

Near Automatic Translation of Autonomie-Based Power Train Architectures for Multi-Physics Simulations Using High Performance Computing

Tomasz Haupt, Gregory Henley, Angela Card, Michael S. Mazzola, and Matthew Doude
 Mississippi State Univ

Scott Shurin
 US Army TARDEC

Christopher Goodin
 US Army ERDC

ABSTRACT

The Powertrain Analysis and Computational Environment (PACE) is a powertrain simulation tool that provides an advanced behavioral modeling capability for the powertrain subsystems of conventional or hybrid-electric vehicles. Due to its origins in Argonne National Lab's Autonomie, PACE benefits from the reputation of Autonomie as a validated modeling tool capable of simulating the advanced hardware and control features of modern vehicle powertrains. However, unlike Autonomie that is developed and executed in Mathwork's MATLAB/Simulink environment, PACE is developed in C++ and is targeted for High-Performance Computing (HPC) platforms. Indeed, PACE is used as one of several actors within a comprehensive ground vehicle co-simulation system (CRES-GV MERCURY): during a single MERCURY run, thousands of concurrent PACE instances interact with other high-performance, distributed MERCURY components. A proof-of-concept implementation of PACE, as applied to a conventional powertrain architecture, was presented at the SAE2016 conference. Since then, a C++ library of components implementing the functionality of the corresponding Simulink subsystems has been developed, followed by streamlining the process of the generation of the C++ code for a particular powertrain; the native Simulink XML representation of the architecture (components and their connectivity) is used for an automatic generation of the simulation workflow and thus effectively reproducing the functionality of the Autonomie models while making it compliant to MERCURY requirements. The resulting PACE models are rigorously verified and validated against results generated by Simulink runs. Furthermore, the whole process is largely automated, thus providing time savings when implementing new vehicle architectures.

CITATION: Haupt, T., Henley, G., Card, A., Mazzola, M. et al., "Near Automatic Translation of Autonomie-Based Power Train Architectures for Multi-Physics Simulations Using High Performance Computing," *SAE Int. J. Commer. Veh.* 10(2):2017, doi:10.4271/2017-01-0267.

INTRODUCTION

The need for the simulation of military vehicles is important in this time of advanced technological development. There are many ground vehicle architecture ideas that might prove useful in the conduct of military missions. However, the cost of creating these vehicles without adequate assurance that they would be successful is a challenge to be overcome with the help of new computer-aided design tools. Powertrain Analysis and Computational Environment (PACE) is part of the step toward quickly creating multi-physics ground vehicle simulations ready for High Performance Computing (HPC) platforms, such as multi-cluster computers.

MERCURY is a high performance computing simulation system that is designed to simulate military ground vehicles for mobility. There are several components that make up the MERCURY cooperative simulation framework. PACE is used as one of several actors within a

comprehensive ground vehicle co-simulation system (CRES-GV MERCURY). During a single MERCURY run, thousands of PACE instances interact with other high-performance, distributed MERCURY components; an example of which is shown in [Figure 1](#).

PACE is a powertrain simulation tool. A proof-of-concept implementation of PACE was presented at the SAE2016 conference [1]. This powertrain simulation tool uses advanced behavioral modeling for the powertrain subsystems of conventional or hybrid-electric vehicles. Since the original report, a number of significant improvements have been made in both the computer science of code generation and the breadth of powertrain modeling. A new work flow has been implemented to nearly automatically create HPC ready C++ code from starting-point Simulink-based powertrain simulations. In the past, all of the Simulink models were derived from Autonomie models. But support for independent model components was always a

goal of the PACE project and this is demonstrated by the addition of new capabilities not included in Autonomie, such as modeling of thermal components in the powertrain. Both new developments are discussed in the next section.

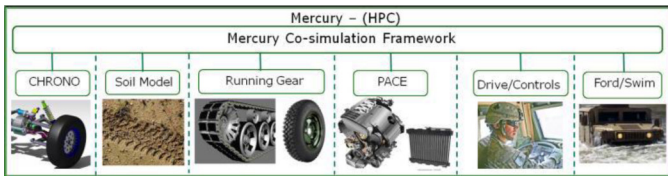


Figure 1. The PACE Federate as an actor in MERCURY [1]

SIMULINK MODELS

Autonomie is a forward-looking, robust simulation tool for the analysis of a vehicle's performance [2]. Autonomie is used to simulate realistic vehicle performance. It provides validated signal processing analysis, which allows the powertrain engineers to have confidence in the vehicle model.

