Vehicle-level Control Systems Framework for Use in CREATE-GV Vehicle Dynamics Simulations

MODELING, SIMULATION, PROTOTYPING & VALIDATION

- Bohumir Jelinek[1], Greg Henley[1], Angela Card[1],
- Tyler Hannis[1],Michael Gibson[1], Jody Priddy[2],
- Sara Boyle[3], Miriam Figueroa-Santos[3], Jeremy Mange[3]

[1]Center for Advanced Vehicular Systems, Mississippi State University, Starkville, MS

[2]U.S. Army Corps of Engineers (USACE) Engineer Research and Development Center (ERDC), Vicksburg, MS

[3]U.S. Army Combat Capabilities Development Command (DEVCOM) Ground Vehicle Systems Center (GVSC), Warren, MI

Outline

- Introduction
- PACE (Powertrain Analysis and Computational Environment): Basis for Framework
- PACE Framework
- Universal Controller Framework
- Example of ABS Controller Development
- Accomplishments and Future Work

Introduction

Introduction

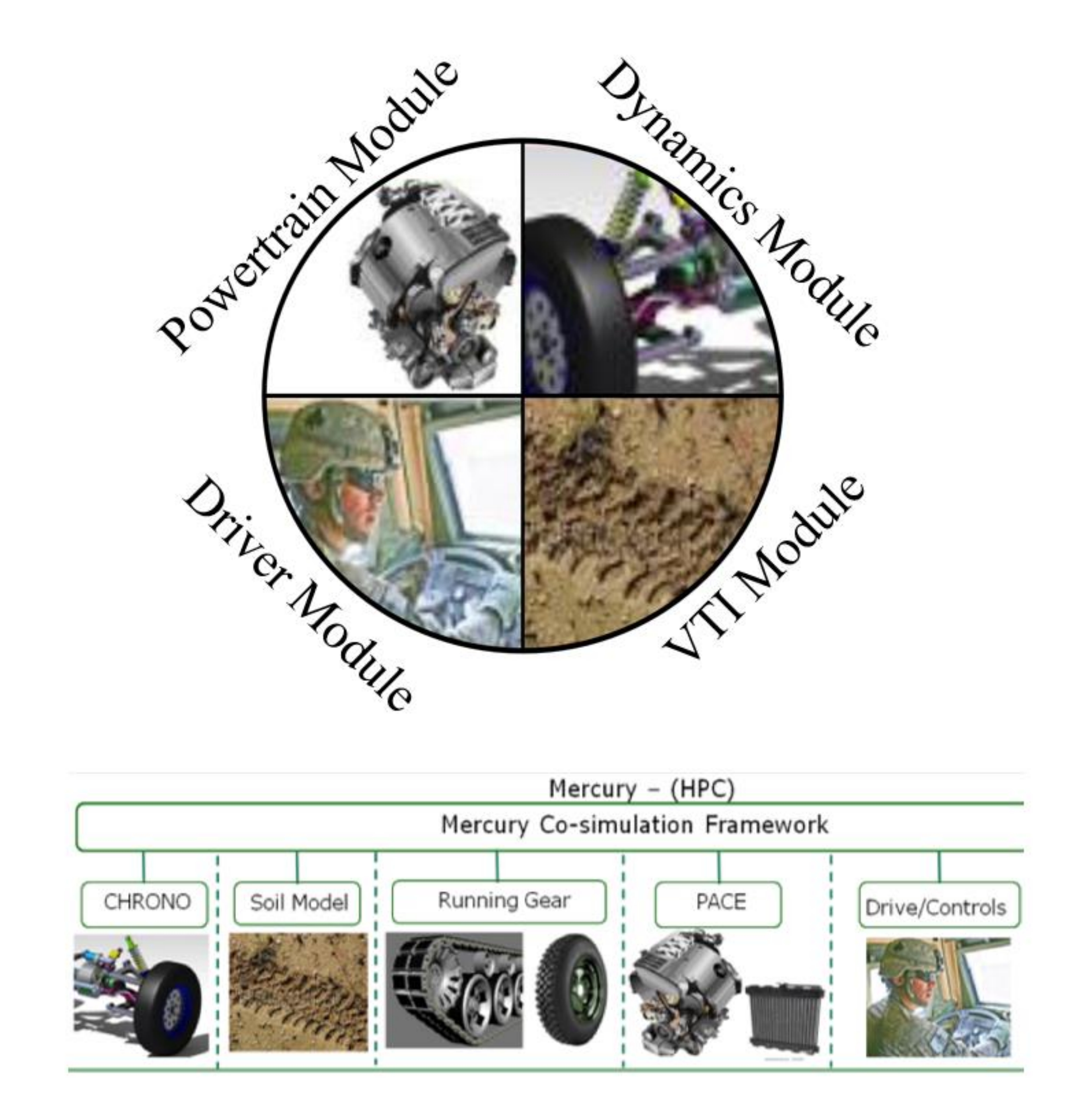
- Objective:
  - Develop an HPC ready simulation framework for vehicle level control systems
- Current practice:
  - Vehicle modeling Desktop/Windows-based platforms usually include standard vehicle control algorithms
- Our approach:
  - Simulates effects of various control systems added to a vehicle
  - Operates with existing and future HPC ground vehicle simulation platforms
  - Allows plug-ins of generic and 3rd party implementations
- Benefits:
  - Enhances the existing Set-based Design (SBD) applications aiming to perform efficient trade space analyses
  - Provides a more advanced performance evaluation and readiness of ground vehicles equipped with such controllers

PACE: Basis for Framework

PACE: A Mercury Component

CREATE-GV Mercury simulation system [1]

