Model-Based Design Approach for Validation of Vehicle Longitudinal Control Algorithm

Jordan Olson[1], Brandon Stevens[1], Ashley Phan[1], Hwan-Sik Yoon[1#] and Paul Puzinauskas[1#]

[1]The University of Alabama
[#]Advisor

ABSTRACT

In this paper, model-based testing strategies are described for the validation of an Adaptive Cruise Control (ACC) algorithm developed for a 2019 Chevrolet Blazer as part of the EcoCAR Mobility Challenge. A team of undergraduate and graduate students developed testing procedures to assess model fidelity, and to identify and resolve issues with the algorithm before deployment to a student-modified production vehicle. The algorithm validation is conducted via three progressive levels of validation environments: Model-In-the Loop (MIL), Hardware-In-the-Loop (HIL), and Driver-In-the-Loop (DIL). When the ACC algorithm is evaluated using system requirements in the testing sequence, the MIL environment performs the tests at least 87% faster than the HIL environment. The MIL environment can also utilize parallel computing, which leverages multi-core CPUs to conduct multiple simulations simultaneously. Although comparisons between MIL and HIL results revealed good agreements, slight differences in system dynamics highlights a need for future Vehicle-In-the-Loop (VIL) testing. By showing how the concepts can be applied to the validation of an autonomous feature in a vehicle with detailed test scenarios and evaluation metrics, the paper will serve as a good reference for the students and engineers interested in this field.

Introduction

Background

Transportation systems are rapidly evolving these days with advanced propulsion systems and autonomous features pushing towards a future of “zero crashes, zero emissions, and zero congestion” (General Motors, 2021). A new service format called Mobility as a Service (MaaS) is also becoming more convenient and affordable for consumers due to its integration with the connected world (Gindrat, 2018). Personal MaaS is currently a niche market but will likely dominate the urban transportation sector in the near future with projected compounded annual growth rates of 31.5% over the next few decades (MarketsandMarkets, 2021).

For over 30 years, the U.S. Department of Energy (DOE) has sponsored Advanced Vehicle Technology Competitions (AVTCs) managed by Argonne National Laboratory (ANL) in partnership with the automotive industry (Britt and Shoults, 2021). The current competition series is the EcoCAR Mobility Challenge (EMC), which is supported by General Motors (GM), MathWorks, DOE, and over 25 other industry sponsors. The EMC tasks the participating 11 university teams with developing and implementing Level 2 autonomous features as well as integrating and optimizing an advanced propulsion system for a 2019 Chevrolet Blazer. The Society of Automotive Engineers (SAE) defines Level 2 autonomous features as those that enable steering and braking/acceleration to support the driver (SAE International, 2018).

The University of Alabama (UA) team consists of 65 students ranging from the first-year undergraduate students to PhD candidates. Five graduate students are fully funded through the program to lead student learning

activities and engineering exercises to accomplish competition goals. Most students participate without receiving formal course credit, but several academic courses have been developed and adapted from ECM activities at UA. Students are recruited to join the engineering team at the start of each academic year across all majors within the College of Engineering. Figure 1 shows the demographic makeup of the UA team.

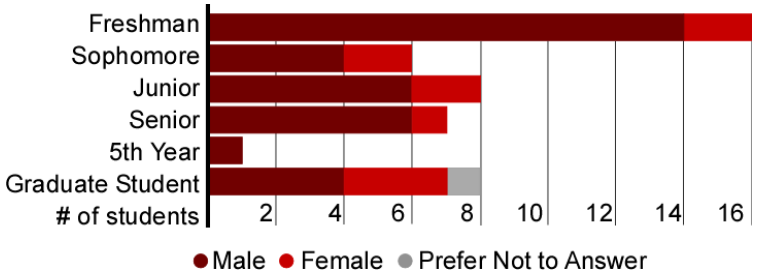

Figure 1. UA Team demographics

The four-year EMC program, which started in the fall of 2018, closely follows GM's global vehicle development plan. Each participating team defined their own vehicle technical specifications (VTS) driven by the needs of their unique market for a fleet-owned and customer-driven vehicle in the MaaS economy. In order to achieve the VTS goals, new hybrid propulsion systems were designed, and sensors and computational hardware were selected and integrated for the connected and autonomous vehicle (CAV) features.

One of the CAV features being developed by the UA team is the adaptive cruise control (ACC). ACC enables a vehicle to maintain a desired speed while keeping a safe distance and relative velocity from other vehicles in its path. In this paper, the development and testing processes of the ACC algorithm developed by the UA student team are described in detail along with the educational process that enabled the student team success. Based on the architecture of the model-based design, these processes evaluate the performance of the algorithm across three progressive levels of testing and validation environments: Model-In-the-Loop (MIL), Hardware-In-the-Loop (HIL), and Driver-In-the-Loop (DIL).

Recently, these model-based testing and validation environments have been widely used in the automotive system development. For example, Gowda *et al.* (2018) applied MIL, HIL, and DIL as well as Software-In-the-Loop (SIL) in developing a hydraulic antilock braking system. Similarly, Barale *et al.* (2014) applied those testing environments to comfort analyses of an active hydraulic suspension system. MIL, SIL, HIL, and DIL have also been utilized to develop Advanced Driver Assistance Systems (ADAS) and autonomous vehicles by many researchers (Abdelgawad *et al.*, 2014; Pikus and Was, 2019; Feng *et al.*, 2020; Allen *et al.*, 2020; Muhammad *et al.*, 2019).

