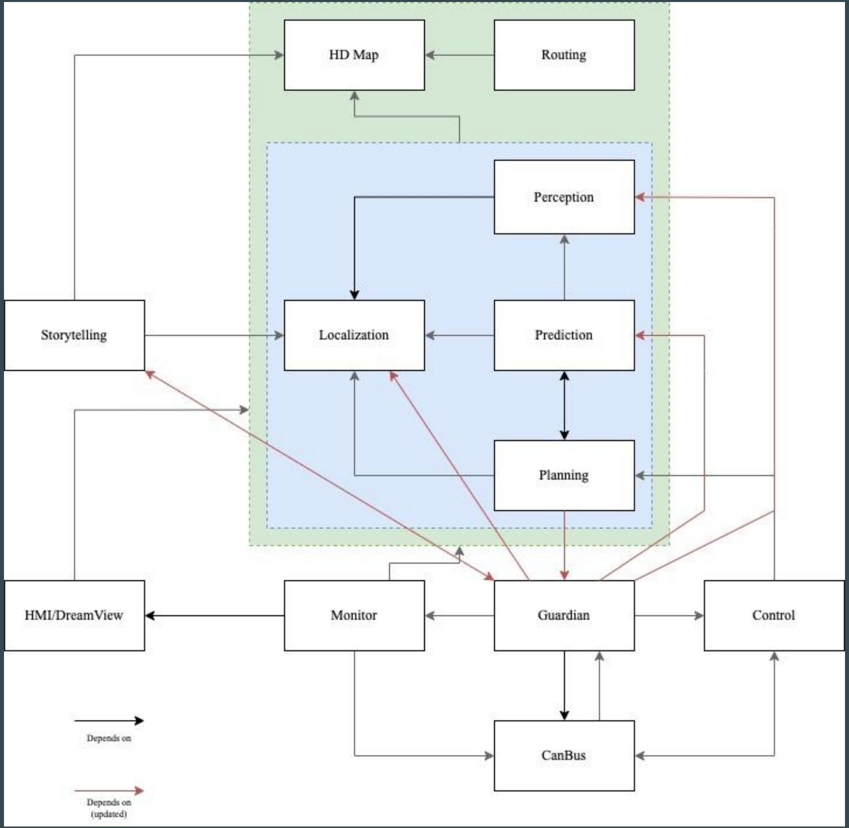


Attribute	Approach A	Approach B
Performance: This approach should have a low response time when dealing with many inputs from its environment and handling failures in different challenging weather and traffic conditions.	Low. This approach chooses from many plans depending on which module fails and is slower compared to the other approach, due to the bottleneck in the failure handler.	High. This approach requires only selecting from three different plans, which is much quicker and effective enough in all different situations.
Safety: This approach needs to identify errors, avoid accidents, notify users in dangerous situations, and allow users to take control when the system malfunctions.	Moderate. This approach uses information on which module failed to choose the safest way to stop, allowing many options for each error. However, because of its huge depository, it may be too slow when the accident happens.	Moderate. This approach uses the risk level to choose the safest way to stop, which can reduce the risk drastically: the higher the risk level, the less chance of making an unsafe decision. However, there is a lack of preparation for some extreme situations.
Maintainability: This enhancement should be easily maintained by developers.	Low. This approach has to deal with every subsystem, is tedious to maintain, and has a high time cost.	High. This approach only has to deal with three risk levels, which is much easier for the developers.
Robustness: This approach must work properly under adverse conditions.	High. This approach has an individual plan for each module failure scenario, which allows it to be really flexible and could be working in all different situations, making it robust.	Moderate. This approach uses three different plans for all situations and works in most cases. However, due to the lack of options, sometimes a plan for each risk level may not be the best method in this scenario.
Testability: In this approach, every scenario should be easily testable.	Low. This approach needs to be tested for each scenario and subsystem. In other words, a huge amount of resources is needed to cover every environment, making it hard to test.	High. This approach has only three main scenarios (low, medium, and high risk), which groups up subsystems that are impacted by each risk level. This allows an efficient and easy environment for testing as the enhancement of Guardian and the failure of other subsystems are grouped up in different risk levels.

Conceptual Architecture Changes

Conceptual Architecture Changes

- 5 new dependencies added to the conceptual architecture
- Guardian now subscribes to Prediction, Localization, Perception, and Storytelling
- Storytelling and Planning subscribe to Guardian



Risks and Limitations

Risks and Limitations

- Unreliable safety of stopping trajectory
- Computation latency
- Availability
- Potential data transfer bottleneck

Testing Plan

Testing Plan

- *Low Risk Use Case:* Sub-test cases for each low risk stopping scenario. Ensure car completes the stop it was supposed to

- *Medium Risk Use Case:* Test accuracy and safety of cached data from failed subsystem

- *High Risk Use Case:* Similar to original Guardian testing, ensure car comes to complete stop when critical modules fail