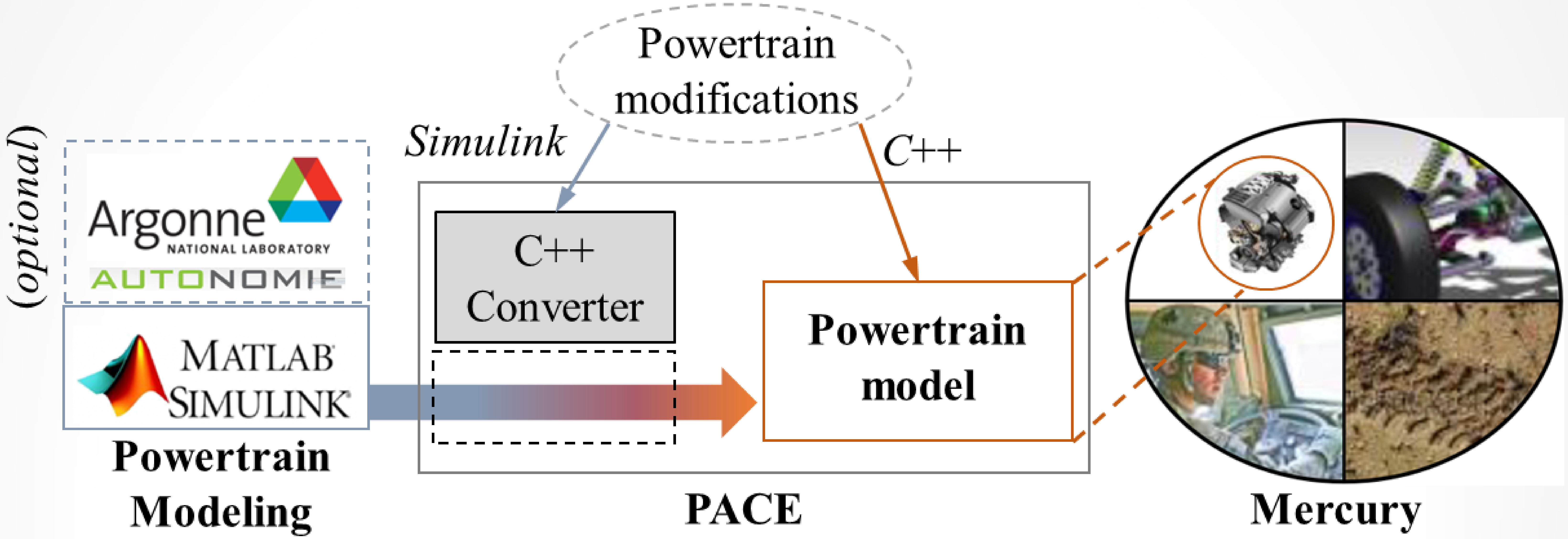
- Developed under the DoD's CREATE-GV program
- CREATE-GV is the Ground Vehicles portion of Computational Research and Engineering Acquisition Tools and Environments (HPCMP CREATE™) Program under the U.S. Department of Defense (DoD) High Performance Computing Modernization Program (HPCMP)
- Comprehensive ground-mobility simulation environment
- Rapid and flexible exploration of the ground-vehicle trade space.
- Implementation of parallel 'modules' to model all aspects of military ground vehicle mobility
- Different sets of modules required depending on the simulation type
- Modular nature permits intra-module modification
- Customizable
    - Utilizes Ground Vehicle Interface (GVI)



[1] J. Brendle, J. Woten, S. Boyle, J. Mange and T. Skorupa, "Simulation of Ground Vehicle Mobility Evaluation with Mercury," in NDIA Ground Vehicle Systems Engineering and Technology Symposium, Novi, Michigan, 2020

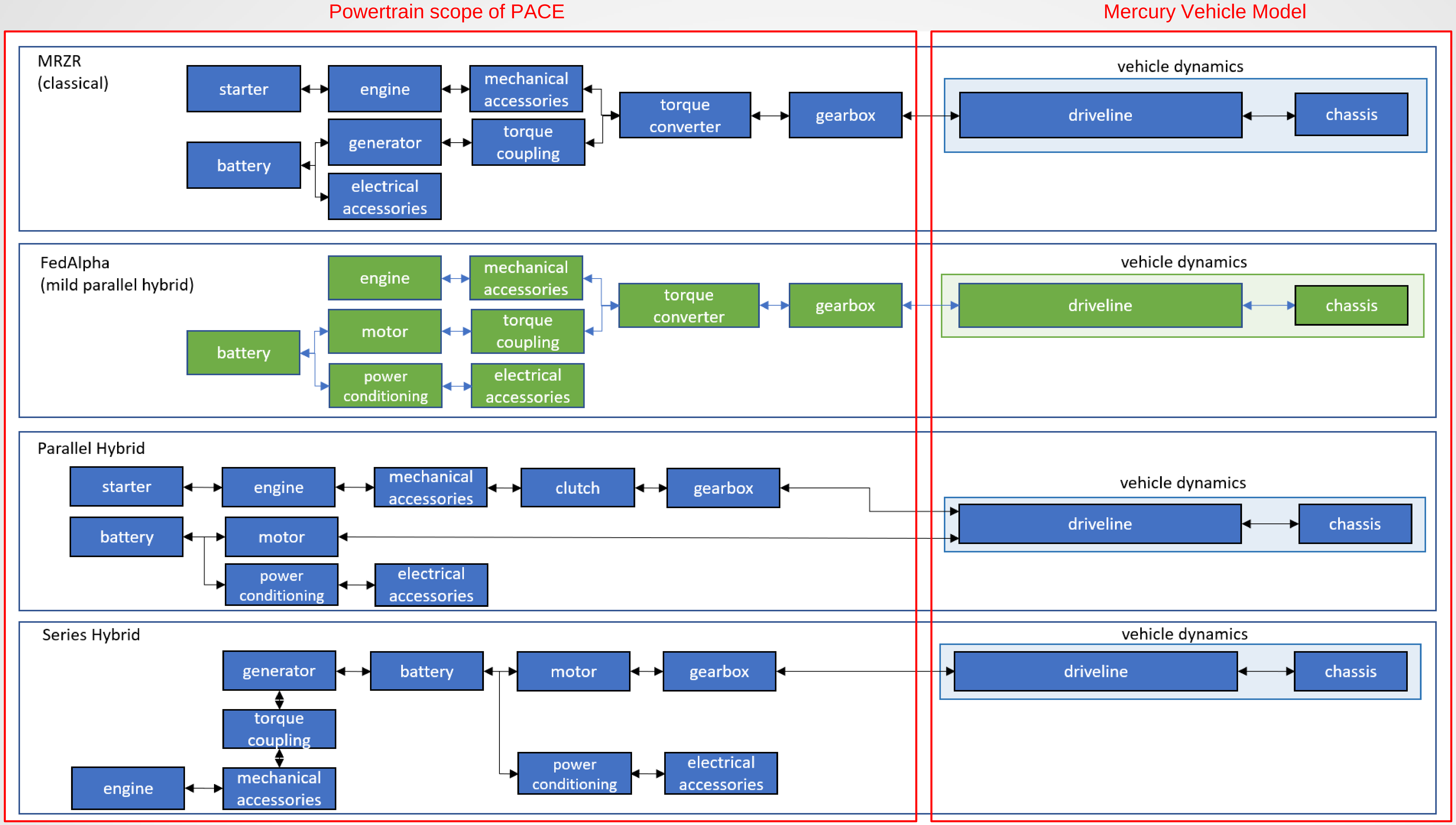
PACE

- Provides fast and high-fidelity simulations of a vehicle powertrain and control systems to be used on HPC platforms
    - developed in C++, runs on Linux clusters
    - allows thousands of PACE instances to be run concurrently in the Mercury environment
- Employs behavioral models
- Inspired by ANL's Autonomie (MATLAB-based, MS Windows only)
- Supports powertrains and *vehicle-level control systems*
    - Traction Control System (TCS), Antilock Braking System (ABS), Electronic Stability Control (ESC), ...