Despite the recent surge of interest in this field, most of the existing publications are focused on the presentation of different physical setups and corresponding general procedures. As a result, it is difficult to find a detailed application case with comparison data for different testing environments. To fill the knowledge gap, this paper presents how an ACC system can be developed and tested in three different levels of testing environments with detailed simulation and comparation data. Test results are analyzed to improve the algorithm, forming a closed-loop, iterative development process that spans from initial requirements through vehicle deployment.

Student Education Procedure

One of the goals of the EcoCAR Mobility Challenge is to provide all participating students with industry-led training opportunities so that the students learn the technical skills needed to complete competition objectives and deliverables (Britt and Shoults, 2021). These training topics include Hardware-In-The-Loop, hardware debugging, technical requirement development, vehicle testing and test plans, and data collection. In addition, the graduate students leading the UA team have built technical onboarding modules and hosted bi-weekly training sessions to provide knowledge

for the operation of cross-functional student teams. These topics include automotive standards, modeling and simulation of automotive systems, operation of hybrid- and battery- electric vehicles, model-based design, software verification and validation, agile process methodology, and overviews of various software tools.

Since the development of the ACC controller primarily occurred during the COVID-19 pandemic, all student development and training took place virtually. Training was conducted and recorded over online tools that allowed any interested student on the team to participate. Recordings of these training sessions will allow for their re-use in later years of the competition for incoming new students. The feedback from the participating students on the internal team training was generally positive, with students specifically citing an increased comfort level with learning new technical topics. It is also notable that the students view the virtual platform as a positive tool helping them attend the training in a more convenient manner. Anonymous feedback was also gathered from students by ANL which showed that 10% of the students rated the industry training "Not at all" or Slightly" valuable, 24% rated the training "somewhat" valuable, and 66% rated the trainings "moderately" or "extremely" valuable (Britt and Shoults, 2021).

Description of System and Approach

As shown in Figure 2, the controlled vehicle is known as the ego vehicle, and other vehicles in its path are known as target vehicles. In the figure, v_{ego} and v_{target} are the velocities of the ego and target vehicles, respectively, and d_{rel} and v_{rel} are the distance and velocity of the target vehicle relative to the ego vehicle. The ACC system requires accurate detection of nearby vehicles, which is achieved by using ranging sensors such as cameras and radars.

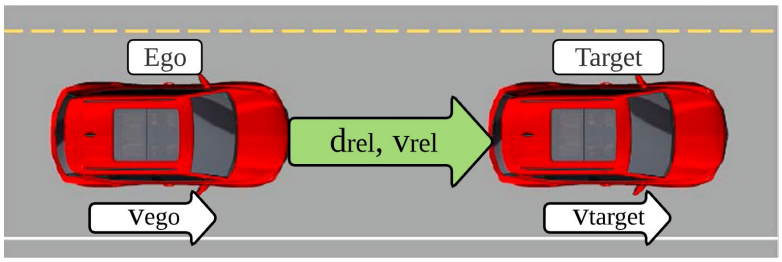

Figure 2. Definition of ego and target vehicles

Model-Based Design Development Process

Model-based design (MBD) is a model-centric approach to the development of dynamic systems (Aarenstrup, 2015). A core concept of MBD is continuous testing at all stages of development, which can help identify issues early and increase overall understanding of the system. This concept can be represented by a diagram known as the V-cycle. A version of the V-cycle adapted by the UA team is shown in Figure 3.

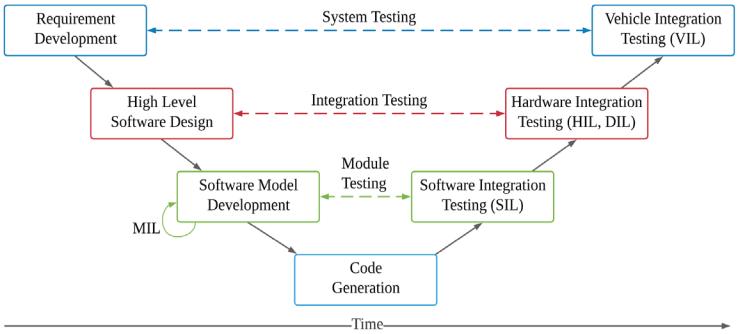

Figure 3. V-cycle for the development of UA CAV system

When a new feature is developed, a set of requirements are defined first, and the feature is progressively developed until it is ready to be tested in a simulation environment via Model-In-the-Loop (MIL) testing. Once its functionality has been confirmed in the simulation environment, the feature is tested in hardware progressively via the Software-In-the-Loop (SIL), Hardware-In-the-Loop (HIL), Driver-In-the-Loop (DIL), and finally Vehicle-In-the-Loop (VIL) environments. Different testing capabilities of these environments are summarized in Table 1. The V-cycle illustrates a practical progression of new features from development to deployment.