Concurrency

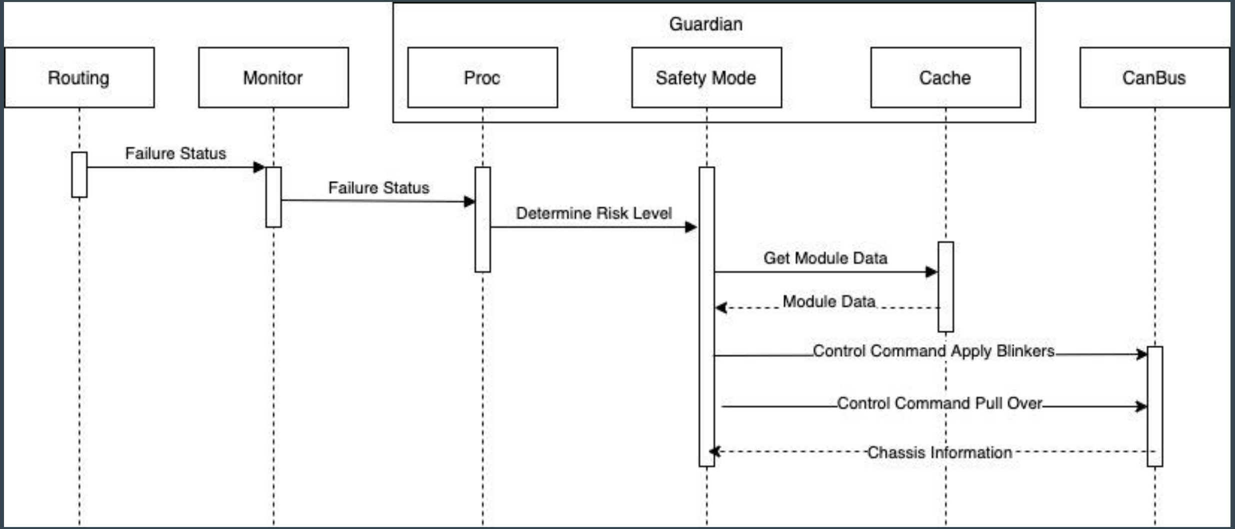
Concurrency

- Concurrency of several functionalities
- Caching data from other subsystems
- Passing through Control signals
- Scanning for module failures

Use Case - Low Risk Emergency Stop

Use Case - Low Risk Emergency Stop

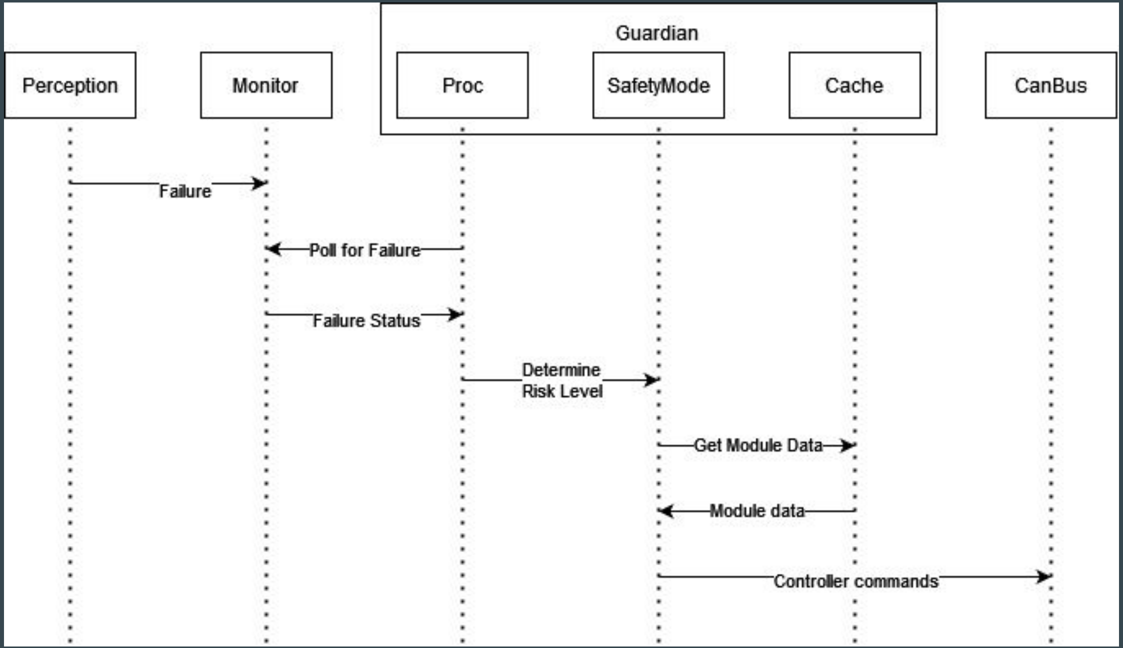
- Routing module failure
- Failure status detected by Guardian
- Low risk assessed
- Stopping trajectory generated by SafetyMode



Use Case - Medium Risk Emergency Stop

Use Case - Medium Risk Emergency Stop

- Perception module failure
- Failure status detected by Guardian
- Medium risk assessed
- Stopping trajectory generated by SafetyMode using cached data



Lessons Learned and Limitations

Limitations and Lessons Learned

- Little reference material for new enhancement

- Enhancing a subsystem with little existing functionality

- Completing a SAAM analysis

Conclusion

Conclusion

- Proposed enhancement to Guardian functionalities
- SAAM Analysis performed to pick best alternative
- Impacts on the architecture and testing
- Use cases and lessons learned