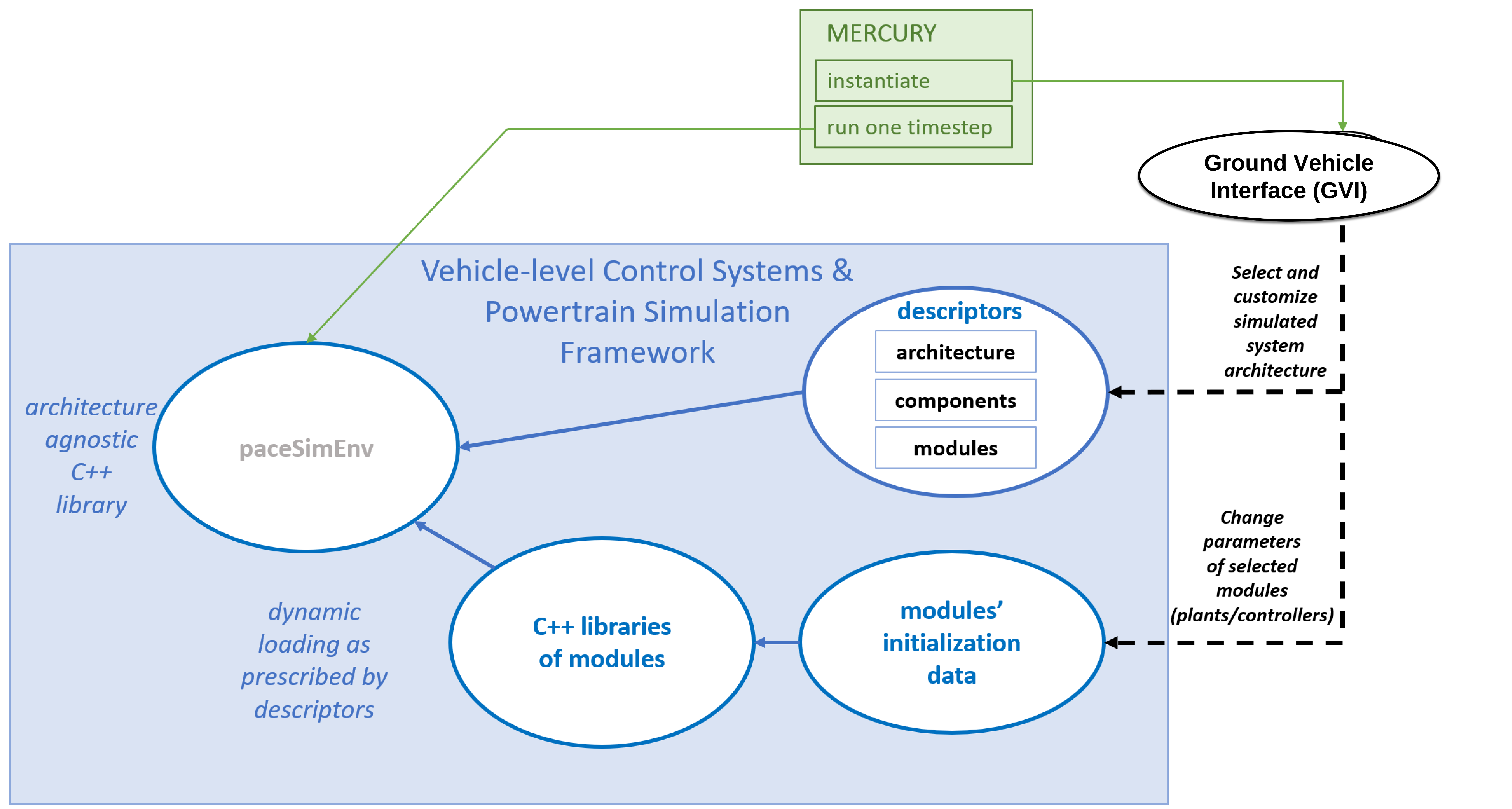
Powertrain architectures in PACE

MODELING, SIMULATION, PROTOTYPING & VALIDATION

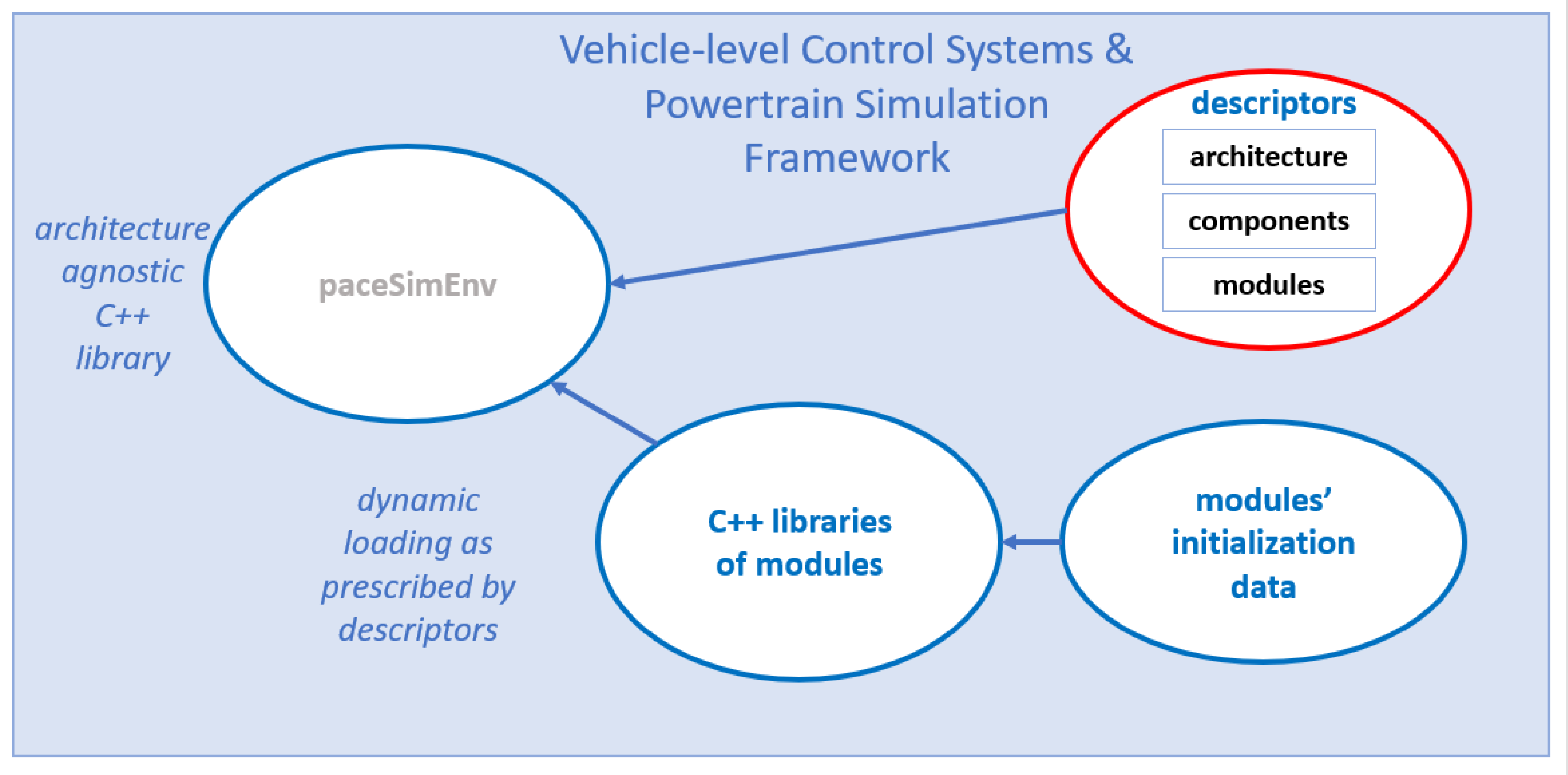


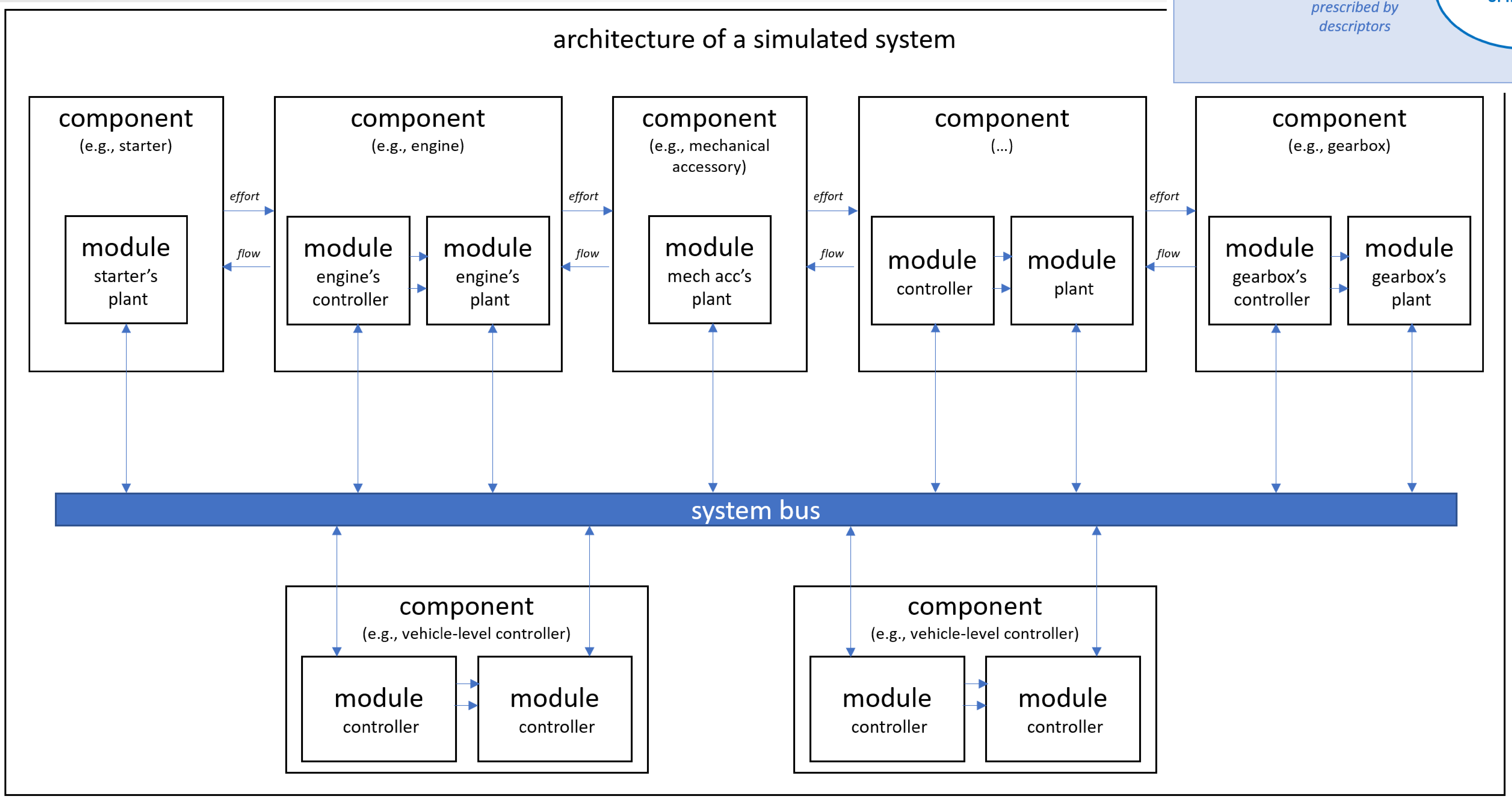
PACE Framework

Framework design



Defining a Simulated System in PACE