Table 1. Comparison of different testing environments

Testing capabilities	MIL	SIL	HIL	DIL	VIL
Faster than real time	✓	✓			
Automated/iterative testing	✓	✓	✓		
Testing of unsafe situations	✓	✓	✓	✓	
Real-time execution		✓	✓	✓	✓
Generated code		✓	✓		✓
Communication interface			✓		✓
True vehicle response					✓
Driver acceptance				✓	✓

Hardware Architecture

The UA team's CAV perception system, as shown in Figure 4, consists of Intel's Mobileye 6 as the front camera-based vision system, one Bosch mid-range radar (MRR) in the front, and two Bosch rear MRRs placed at the rear corners. Table 2 lists the detection range and field of view (FOV) specifications for the Intel Mobileye camera and Bosch MRRs. Figure 5 shows the FOV of the front perception system using the Mobileye and the front radar's two antennas: a main antenna and an elevation antenna.

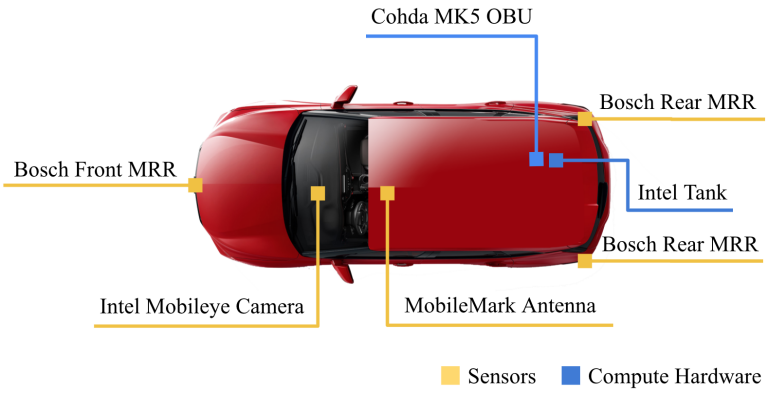

Figure 4. UA CAV system hardware

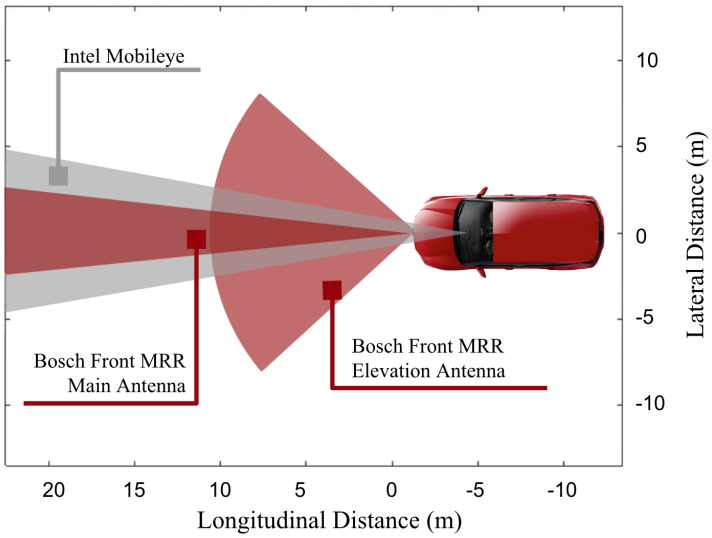

Figure 5. FOV of the front perception system

The perception hardware interface with a controller, Intel's IEI Tank 870-Q170, which is a high-performance embedded computer running a Linux operating system. As the CAV controller, the Tank performs data logging and parsing as well as executing the sensor fusion and tracking algorithm and the ACC algorithm.

Table 2. Front perception system sensor specifications

Sensor	Field of View (FOV)	Detection Range
Intel Mobileye 6	Width: 38° Height: 30°	150 m
Bosch Front MRR	Main antenna: 12° Elevation antenna: 84°	160 m 12 m
Bosch Rear MRR	150°	80 m

Adaptive Cruise Control Algorithm

Functionality Expectations

The high-level expectations for the ACC system were conceived by the UA team as follows:

1. When a target vehicle is detected and traveling at a slower speed than the ego vehicle's set speed, operate in distance control mode (following mode).
2. When no target vehicle is detected, or the target vehicle speed is greater than the ego vehicle's set speed, operate in velocity control mode.
3. Respond safely and quickly to dangerous situations (e.g., target vehicle cuts into the ego vehicle's lane).
4. Optimize the system control for fuel economy while maintaining acceptable driver comfort and system safety.

These expectations are used to identify specific system-level requirements that define the ACC system's minimum viable product (MVP). The MVP defines the mandatory validated capabilities the system must possess to deem it acceptable for use. The system's technical specification includes additional requirements that add desirable features well beyond the MVP, but an absence of these additional features does not disqualify the system for use. Only the requirements directly defining the MVP are addressed herein, and they are listed in Table 3. A requirement ID with the prefix "SR" denotes a safety requirement developed by the UA team's systems safety working group, while a prefix "OR" denotes an operational requirement developed by the ACC development team.

Table 3. Definition of ACC system requirements

ID	Requirement Description
SR.50.100	An ACC system command shall not result in an acceleration lower than -4.90 m/s^2 under any circumstance (Wu *et el.*, 2009).
SR.50.110	An ACC system command shall not result in a vehicle speed below 0 m/s or above 36 m/s.
OR.50.100	An ACC system command shall not result in an acceleration lower than -2 m/s^2 unless the time to collision (TTC) falls below 4 s (Sultan and McDonald, 2003).
OR.50.110	An ACC system command shall not result in an acceleration greater than 2 m/s^2 (Bae *et al.*, 2019).
OR.50.150	An ACC system command shall not result in an acceleration rate of change exceeding +/- 0.9 m/s^3 unless the time to collision (TTC) falls below 4 s (Barnes *et al.*, 2020).

