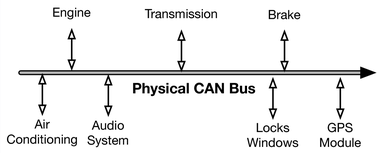
Poster: Securing Appified Autonomous Vehicles Platform With AVGUARD

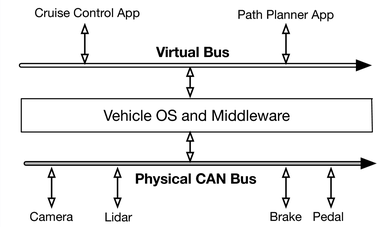
Yunhan Jack Jia, Ding Zhao, Qi Alfred Chen, Z. Morley Mao
University of Michigan, Ann Arbor

Appified AV Platform

- ***Appified*** autonomous vehicle (AV): self-driving functionalities are developed in a crowd-sourcing manner and installed as modules
- Traditional vehicle architecture:



  - Outsourcing
  - Hardware suppliers
  - Heavy testing work
  - Usually takes over 24 months
- Appified AV architecture:

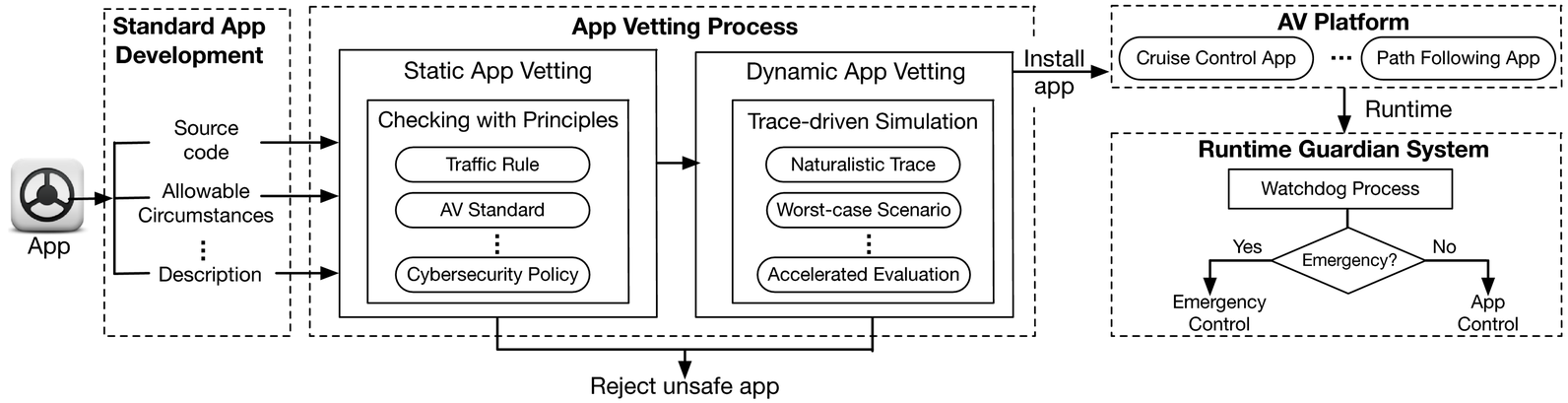


  - Crowd-sourcing
  - Software suppliers
  - Software testing
  - High requirement on the functionality

Example: Path-Following AV App on PolySync

```
1  class PathFollowingNode
2  {
3      //Subscribe on the vehicle report
4      registerListener("VEHICLE");
5      //Load the trajectory data
6      map = loadMap("map.dat");
7
8      void MessageHandler(Message msg)
9      {
10         Position pos = msg.position;
11         for (i;i<map.length();i++){
12             //Calculate distance and heading errors
13             _distError = getError(pos,map[i]);
14             _headingError = getError(pos,map[i]);
15
16             Message message  = new Message();
17             //If vehicle is in path, continue wih
18             //the adjusted steering angle
19             if(_distError<MAX_DIST_ERR &&
20                 _headingError<MAX_HEADING_ERR)
21             {
22                 message.setSterringAngel(_angel);
23                 message.publish();
24             }
25             //If vehicle is not in path, perform
26             //emergency stop with maximum throttle
27             else{
28                 message.setThrottleGain(100);
29                 message.publish();
30                 break;
31             }
32         }
33     }
34 }
```