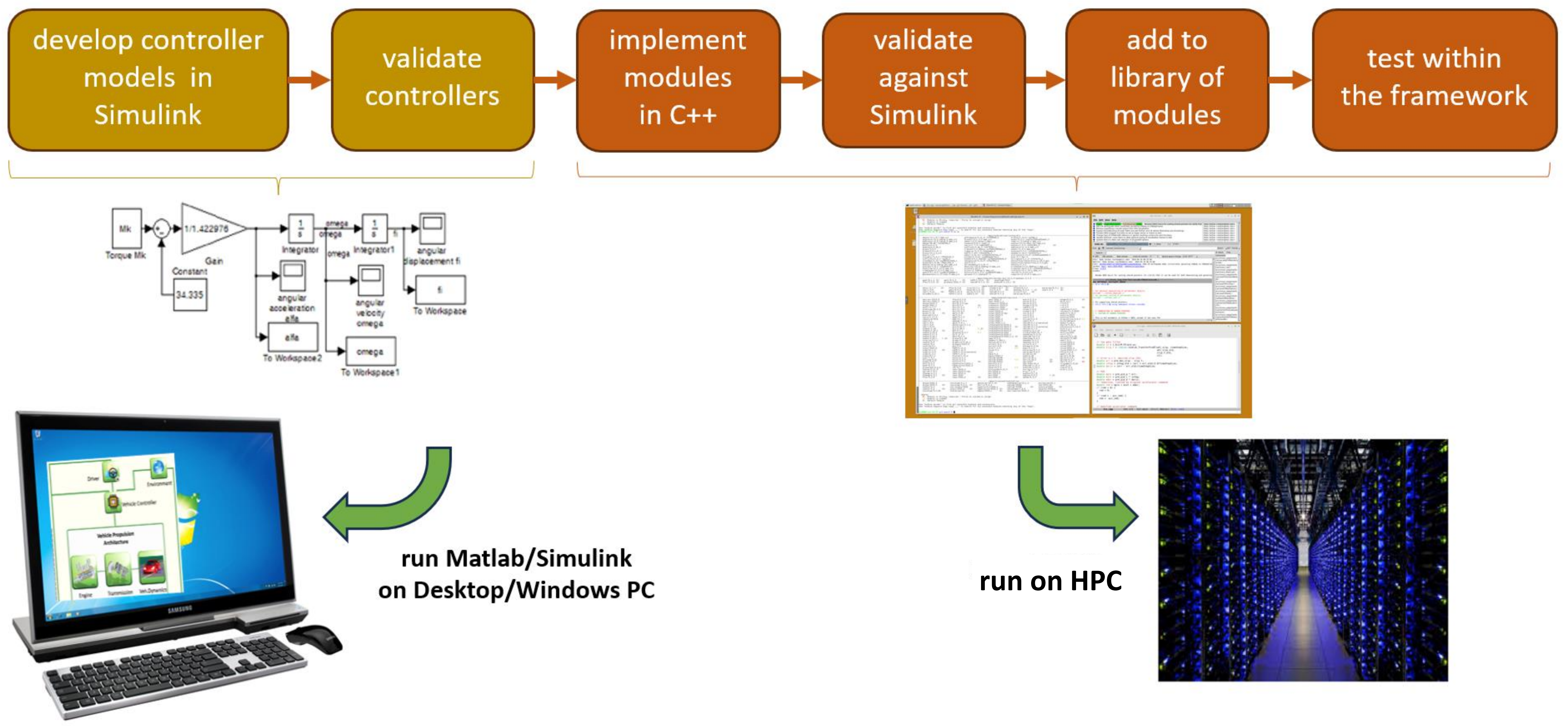
Universal Controller Framework

Universal Controller Framework Structure

- Design and implement structure for adding new control systems
  - Include C++ classes to support plug-ins
  - Implement 3rd party plug-ins using an API