The time-to-collision (TTC) referred to in the table is defined by: $TT = d_{rel}/v_{rel}$
where,
d_{rel} = distance of target vehicle relative to ego vehicle
v_{rel} = velocity of target vehicle relative to ego vehicle

Algorithm Structure

The ACC algorithm, along with a sensor fusion algorithm, was developed in MathWorks' Simulink environment, which allowed for a simple interface between the two algorithms. The high-level structure of the ACC system is shown in Figure 6. The ACC algorithm uses a model predictive control (MPC) strategy, taking relative distance and velocity as controller inputs and generating an acceleration command. This acceleration command is then mapped to a wheel torque command which is fed to the hybrid supervisory controller (HSC). The HSC then controls the vehicle propulsion actuators to accelerate or decelerate the vehicle.

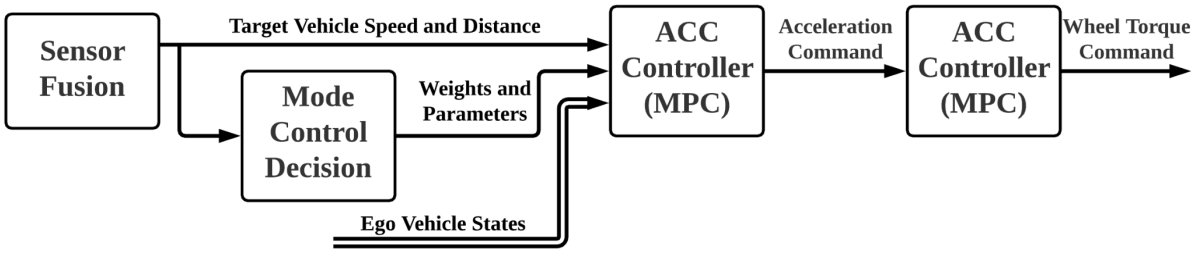

Figure 6. ACC controller architecture

The ACC algorithm is primarily tuned by changing controller weights, which affect how much different parameters are prioritized during optimization. These parameters are described in Table 4, along with their initial values obtained from development work in the previous year. The controller weights are gain-scheduled into different operating regions, where the first value represents the weight in distance control mode (region I) and the second value represents the weight in velocity control mode (region II).

Table 4. ACC controller parameters

Metric	Definition	Parameter Description	Initial Weight	
			I	II
Ego Acceleration	$a_{ego} = dv_{ego}/dt$	Magnitude of ego vehicle acceleration	0.2	0.3
Ego Jerk	$j_{ego} = da_{ego}/dt$	Time rate of change of ego vehicle acceleration	0.01	0
Distance Error	$\Delta d = v_{rel} \cdot \tau + 10$	Error between relative distance and preferred distance	0.02	0
Velocity Error	$\Delta v = v_{target} - v_{ego}$ or $\Delta v = v_{set} - v_{ego}$	Error between ego velocity and target velocity (if in following mode) or set speed (if not in following mode)	0.1	0.1

The variables in Table 4 are defined as follows:

a_{ego} = acceleration of ego vehicle
v_{ego} = velocity of ego vehicle
j_{ego} = time rate of change of the acceleration (jerk) of ego vehicle
Δd = distance error
τ = driver-selected time gap
Δv = velocity error

Algorithm Testing

Description of Testing Environments

All the testing environments in Table 1 were considered for use in ACC testing, and they offer different benefits for ACC development and testing. MIL offers the advantage of automated testing that can be quickly iterated, allowing for rapid validation of requirements over multiple complete episodes. HIL tests the communication interface and computation ability on target hardware, making it essential for validation. DIL testing adds the benefit of qualitatively testing driver acceptance of the ACC feature. SIL testing offers no unique advantages to ACC when MIL and HIL are used and is therefore eliminated from the testing sequence. Because of the current inability to reliably command acceleration or braking on the team vehicle, VIL testing is also eliminated until vehicle capabilities are expanded. Therefore, the environments selected for the ACC testing are MIL, HIL, and DIL. This testing sequence is shown in Figure 7.

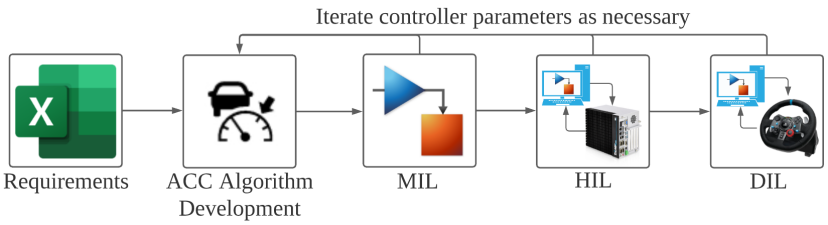

Figure 7. Testing sequence of the ACC algorithm

Requirements written by UA's ACC team and the systems safety working group are placed in a shared Excel document, which is then periodically imported to Simulink and converted to Simulink requirements. These requirements are used by the ACC team to add functionality and diagnostics to the algorithm, which are then pushed down the pipeline for testing.

Testing Environment Setups

Model-In-the-Loop (MIL) Testing

MIL testing is carried out entirely in Simulink. The MIL setup consists of models of the ACC controller code, the HSC code, a high-fidelity vehicle plant model, and a driving scenario source. The HSC is part of the UA team's vehicle system and sends forward and braking propulsion commands to the high-fidelity plant model actuators. Including it as part of MIL testing better captures the real response of the system to ACC commands. This setup is shown in Figure 8.