PACE is a C++ implementation of a vehicle architecture designed by MSU powertrain engineers. Currently, the specifications for a military vehicle architecture are provided to MSU, whose powertrain experts then utilize a work flow that results in a Simulink representation. One method for doing this is Autonomie because it reduces the powertrain simulation to a Simulink representation [3] which is the starting point for creating PACE C++ code using the new auto-code generation workflow. However, any Simulink representation of a powertrain that uses modeling blocks supported by the PACE workflow can be used to create the operative HPC ready C++ code. In other words, PACE is a powerful tool for creating HPC ready code for the MERCURY environment using a potentially wide variety of independent third party Simulink-based simulations.

For example, if the source of a powertrain simulation is Autonomie, it generates MATLAB/Simulink models for the vehicle, which are constructed in a hierarchical format and saved as an SLX file. The top level of a vehicle model's hierarchy is shown in Figure 2. After the creation of the vehicle model, Autonomie is no longer utilized in the PACE implementation. Instead, MATLAB is used next to edit and verify the SLX file.

In Figure 2, the top level of the vehicle model is shown to have a driver component, an environment component, a vehicle propulsion controller (VPC), and a vehicle powertrain architecture (VPA). Certain aspects of the VPC and the VPA are supported in PACE. The purpose of PACE is to be an actor within MERCURY. Some of the vehicle components implemented in Fig. 2 have higher fidelity components in MERCURY. For this reason, some of the elements of the vehicle model in Fig. 2 are ignored and internal control connections implemented differently in PACE. This is automatic and transparent to the powertrain modeling engineer.

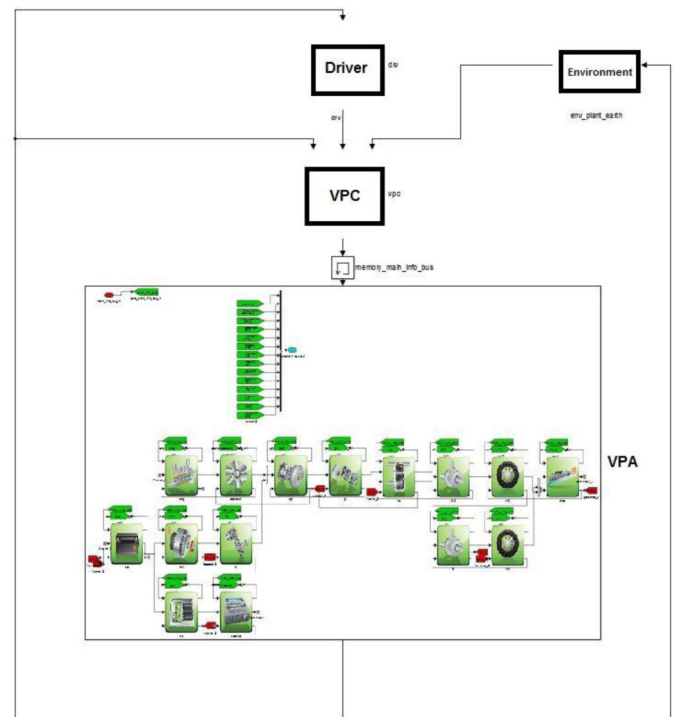


Figure 2. Image of the top level of the Autonomie vehicle model.

The components on the second level of the VPA module are the engine, the gearbox, the motor, etc. Within each of the components on the second level of the vehicle powertrain architecture are sub-hierarchies that make up the logical aspects of those components. These aspects always include a plant module, and optional controller module(s) within a VPA component's subsystem. Each module level is comprised of three blocks: an input block, a logic block, and an output block. An example is illustrated in Figure 3. The first block is the input block that connects the input to this module from the other components in the system. The middle block is the powertrain logic module that houses the functionality of the module. The last block is the output block that connects the output of this block to the rest of the system.

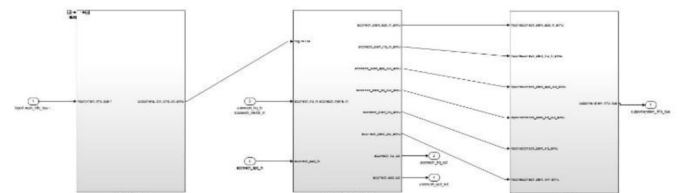


Figure 3. Image of the vehicle model VPA component's plant module

The significance of editing an SLX file in MATLAB/Simulink is that this allows the independent development of new modules. This means that third party modules can be incorporated into a vehicle model originally derived in part from an Autonomie vehicle model, which results in more functionality to the overall vehicle powertrain simulation.

For example, an independently developed thermal model is reported in a companion paper [4] that has been implemented within the PACE component. The newly incorporated thermal model was developed in