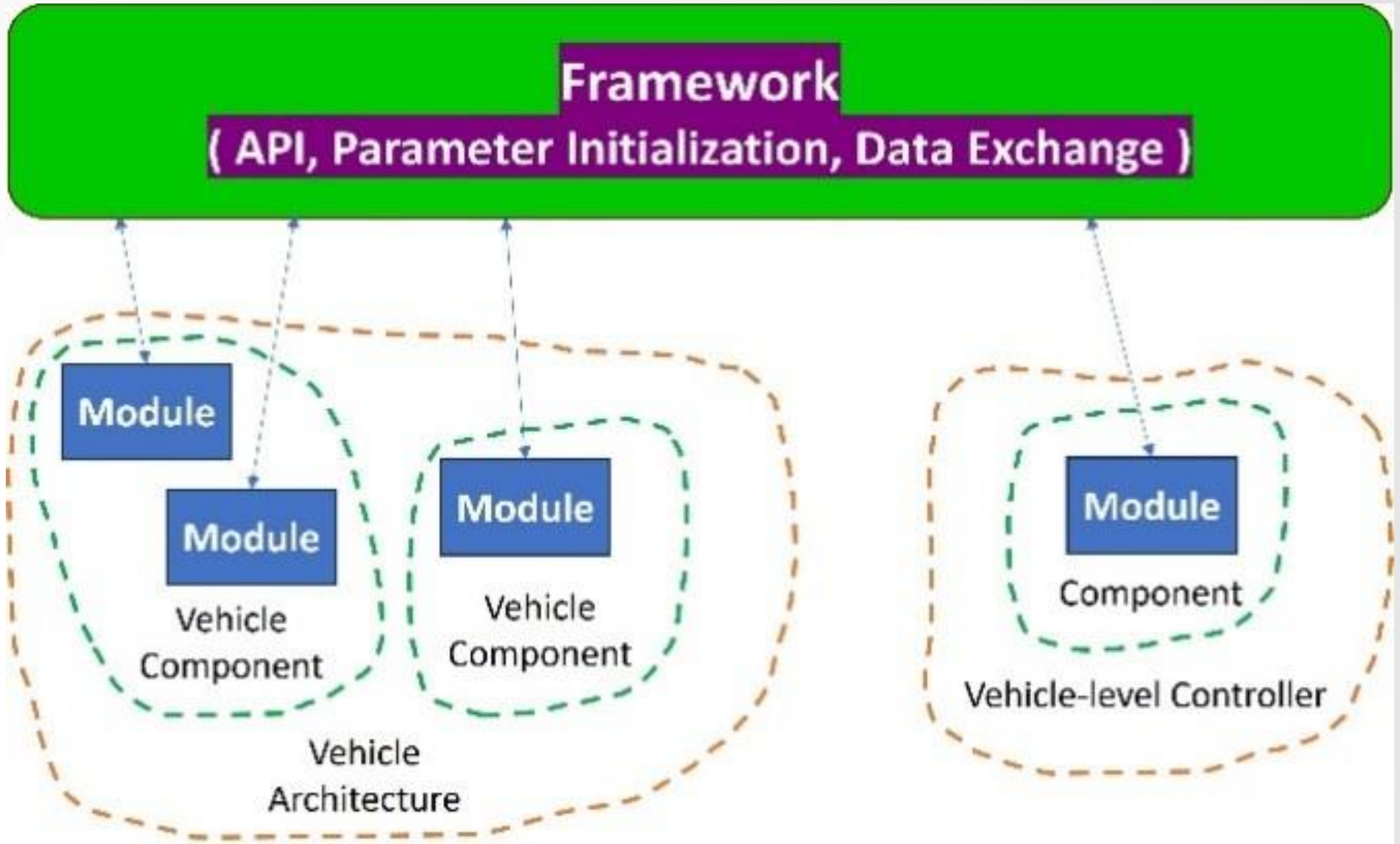
- Validation of the new software architecture
  - Develop algorithms for ABS, TCS, and Electronic Stability Control (ESC) systems

Implementation of the control system models



Universal Controller Framework Structure

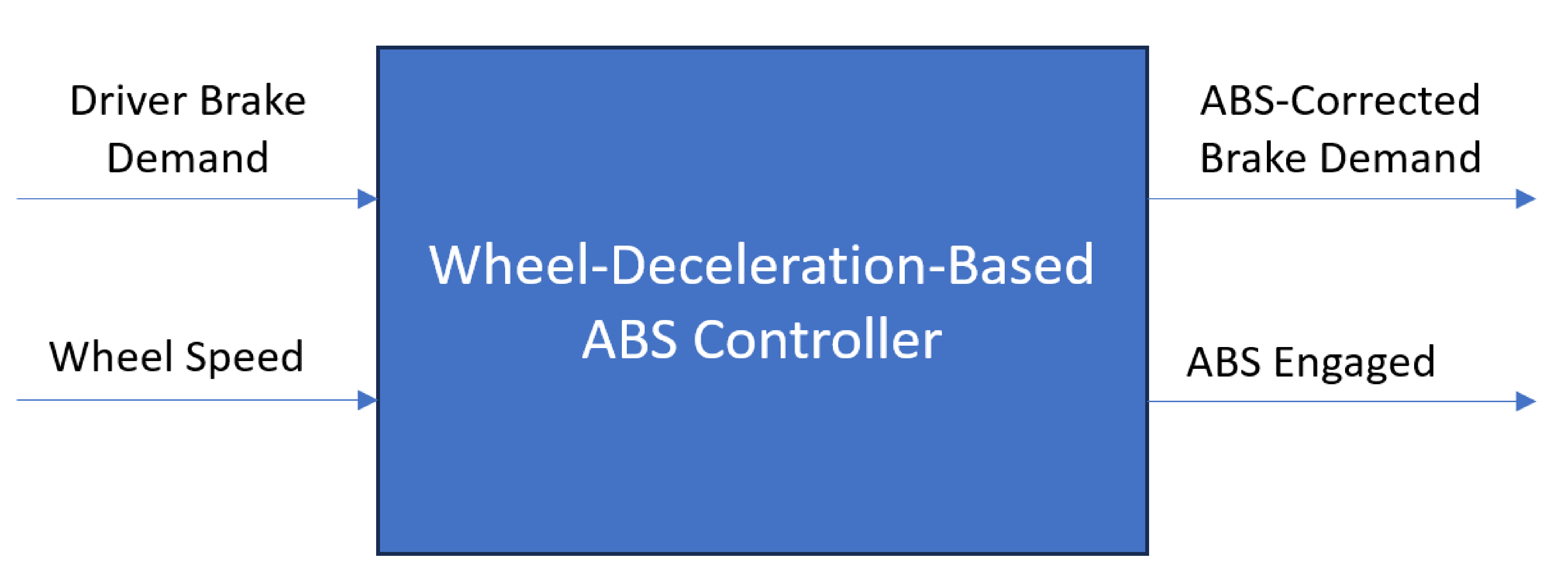
- C++ objects library of vehicle controllers
  - Hybrid Vehicle Torque Demand Controllers
  - ABS
  - TCS
- API (Application Programming Interface)
- C++ code to integrate control components within ground vehicle simulation environment