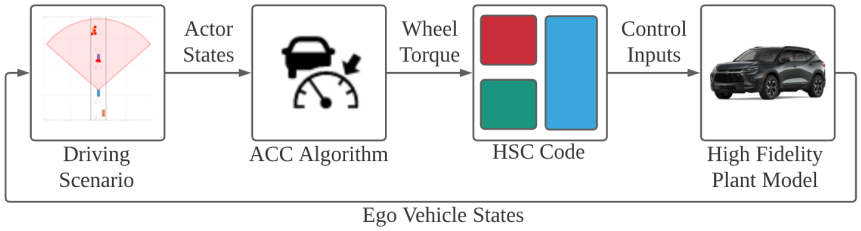

Figure 8. MIL testing configuration for ACC

The driving scenario source simulates one or more non-controlled vehicles, or actors. These actors follow predetermined paths, while the location and velocity of the ego vehicle are updated based on the vehicle dynamics information received from the plant model. The longitudinal distance and velocity of each actor relative to the ego vehicle are reported back to the ACC controller, as well as each actor's lane assignment. The ego vehicle is updated in the scenario based on the plant model outputs, resulting in a closed-loop system. The interface between the CAV controller and HSC is made to mimic its physical counterpart to accurately assess system performance and aid in the transition to hardware.

Hardware-In-the-Loop (HIL) Testing

The HIL environment consists of a desktop PC and the CAV controller which communicate via a Controller Area Network (CAN) bus, as it would in the team vehicle. This CAN bus connection is achieved using a Kvaser Leaf Light v2, which bidirectionally connects the desktop PC to a CAN bus network where the CAV controller can also

communicate. These hardware modules add J1939 high speed CAN bus connectivity and are easily integrated with MathWorks' Vehicle Network Toolbox. The HIL setup is shown in Figure 9.

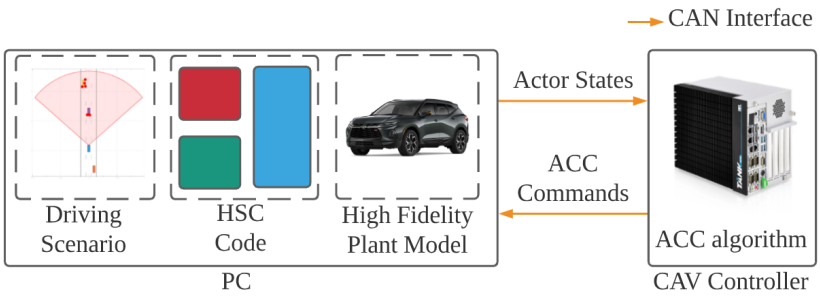

Figure 9. HIL testing configuration for ACC

The HIL setup is similar to the MIL setup except that the CAV controller runs the ACC algorithm model, not the desktop PC. This allows for system testing with the true communication interface as it will exist in the team vehicle, providing accurate communication delays and signal resolution. The HIL setup also ensures that the target hardware can run the ACC algorithm in real time.

Driver-In-the-Loop (DIL) Testing

DIL provides a first-person driver perspective of the ACC-enabled ego vehicle as shown in Figure 10. A setup configuration for the DIL testing is shown in Figure 11.

Figure 10. Example DIL scenario

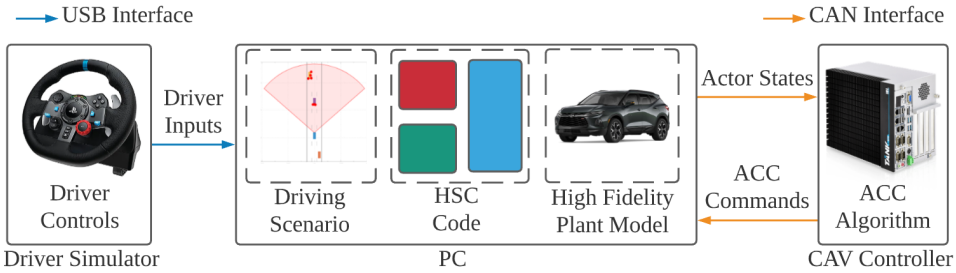

Figure 11. DIL testing configuration for ACC

Using a Logitech G29 driving simulator, the driver provides pedal, steering, and button inputs to control the ego vehicle in the simulation scenario, including engagement and disengagement of ACC. Similarly, to the HIL setup, the HSC communicates with the CAV controller over a physical CAN interface. This setup allows for user acceptance testing of the overall system as well as verification of driver overrides. Additionally, different controller parameters can be implemented in the DIL environment to compare relative driver preference. DIL is not used to quantitatively validate requirements but is instead used as a qualitative check before deployment to the vehicle.

Algorithm Testing Scenarios

The ACC system is subjected to several scenarios to test the robustness of the control algorithm in different use cases. These scenarios are described in Table 5.