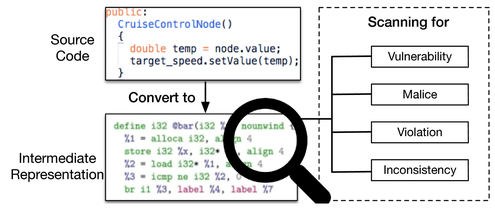
Overview: AVGuard Approach



Step One: Standard App Development

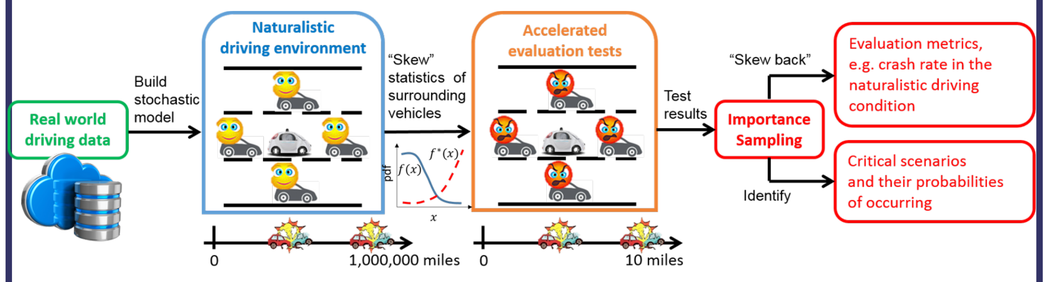
- Facilitate app vetting by requiring developers to provide
  - **Purpose statements**
    - Matching user expectation with app behavior; Justifying access control decisions
  - **Required resources**
    - E.g. exclusive usage of throttle pedals, radar, or required network bandwidth
  - **Usage constraints**
    - Required by the DoT standard for autonomous driving functionalities
    - E.g., high-way only, sunny day only (with clear camera vision)

Step Two: Static App Vetting



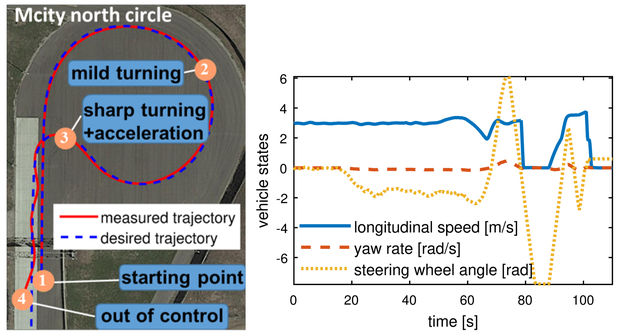
Step Three: Dynamic App Vetting

- Accelerated Evaluation Approach
  - **Based on importance sampling techniques**
  - **Quantifies potential risks of AV apps, and runs off-the-vehicle.**
  - **Reduces the required test mileages for each analysis by a factor of 10,000 to 100,000.**



Case Study & Discussion

- Dynamic analysis revealing potential risks of the path-following AV app



- Current appification status in AV field
  - **Industry: pioneering companies to open critical access of vehicles to researchers**
    - e.g., Ford XC, PolySync
  - **Academia: U-M to provide OpenAV to researchers; Udacity courses to targets for developing library of open source AV functionalities.**
- Ongoing Work
  - Complete the implementation of AVGuard to be used for students taking Automated Vehicle Control course in 2017 provided by U-M
  - Collete rules and principles to be added to the static app vetting process using crowd-sourcing.
  - Build database for common faults made by developers