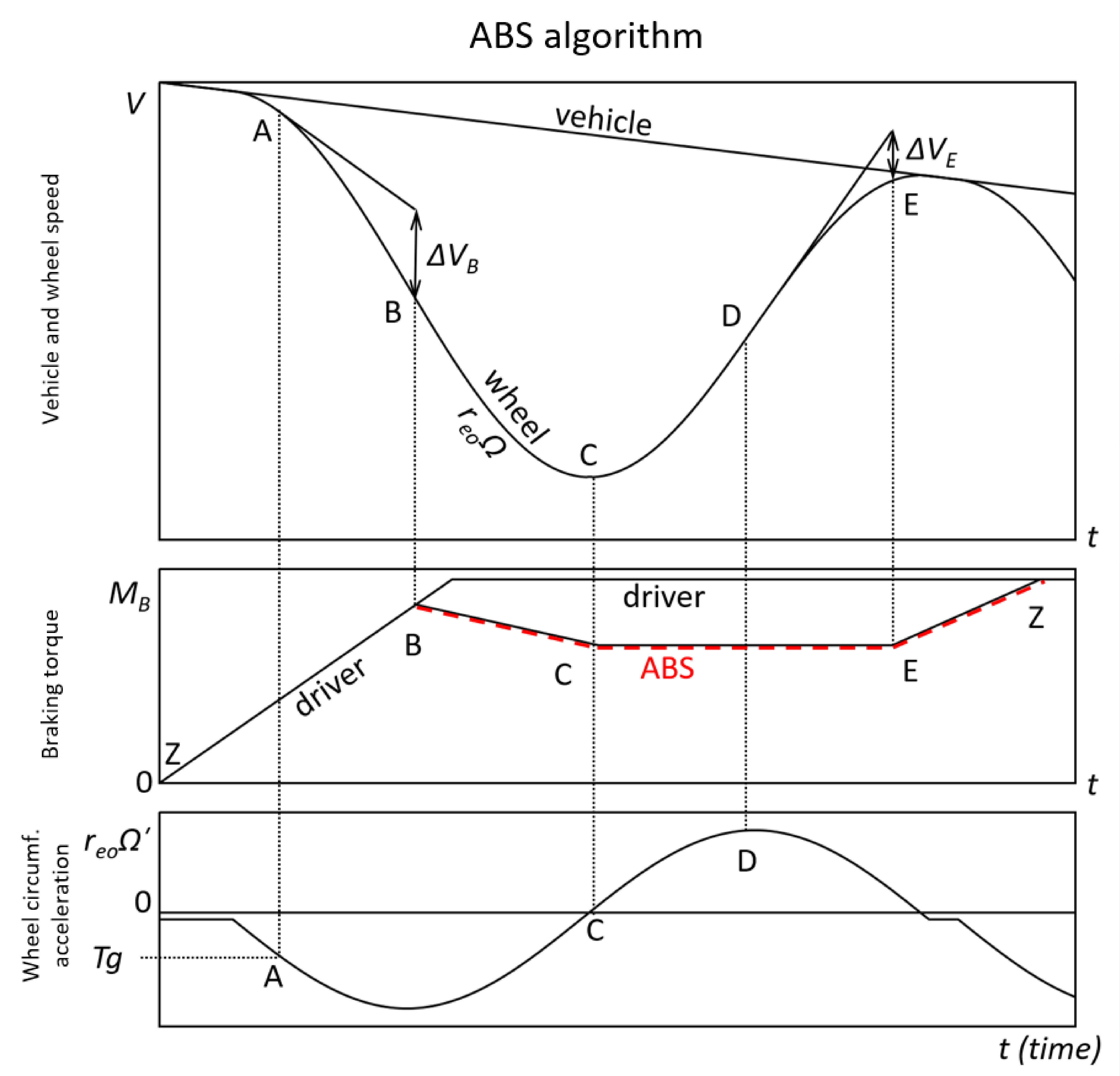


Example of ABS Controller Development

ABS Controller Model

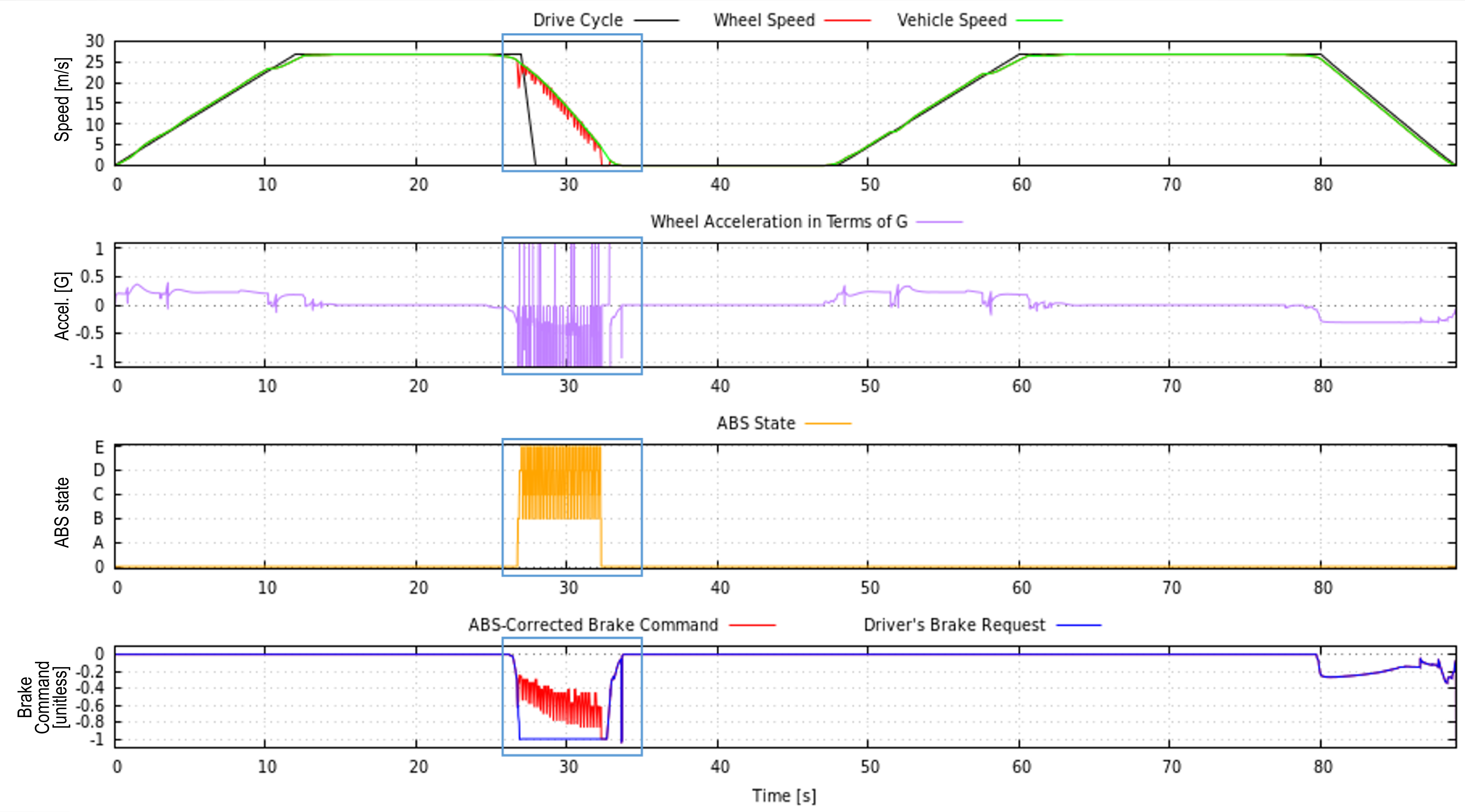
- Our ABS algorithm follows the Pacejka & Besselink model [1]
- Deceleration based controller
- Deactivates below a specified vehicle speed threshold





