

ADAS Virtual Prototyping with the OpenMETA Toolchain

Scott Eisele
Vanderbilt University

Masahiro Yamaura, Nikos Arechiga, and Shinichi Shiraishi
Toyota Info Technology Center USA

Joseph Hite, Jason Scott, Sandeep Neema, and Theodore Bapty
Vanderbilt University

ABSTRACT

Complex systems, such as modern advanced driver assistance systems (ADAS), consist of many interacting components. The number of options promises considerable flexibility for configuring systems with many cost-performance-value tradeoffs; however the potential unique configurations are exponentially many prohibiting a build-test-fix approach. Instead, engineering analysis tools for rapid design-space navigation and analysis can be applied to find feasible options and evaluate their potential for correct system behavior and performance subject to functional requirements.

The OpenMETA toolchain is a component-based, design space creation and analysis tool for rapidly defining and analyzing systems with large variability and cross-domain requirements. The tool supports the creation of compositional, multi-domain components, based on a user-defined ontology, which captures the behavior and structure of components and the allowable interfaces. Design spaces in OpenMETA allow product families to be defined in a single model, with component/subsystem alternatives and parametric variation. Using this system design space, OpenMETA then enables analysis of the system, via composition of the system and environment/scenario models into engineering tools, and executing simulations to compute metrics.

System models can be created and executed in many abstractions based on the required accuracy, phenomena, and execution speed. This paper explores use cases from simulations with high fidelity components, to a gamified environment using Unity with a simple model of vehicle physics. This allows for user-in-the-loop analysis of controllers and components. This approach benefits ADAS by allowing for rapid prototyping across an array of candidate designs while evaluating the requirements of the vehicle at the appropriate fidelity level.

CITATION: Eisele, S., Yamaura, M., Arechiga, N., Shiraishi, S. et al., "ADAS Virtual Prototyping with the OpenMETA Toolchain," *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 9(1):2016, doi:10.4271/2016-01-0002.

INTRODUCTION

Motivation

ADAS (Advanced Driver Assistance Systems) are rapidly growing the automotive industry. For example, Toyota Safety Sense, which is a type of ADAS, will be available in most passenger cars that are released by Toyota in Japan, North America, Europe by the end of 2017 [1]. Inherently, ADAS development is a difficult problem because not only must the vehicle be considered, but also its environment. The above announcement also implies that there will be a huge system variation problem even within a single vehicle. The complexity of ADAS is increasing along with their extended features, e.g., communication with other vehicles, cooperation with a navigation system, etc. the quantity of cars containing ADAS.

The above problems imply that a highly-efficient design and evaluation methodology for ADAS development is required. Additionally, van Waterschoot and van der Voort have recognized this same need for efficient design when looking at ADAS as a human factors problem [2]. Simulation-based verification and validation can be a key component in such a methodology as shown by Gruyer et al [3]. More precisely, closed-loop simulation including vehicle dynamics and road environments is essential. However, there are no existing conventional tools that can simulate ADAS while considering large system parameter and architecture variations. OpenMETA is a tool that has been developed to represent a family of products by using a class container which encompasses them called a design space. For this reason, there is interest in applying OpenMETA to ADAS. Additionally, ADAS are present in cars which are subject to significant user input. In order to improve comprehension and

evaluation of a system we decided to integrate with the Unity 3D game engine in order to gamify the simulation and open the possibility of user in-the-loop testing and feedback.

Overview

There are several design flows through the OpenMETA tool suite. One of these is to begin design from a seed design. This provides an outline for the creation of the design space. In this case the seed design will contain a car drivetrain that includes the engine, transmission, driveshaft, differential, and an ACC (Adaptive Cruise Control) controller, while the model for the suspension, tires, chassis and environment such as road conditions are captured in Unity. A co-simulation approach was chosen to model the vehicle drivetrain at varying fidelities in Modelica and modeling the overall physics of the vehicle using lower-resolution physics engine in Unity. The drivetrain model will produce a torque that is sent to a corresponding car model in the Unity environment. The car model in unity will provide the interaction with the world. Unity will send to OpenMETA the following information: current speed of the vehicle being operated, rotational speed of the wheels, sensor data from the MMR (millimeter-wave radar) which supplies ahead-of-car vehicle detection, velocity, and distance, and inputs from the user (cruise control settings, brake, accelerator).

To provide a point of comparison, as well as an alternate fidelity representation a second seed design in which all components are represented using Dymola is also created. The co-simulation with Unity requires the model in Dymola to be simple enough that it can be simulated in real time. This seed design is appropriate for offline simulations.

