Using Modeling and Simulation to Support Future Medium and Heavy Duty Regulations
Aymeric Rousseau, Phil Sharer, Shane Halbach
Argonne National Laboratory, 9700 S. Cass Ave, Argonne, IL, 60439-4815 USA
E-mail arouseau@anl.gov, psharer@anl.gov, shalbach@anl.gov

Abstract— Other than in Japan, medium and heavy duty vehicles (MHDVs) are not regulated despite accounting for a significant portion of the fuel consumed (about 26% in the US in 2008). Government agencies worldwide are currently evaluating options to address that issue. Due to the large number of vehicle applications, some of them being “one of a kind”, vehicle modeling and simulation offers an attractive solution to medium and heavy duty regulations. This paper discusses the advantages and challenges of vehicle simulation to support regulations.

Keywords— Heavy Duty, Regulation, Standards, Vehicle Modeling, Simulation, Fuel Economy

1. Introduction

Liquid fuel consumption by medium and heavy-duty vehicles represents 26 percent of all U.S. liquid transportation fuels and has increased more rapidly than consumption by other sectors. To maximize fuel displacement, it appears obvious that specific attention has to be placed on these applications. As a consequence, the Energy Independence and Security Act of 2007 (Public Law 110-140—Dec. 19, 2007), Section 108, was passed, requiring the U.S. Department of Transportation (DOT), for the first time in history, to establish fuel economy standards for MHDVs.

To reach this goal, the National Research Council (NRC) appointed a Committee to Assess Fuel Economy Technologies for Medium- and Heavy-Duty Vehicles [1]. The committee provided a series of recommendations and guidance to properly implement future medium and heavy duty regulations in the U.S.

Due to the large number of powertrain configurations and the lack of dynamometer testing facilities, it is impossible to test every single variation of vehicle. As a consequence, the NRC committee highlighted the need for modeling and simulation to support future regulations in the U.S.

This paper discusses some of the fundamental ideas related to vehicle modeling and simulation requirements for labeling purposes and how they have been implemented in Autonomie, Argonne latest vehicle modeling tool.

2. Modeling Tool Requirements

2.1. Flexibility

Regulations are necessary to improve fuel displacement of medium and heavy duty vehicles. Due to the large number of international companies, it is critical to use a single tool to minimize the effort and cost associated with labeling, especially since processes can be implemented in many different ways depending on the country or continent, at least at the beginning. It is expected that such regulations will evolve over time due to improvements in modeling expertise as well as introduction of new technologies. As a consequence, since one would like to use the same tool over time, such a tool should be flexible.

Flexibility can take the shape of many different features. First, the tool should support any step of Model Based Control (MBC). MBC is a math-based visual method for designing complex control systems, and is being used successfully in many motion-control, industrial, aerospace, and automotive applications [2]. Model-Based Control integrates the development phases — modeling a plant (from first principles or system identification), synthesizing and analyzing a controller for the plant, simulating the plant and controller together, and programming/deploying the controller — providing efficiency and a common framework for communication throughout the process.

The phases of Model-Based Control are shown in Figure 1.

![Figure 1: V Diagram for Software Development](image)

Different steps in the development process are supported by a variety of approaches, from model-in-the-loop (MIL), to software-in-the-loop (SIL), hardware-in-the-loop (HIL), rapid-control prototyping (RCP), or component-in-the-loop (CIL). One can imagine that any of these steps be used for regulatory purposes. It is interesting to notice that
Japan is already using different processes for different applications [3]:
- Simulation for conventional vehicles
- HIL for Hybrid Electric Vehicles

One could also imagine using an engine connected with a low inertia dynamometer that would emulate the rest of the vehicle. The approach, known as Component-in-the-loop (CIL), would have the advantage of removing the modeling uncertainties of the most difficult component to model: the engine.

Second, the tool should be able to model any type of powertrain configurations, from conventional to hybrid electric vehicles (HEVs), Plug-in HEVs or electric vehicles (EVs). Even if the number of electric drive vehicle is currently small, their number, and diversity, is expected to significantly increase in the upcoming years.

Finally, the tool should be able to evolve over time to allow users to modify only the parameters allowed for labeling purposes. For example, while an initial rule might only allow the modification of few selected parameters, whereas later regulations could also allow users to modify entire initialization files or even component models and control strategies.

2.2. Reusability

With the consistently increasing number of possible powertrain configurations for medium- and heavy-duty applications, the need to quickly simulate any application is crucial. A vehicle modeling software should be able to provide the following features:
- Allow any configuration (assembly of systems) to be quickly modified and built automatically. For maintenance purposes, saving hundreds of options (a number that can easily be achieved through combination of configurations and model complexity) is not feasible.
- Allow users to quickly add their own configurations, especially for electric drive vehicles.
- Allow reusability of component models throughout all the possible configurations.

2.3. Selectable Model Complexity

While the initial regulations will be most likely be based on a single frozen model for each particular component, it is expected in the future that particular technologies, especially for the engine, will require specific models to properly represent their fuel consumption potential. In addition, if a company has developed and validated a higher fidelity model that the one currently used, they might be able, under the condition that they provide sufficient validation information, to use that model for regulations. As such, the tool selected for current and future regulations should be able to handle any complexity. To reach such a goal, the tool should have the following requirements:
- Common nomenclature, including naming convention, units (Standard International). If nomenclature is not consistent, an automated process should be provided to users to easily integrate any legacy code into the agreed upon format.
- Common model organization to facilitate interactions of different expert models.