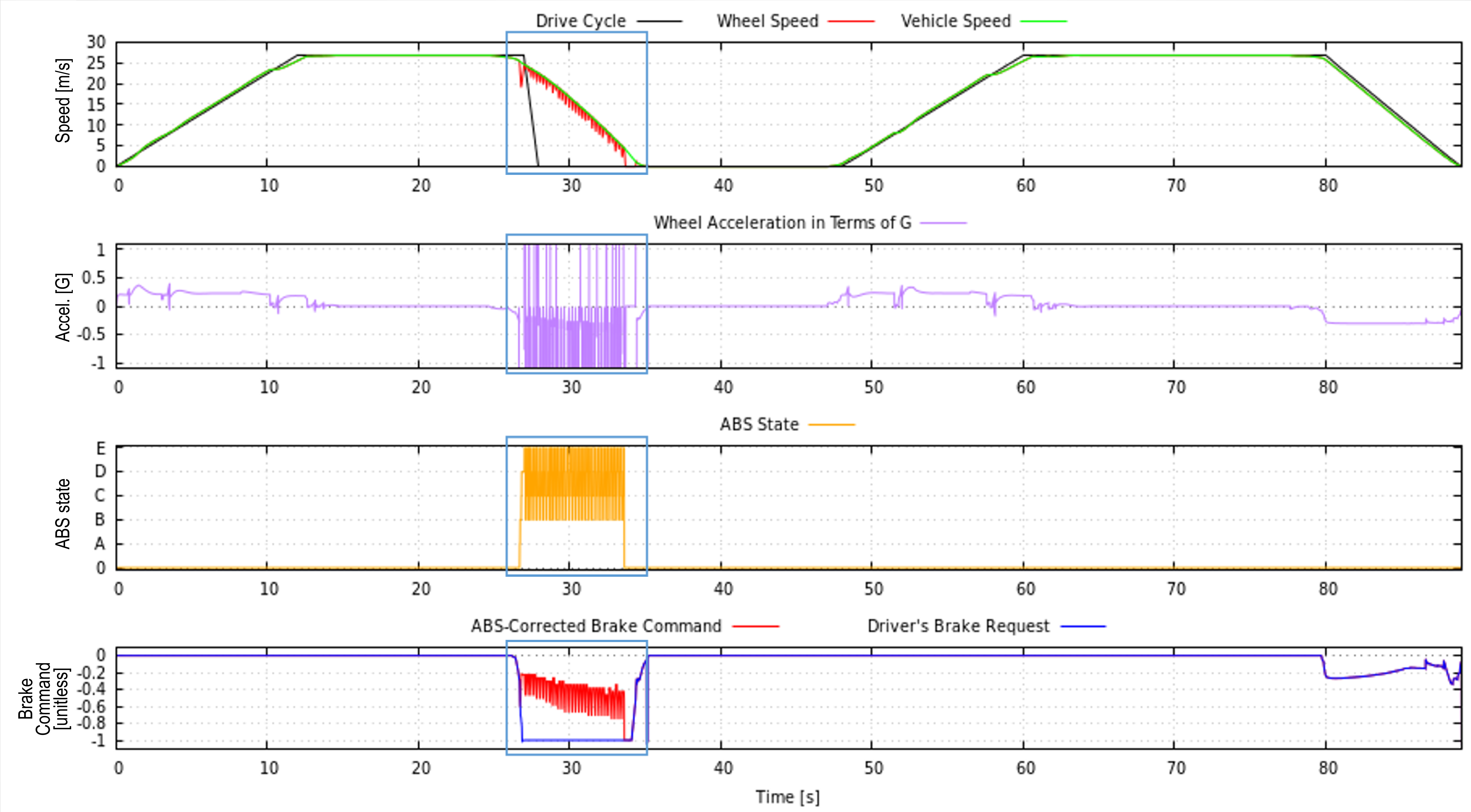
[1] Pacejka and I. Besselink, Tire and Vehicle Dynamics, 2012