After the seed design is in place scenarios, or test benches, can be defined. The seed model can then be extended to encompass a design space. Once the design space is defined some static constraints may be applied to obtain a set of valid configurations. These configurations are simulated and evaluated with respect to some additional performance criteria and compared to one another. The objective is to design a controller that is performant over the set of possible vehicle configurations and test benches.

In the remainder of the paper we will cover the background of the tools used followed by the progress up to the present. We finish with conclusions on how to bring the pieces together.

BACKGROUND

OpenMETA Toolchain

The OpenMETA Toolchain brings some key features to the design of ADAS that have been previously missing: The ability to parameterize a component from a central tool which propagates the information into external analysis tools, and the ability to define and explore a design space.

As a central location for model interfaces and structure, as well as data flow, a model in OpenMETA may have references to a model in a dynamics simulator such as Dymola, and also reference a model in

CAD software, Unity or any other external tool. In order to interface with an external tool some parametric model needs to be created in that tool, as well as an interpreter in OpenMETA to transform the model into a format the external tool can use. Once the tool has an interface with OpenMETA then any parametric change made to the model in OpenMETA will appear in the simulation of the model in the external tool. Additionally if the internals of any tool specific model are changed, as long as the interface remains, the models will function as they would if the model had been generated entirely in the external tool.

The second key feature is OpenMETA's design space capabilities. A design space in OpenMETA is the combination of parametric components, and their alternatives, along with alternate architectures for the system in question. Design space exploration is the "activity of exploring design alternatives prior to implementation" [4]. The activity involves two major sub activities, expansion and pruning. The explorable design space can be expanded by the inclusion of additional component and architectural alternatives. Pruning is the process of applying constraints which eliminates a subset of the space. If a user is creating a design space in OpenMETA there will be a significant amount of pruning that occurs simply because the user is not going to include options that are obviously impractical. After creating a space, which has the potential to be enormous, a number of techniques are used to further restrict the potential design space based on specifications for the design. The first cut removes designs that fail based on static constraints, which can be evaluated without simulation.

The space is further reduced by running simulations to compare the dynamic behavior of a design against the design specifications. The design specifications are evaluated as executable requirements called test benches. Test benches include the environment, any user-in-the-loop interfaces, and other simulation components that are not part of the system design. The results from the behavior simulation are used to generate metrics which are used to evaluate the performance of a candidate design. Depending on the design activity additional analysis may be performed in order to assess design candidates' viability. After running a set of simulations they may be re-run with more computationally involved higher fidelity models in order to verify the set of viable designs with respect to the provided specifications.

Another capability of design spaces is the Parametric Exploration Tool (PET). This tool allows the designer to explore how some set of parameters affect the behavior of the system as they are varied. This can be used in several ways. The PET may be used to find an optimal set of parameters for a component given a particular system. An example is to identify acceptable gains for a controller. It can also be used to determine the input necessary to achieve the desired behavior (e.g. sizing experiments - given a vehicle, what size does the engine need to be in order to achieve 100 km/hr?). Additionally, since systems involve many trade-offs, by sampling a space and mapping the behavior to surrogate equations it is possible to perform static comparison of estimated system behavior within a specified parameter range. The range of parameters may be sampled using several techniques made available through OpenMDAO which has been integrated into the OpenMETA Toolchain. Figure 1 shows a high level view of the features of OpenMETA.

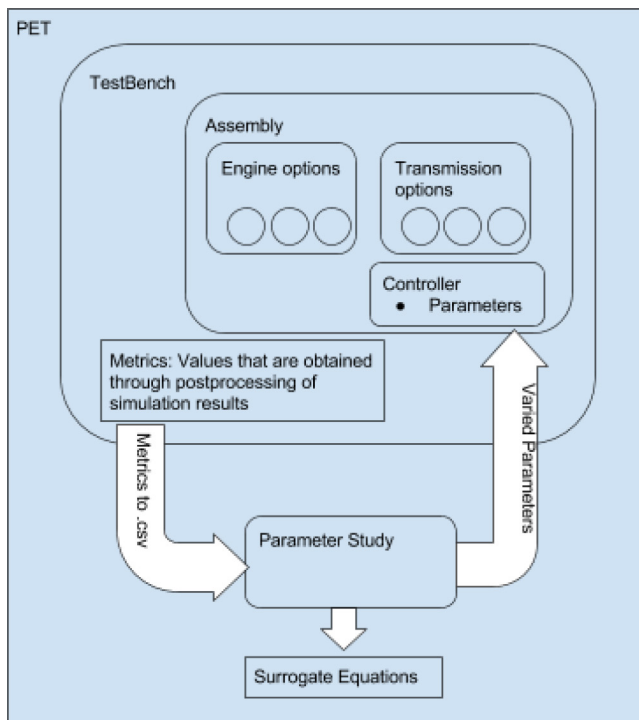


Figure 1. Overview of OpenMETA process

Modeling Tools

Simulink [10] is an object-oriented, multi-domain, graphical modeling tool produced by MathWorks. It is capable of modeling mechanical, electrical and thermal domains of physics simultaneously. Such a multi-domain modeling language is necessary to simulate ADAS vehicle dynamics. Simulink is used extensively by the automotive industry in a number of applications from controller design to optimizing user ride quality. Some of the ADAS controllers were natively developed and maintained in Simulink before being imported into the OpenMETA tools.