Table 5. ACC testing scenarios

ID	Scenario Name	Scenario Description
1	Stationary target	Ego vehicle moving at 25 m/s approaches a stationary target from 160 m away.
2	Slower target	Ego vehicle moving at 30 m/s approaches a target moving at 15 m/s from 160 m away.
3	Target aggressive brake	Ego vehicle moves in pace 60 m behind target vehicle at 30 m/s, and target vehicle brakes to stop at 1 g (9.81 m/s^2).
4	Ego acceleration	Ego vehicle accelerates to 10 m/s from stop.
5	Target cut in	Ego vehicle moves at 30 m/s, and target moving at 25 m/s cuts into lane at 20 m ahead.
6	Target cut out	Ego vehicle set to 30 m/s moving in pace 50 m behind target moving at 25 m/s, and target cuts out of lane.
7	Target US06	Ego vehicle follows target vehicle moving performing US06 drive cycle.
8	Target EMC EC	Ego vehicle follows target vehicle performing EMC Energy Consumption (EC) drive cycle

Each scenario is realized as a unique test case, and all requirements are validated over the entirety of each test case using check blocks. All five validation tests must be passed across eight scenarios before the ACC algorithm is deemed ready to progress to the next stage of the testing sequence. This results in a 40-element validation matrix at each step of the testing sequence. In this manner, it is easy to track when failures occur and for which scenarios, simplifying controller debugging. If any one of these test elements fails, the algorithm parameters are changed, and the testing sequence is repeated from the beginning. This process is highlighted in the following section.

Algorithm Improvement

The scenarios and validation tests described in the previous section were performed in the MIL setup using initially chosen controller parameters. The resulting validation matrix is shown in Figure 12, where an "X" denotes a failed test. The algorithm validation matrix makes it clear that the initial ACC algorithm required improvement to meet the system expectations. Figure 13 shows detailed results of one of the failed tests.

	OR.50.100	OR.50.110	OR.50.150	SR.50.100	SR.50.110	
Scenario # 1						
Scenario # 2	X		X			X
Scenario # 3	X	X	X			X
Scenario # 4						
Scenario # 5			X			X
Scenario # 6			X			X
Scenario # 7		X	X			X
Scenario # 8		X	X			X
Results	X	X	X			

Figure 12. Validation matrix of initial algorithm

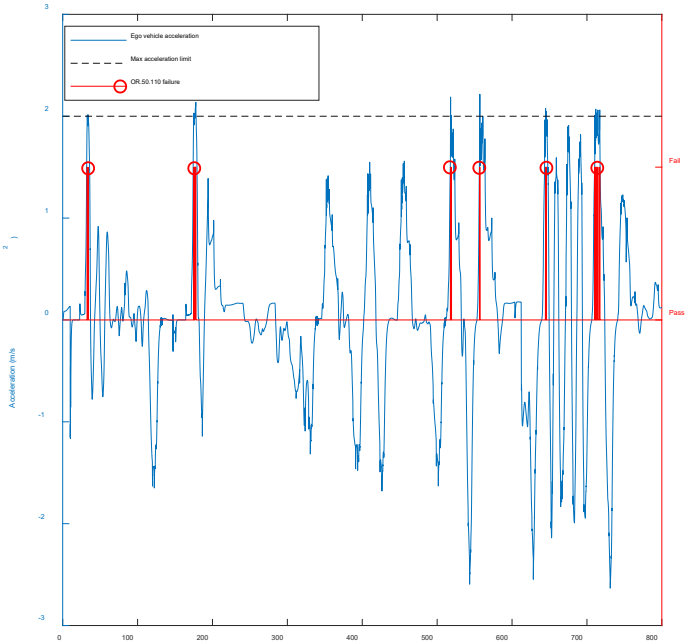

Figure 13. Initial algorithm performance in Scenario #7 (US06)

Analysis of these results reveals that the ACC algorithm is commanding too high of an acceleration from a stop, and that changes in the acceleration are unnecessarily sudden. To mitigate this, a third operating region was added to the controller for gain scheduling. This region relates to situations where the ego vehicle is operating in the

distance maintaining mode and the target vehicle is decelerating, as inspired by Takahama and Akasaka (2018). The updated controller weights with the new gain scheduling structure are shown in Table 6, where italicized entries denote a new or changed parameters. With the new controller structure and updated weights, the ACC system successfully completed all eight scenarios without a failure in both the MIL and HIL environments.

Table 6. Updated ACC controller parameters based on testing results

Metric	Initial Weight		Updated Weight		
	I	II	I	II	III
Ego Acceleration	0.2	0.3	***0.3***	0.3	***0.1***
Ego Jerk	0.01	0	0.01	***0.01***	***0.01***
Distance Error	0.02	0	0.02	0	***0.01***
Velocity Error	0.1	0.1	0.1	0.1	***0.1***

Comparison of Testing Environments

Table 7 shows the time required to perform different tests in both the MIL and HIL environments, averaged over five runs each. In all cases, the MIL environment performs the tests at least 87% faster than the HIL environment. This is partially because MIL testing has the capability to accelerate simulations using the Rapid Accelerator mode in Simulink, which greatly reduces the execution time of a single simulation. The HIL environment is unable to take advantage of this feature because it must physically relay data from the desktop PC to the CAV controller via the real-time CAN bus. The MIL environment can also utilize parallel computing, which leverages multi-core CPUs to conduct multiple simulations simultaneously. Since the HIL testing uses a single physical CAN connection between the CAV controller and the desktop PC, running parallel tests is not possible in this environment. This means that all HIL tests must be performed in sequence, dramatically increasing the time required to complete all scenarios for the validation testing.