ABS Model Validation - Dry Asphalt



ABS Model Validation - Wet Asphalt

MODELING, SIMULATION, PROTOTYPING & VALIDATION



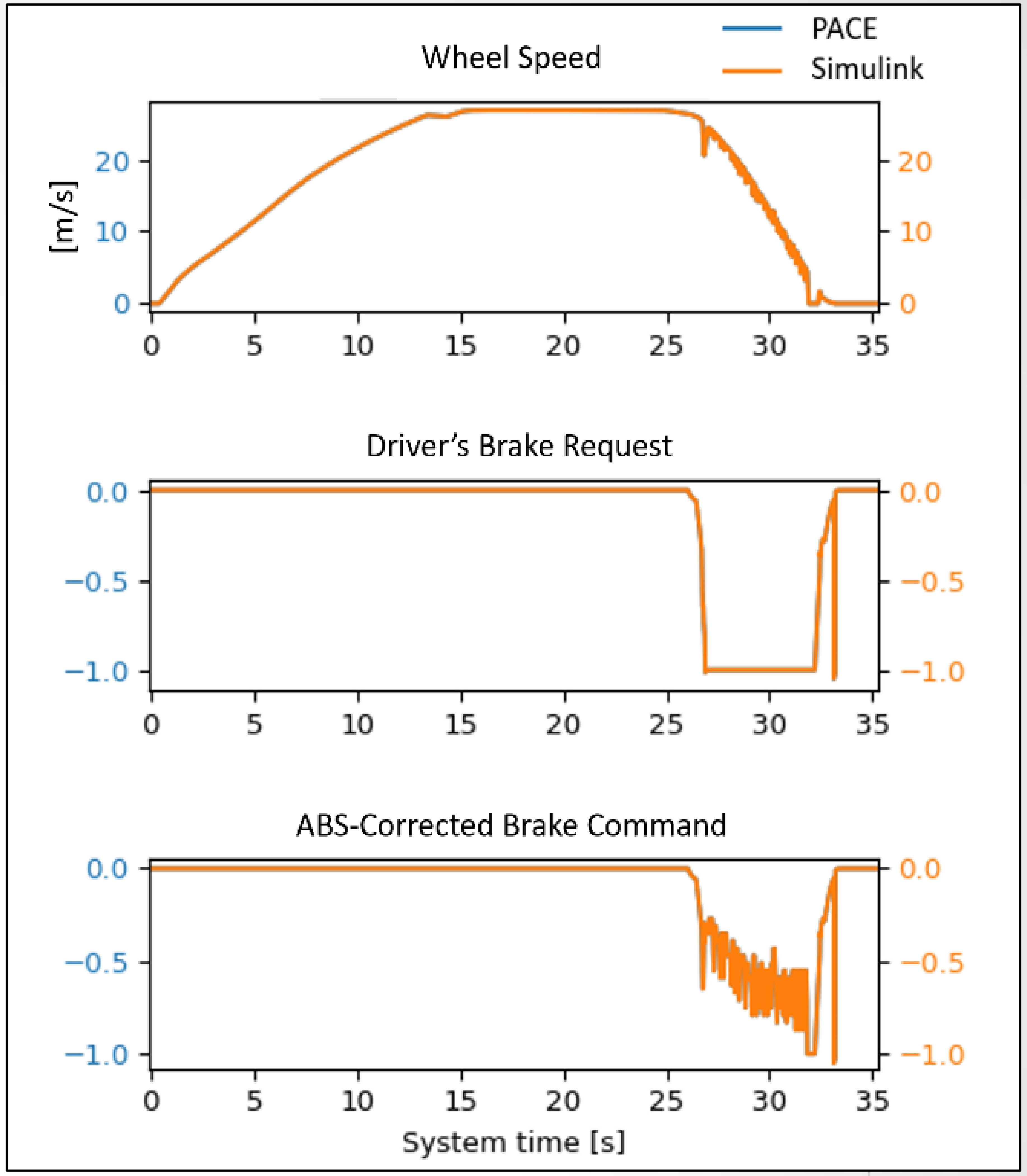
ABS Model Verification: PACE vs. Simulink

- Converted Simulink model to a C++ module
- Comparison
  - Statistical Methods for Data Comparison

$$R^2(y,\hat{y}) = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

  - Calculate average ${R_{avg}}^2$ for the module
  - ABS results in ${R_{avg}}^2$ = 0.999

Comparison of ABS signals



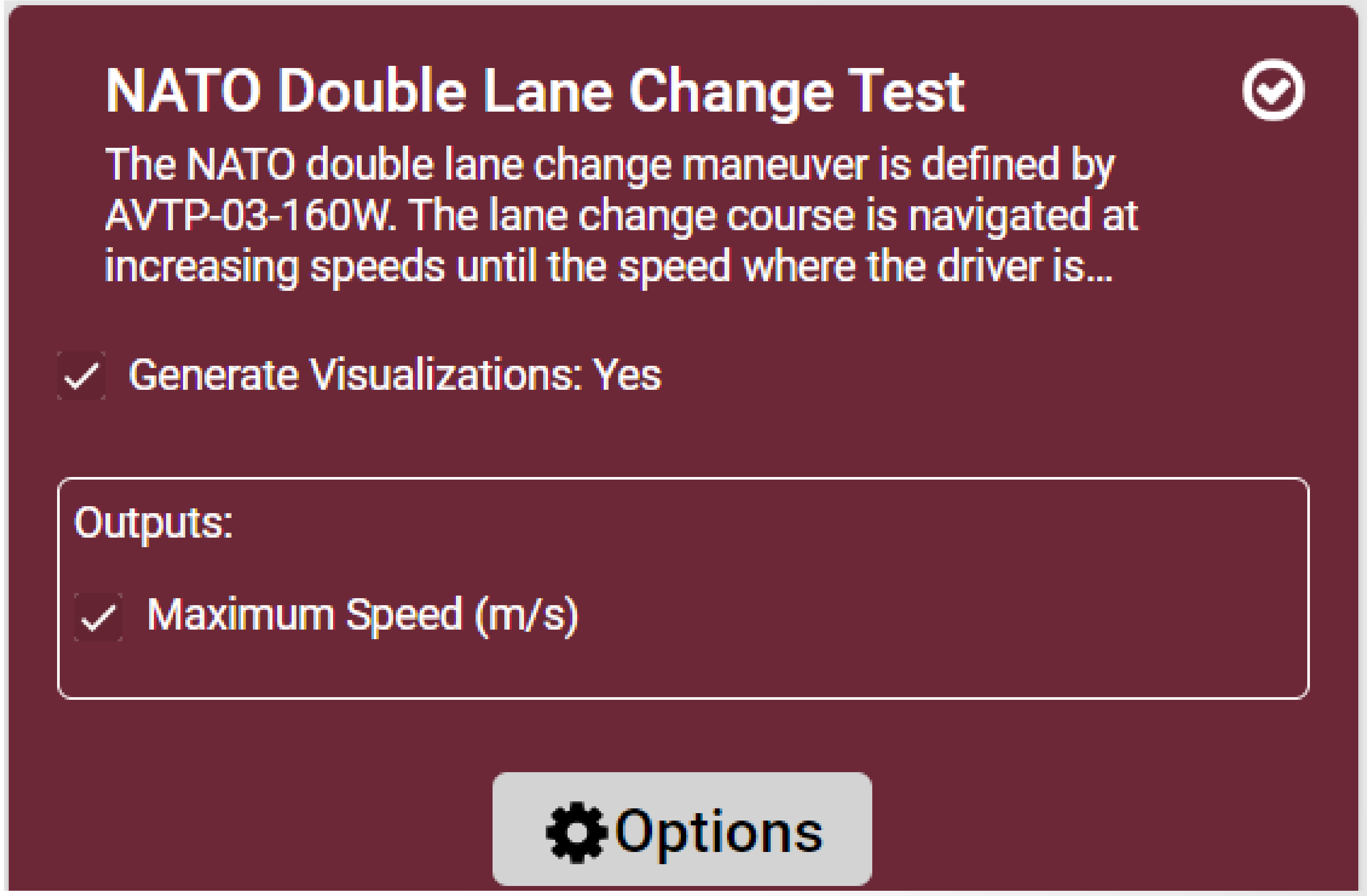
Conclusions and Future Work

Accomplishments

- Developed a modular C++ software framework for HPC simulations of vehicle-level control systems in ground vehicles
- Used the framework to examine the effect of ABS on braking under various surface conditions
- Included interactions of hybrid powertrain, regenerative braking, and ABS

Future Work

- Develop and package basic TCS and ESC module
- Enhance testing of controllers
  - Implement test scenarios for evaluation of the control systems
  - Assess impact of controllers on off-road performance



Thank you! Questions?