Modelica is an object-oriented, multi-domain, acausal modeling language. Similar to Simulink, Modelica can handle multiple domains simultaneously, but it is able to do so without the making assumptions about signal flow a priori (i.e. Modelica is acausal). Dymola [8], developed by Dassault Systems, is used as an editor and simulator of Modelica models and libraries such as the Vehicle Dynamics Library [11] developed by Modelon. Vehicle components, such as engines and transmissions, are available in Vehicle Dynamics Library. It is comparatively easier to develop whole vehicle models using this library.

Unity [9] is a 3D game engine for video game development with a GUI interface. Its features include 3D modeling, 3D physics engine, animation, and scripting in C# and JavaScript. These features can be used for modeling road and other vehicles necessary for closed-loop ADAS simulation. Although other 3D game engines are also available, Unity was selected because of its community and the additional assets available through the Unity Asset Store which helps accelerate development. For example, assets such as road editor, vehicle physics, and car traffic simulator are available.

ADAS

ADAS are systems whose purpose is to enhance driver and vehicle safety by augmenting the information available to the driver as well as taking action when necessary. As we see in Figure 2 the majority of the accidents are due to inability to recognize dangerous situations. There are systems to reduce or eliminate each class of driver error, some of which mitigate driver error of each type.

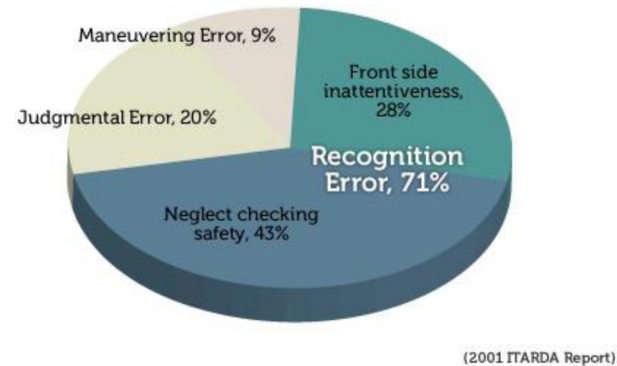


Figure 2. Causes of accidents [5]

One of these systems is the Adaptive Cruise Control. The purpose of this system is to aid the driver particularly when driving for an extended time. The ACC allows the driver to set a cruise control speed as nominal, but rather than solely maintaining that speed, the vehicle also monitors for vehicles ahead. Upon detecting a vehicle, the ACC adjusts the speed in order to maintain a safe following distance. The control flow is depicted in Figure 3. Drivers can choose the distance to be set from short, mid-range and long. Actual distances of these three settings depend on vehicle velocities. In general, sensors, such as a radars or camera, are used to monitor distance between two vehicles. These sensors also serve to provide pre-collision safety (PCS). By monitoring the distance to the preceding vehicle and the speeds of the vehicles the system can provide an alert to the driver to take braking action to prevent an accident. If the driver does not take action the system will engage the brakes to mitigate or prevent the accident.

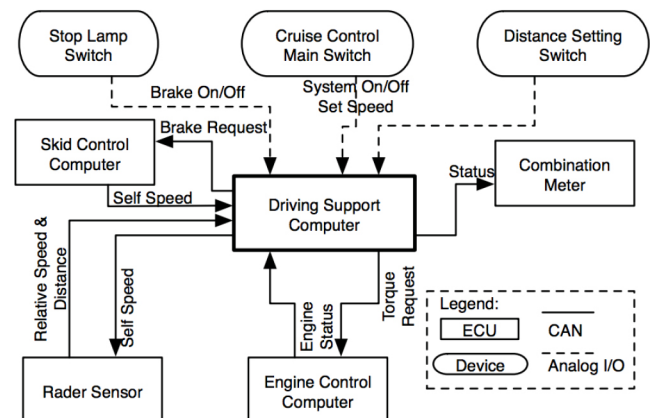


Figure 3. Adaptive cruise control diagram [6]