Table 7. Comparison of execution time between MIL and HIL environments

Scenario	HIL	MIL (series)	MIL (parallel)	Time Reduction	
				Series	Parallel
Scenario #7 only (US06)	632.2 s	76.6 s	-	87.9 %	-
Scenario #8 only (EMC EC)	3735.4 s	82.2 s	-	97.8 %	-
Scenarios 1-8 (all)	4962.2 s	723.4 s	197.5 s	85.4 %	96.0 %

Because of its ability to rapidly conduct validation tests, the MIL testing is effective for performing design iterations, making it a useful tool for expedited algorithm development. The primary objective of the ACC HIL testing is to ensure that executing the algorithm on the target hardware does not significantly alter the system performance. To confirm this point, the ego vehicle acceleration and velocity envelopes were examined during Scenario #4 (ego

vehicle acceleration). These results were then compared between both the MIL and HIL environments, with the results shown in Figure 14 and Figure 15.

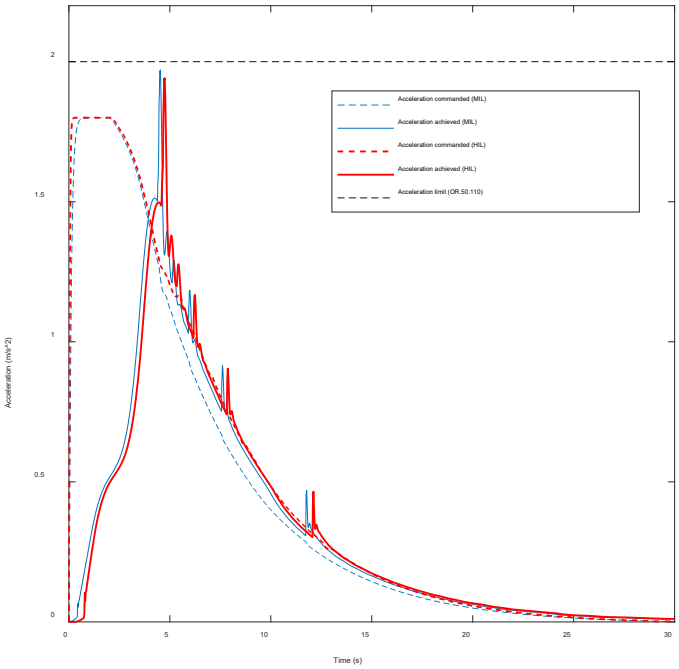

Figure 14. Comparison of acceleration envelopes between MIL and HIL environments

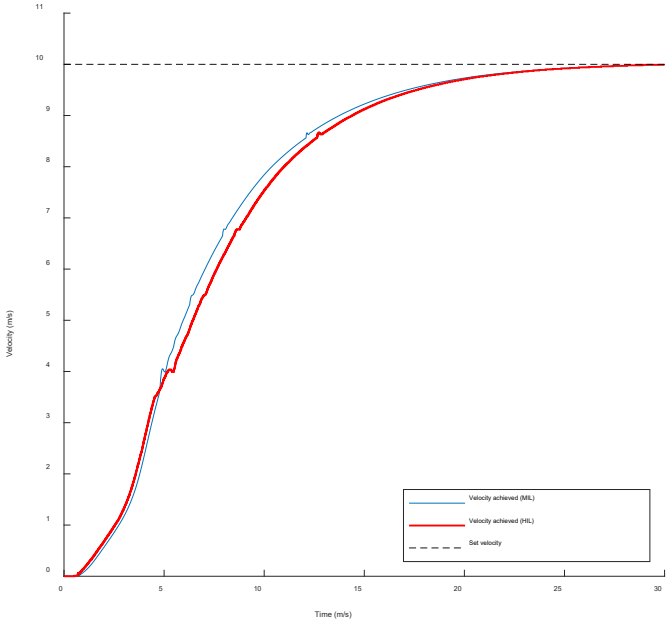

Figure 15. Comparison of velocity envelopes between MIL and HIL environments

Both the acceleration and velocity traces are very similar between the two environments, with specific metrics shown in Table 8.

Table 8. Comparison of Scenario #4 performance between MIL and HIL

Metric	MIL	HIL	Difference
Time constant of velocity response	7.46 s	8.05 s	+7.3 %
Root mean squared error (RMSE) of acceleration achieved vs. commanded	0.545	0.483	-11.4 %

As expected, the system response is slightly slower in the HIL environment. This is due to the time required for messages to be sent and received across the physical CAN interface in addition to both computers' application layer scheduling of the relevant messages. It is unexpected and notable that the ego vehicle acceleration tracks the commanded acceleration in the HIL environment better than in MIL.

Conclusion and Future Work

This paper detailed how the UA EcoCAR Mobility Challenge team is applying the model-based design methodology to verify its adaptive cruise control algorithm. The ACC algorithm was evaluated using system requirements in a testing sequence. A HIL environment was developed and used to further confirm findings from MIL and validate that the ACC algorithm met the system's minimum viable product requirements. Test automation and parallel computing greatly improved the existing MIL environment, executing tests up to 97.8% faster than real-time environments such as HIL. Comparisons between MIL and HIL results revealed differences in system dynamics, highlighting a need for future VIL testing.

The team will continue to expand and validate system requirements to further optimize the ACC system for fuel economy and driver comfort. The team's current focus is on transitioning the ACC algorithm to the team vehicle for VIL testing. As mentioned previously, to ease the transition to vehicle deployment, it is of paramount importance to maintain accurate system architecture throughout all stages of testing. Since it is not possible to perform all scenarios in the VIL environment, new scenarios will need to be created to safely validate the algorithm's performance in the vehicle. An evaluation of the system response time will be performed in VIL to obtain accurate system dynamics. This data will then be used to improve the simulated plant model and to implement a time delay compensation in the ACC algorithm to improve overall performance.

Acknowledgements

We would like to thank the sponsors and organizers of the EcoCAR Mobility Challenge program. We also would like to show our sincere gratitude to the College of Engineering at the University of Alabama for providing resources and financial support for this research.

References

Aarenstrup, R. (2015). *Managing Model-Based Design.* Natick: The MathWorks, available at https://www.mathworks.com/content/dam/mathworks/ebook/gated/MBD_Book_PDF_Version.pdf (Accessed February 19, 2022).

Abdelgawad, K., Abdelkarim, M., Hassan, B., Grafe, M. and Gräßler, I. (2014). "A modular architecture of a PC-based driving simulator for advanced driver assistance systems development", *15th International Workshop on Research and Education in Mechatronics (REM)* (pp. 1-8). IEEE.

Allen, J., Koo, W., Murugesan, D. and Zagorski, C., (2020). "*Testing Methods and Recommended Validation Strategies for Active Safety to Optimize Time and Cost Efficiency",* SAE Technical Paper, No. 2020-01-1348.

Bae, I., Moon, J. and Seo, J. (2019). "Toward a Comfortable Driving Experience for a Self-Driving Shuttle Bus," *Electronics*, Vol. 8 No. 9, p.943.

Barale, S., Vars, P., Guillet, J., Alirand, M., Lagnier, J. and Toulemont, P. (2014). "Comfort analyses of the hydractive suspension using a driving simulator", *Driving Simulation Conference (DSC).*

Barnes, D., Folden, J., Yoon, H. and Puzinauskas, P. (2020). "Scalable Simulation Environment for Adaptive Cruise Controller Development", *SAE Technical Paper*, No. 2020-01-1359.

Britt, J. and Shoults, L. (2021). "How COVID-19 Led to Improvements and Adaptations to Experiential Learning Opportunities for an Increasingly Remote Environment." In *2021 ASEE Virtual Annual Conference Content Access*, Cooperative and Experiential Education Division Technical Session 1, virtual, July 26-29.

Feng, J., Song, W., Zhan, H. and Bai, J. (2020). "Research on Testing System for an Intelligent and Connected Vehicle", *Journal of Physics: Conference Series*, Vol. 1576, No. 1, p. 012047.

General Motors, (2021). "General Motors forges ahead toward a future of Zero Crashes, Zero Emissions and Zero Congestion at CES 2021", available at https://media.gm.com/media/me/en/gm/news.detail.html/content/Pages/news/me/en/2021/gm/01-19-General-Motors-forges-ahead-toward-a-future-of-Zero-Crashes-Zero-Emissions-and-Zero-Congestion-at-CES-2021.html (Accessed February 19, 2022)

Gindrat, R. (2018). "How MaaS Public Transit Is Changing the World", available at https://www.forbes.com/sites/forbestechcouncil/2018/11/06/how-maas-public-transit-is-changing-the-world/?sh=509683497518 (Accessed February 19, 2022)

Gowda, D.V., A.C, R., Thippeswamy, M.N. and Pandurangappa, C. (2018). "Automotive braking system simulations V diagram approach", *International Journal of Engineering & Technology*, *7*(3), pp.1740-1744.

MarketsandMarkets, (2021). "Mobility as a Service Market", available at https://www.marketsandmarkets.com/Market-Reports/mobility-as-a-service-market-78519888.html (Accessed February 19, 2022)

Muhammad, U., Vyas, P., Abraham, A., Sundaram, A.R., Mehta, P.R., Dauwels, J. (2019). "An integrated simulator for testing and validation of autonomous vehicle applications with physics-based rendering sensors", *Proceeding in 26th ITS World Congress*, Singapore, 21-25 October 2019.

Pikus, M. and Was, J. (2019). The application of virtual logic models to simulate real environment for testing advanced driving-assistance systems. In *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)* (pp. 544-547). IEEE.

SAE International, (2018). "SAE International Releases Updated Visual Chart For Its 'Levels Of Driving Automation' Standard For Self-Driving Vehicles." Available at https://www.sae.org/news/press-room/2018/12/sae-international-releases-updated-visual-chart-for-its-%E2%80%9Clevels-of-driving-automation%E2%80%9D-standard-for-self-driving-vehicles (Accessed February 19, 2022).

Sultan, B. and McDonald, M. (2003). "Assessing the Safety Benefit of Automatic Collision Avoidance Systems (During Emergency Braking Situations)", available at https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/18esv-000381.pdf (Accessed February 19, 2022).

Takahama, T. and Akasaka, D. (2018). "Model Predictive Control Approach to Design Practical Adaptive Cruise Control for Traffic Jam", *International Journal of Automotive Engineering*, Vol. 9 No. 3, pp.99-104.

Wu, Z., Liu, Y., and Pan, G. (2009). "A Smart Car Control Model for Brake Comfort Based on Car Following." *IEEE Transactions on Intelligent Transportation Systems*, Vol. 10 No. 1, pp.42-46.