Since many tools, written in many different languages are used throughout the industry by experts (i.e., engines might be modeled in GT-Power while transmissions are modeled in AMESim), it is crucial for the vehicle tool to be developed from the ground up as a plug-and-play platform to allows the user to integrate any legacy code from any software package as well as run all models in the same environment or through co-simulation.

2.4. Graphical User Interface

Since many of the users of the tool for labeling might not be familiar with advanced modeling techniques, it is crucial to provide a simple by efficient Graphical User Interface (GUI) to perform the simulations.

2.4.1. Setup Simulation

The graphical user interface (GUI) should be able to allow users to quickly set up different simulations, including:
- Select applications (i.e., line haul, bus…)
- Select architecture and data (entire files or only few parameters)

2.4.2. Generic Processes

When evaluating specific technologies, having consistent processes (i.e., drive cycles) is critical for proper comparison. Differences in the definitions of processes could lead to discrepancies in results, which could become a significant issue for regulatory purposes. Due to the variety of applications, different processes will have to be defined to properly represent each truck usage. As such the tool should be able to handle any type of driving profile (whether time or distance based) or post-processing.

In addition, since the applications will evolve over time, the processes implemented should be easily modifiable.

2.4.3. Results Visualization

The GUI should allow users to quickly analyze the simulation to verify the validity of the simulation. Such a task requires several features:
- Post-processing algorithms should provide energies, efficiencies… in addition to efforts (e.g., torque, voltage) and flows (e.g., rotational speed, current)
- Predefined plots should be available to quickly analyze the operating conditions of each component or control strategies.
As models and data evolve with time due to improved data producer and consumer teams spread across the country or even the world for some global companies—for example, companies, the tool should be able to be shared and accessible throughout the entire company so that any engineer can load the latest initialization parameters or results. This requirement is also necessary if companies are allowed to use their own component models to better represent specific technologies. In that case, another requirement for the sharing and distribution of proprietary models is their enterprise-wide accessibility, including for producer and consumer teams spread across the country or even the world for some global companies—for example, a control design team can have members in the United States and England, or a model calibration and validation team might be located hundreds of miles from the model design team. Up-to-date models should be accessible to all people who have the right access, wherever they are located.

2.7. Version Control
As models and data evolve with time due to improved data and/or algorithms, or even issues such as new modeling software version compatibility, the need for version control is mandatory for auditing and regulatory purposes. Any study done with any models needs to specify which version was used to ensure results traceability.

3. Implementation Example - Autonomie
Several tools already exist to develop detailed plant model, including GT-Power [4], AMESim [5], TruckSim [6], and SimScape [7]. The objective of Autonomie is not to provide a language to develop detailed models; rather, Autonomie [8] supports the assembly and use of models from design to simulation to analysis with complete plug-and-play capabilities. Autonomie provides a plug-and-play architecture to support this ideal use of modeling and simulation for math-based automotive control system design. Models in the standard format create building blocks, which are assembled at runtime into a simulation model of a vehicle, system, subsystem, or component to simulate. All parts of the graphical user interface (GUI) are designed to be flexible to support architectures, systems, components, and processes not yet envisioned. This allows the software to be molded to individual uses, so it can grow as requirements and technical knowledge expands. This flexibility also allows for implementation of legacy code, including models, controller code, processes, drive cycles, and post-processing equations. A library of useful and tested models and processes is included as part of the software package to support a full range of simulation and analysis tasks, immediately. Autonomie also includes a configuration and database management front end to facilitate the storage, versioning, and maintenance of all required files, such as the models themselves, the model’s supporting files, test data, and reports. While the tool, developed in collaboration with General Motors, was not developed for labeling purposes, its flexibility makes such application possible without much modification.

3.1. Automated Model Building
The model files created for each component need to be combined in a way that allows a full vehicle simulation. One option is to create every possible combination of the systems and save each complete vehicle as a separate model file. This option quickly becomes infeasible when one considers the staggering number of combinations for medium and heavy duty vehicle applications. Not only are we dealing with many different components, but we also must also consider that we might have different levels of fidelity and model versions for each component. Changing the version of a single component model would result in a new version of the entire vehicle. This method is clearly storage intensive and impractical.

A second option is to save every model in its own file and manage a library of the models. This would be an improvement over the first option; however, it still presents some difficulties. When a user wishes to create a new vehicle, he or she has to select all of the appropriate models from the library and connect them by hand into a vehicle context. Not only is this manual process time consuming, but it introduces many opportunities for error. Autonomie uses a novel approach that combines the second option with an automated building process. This gives the user the flexibility of saving and versioning models independently without potential pitfalls of manual connections. Users select the correct files in a user interface, and the automatic building uses metadata associated with the models to create the correct connections, as shown in Figure 2.
Building entire vehicle models automatically offers the advantage that any configuration can be quickly implemented. This is particularly important for electric drive vehicles such as HEVs, PHEVs or EVs.

However, a limitation of the current process is the need to have Matlab, Simulink and StateFlow licenses, which leads to additional licensing cost. While it is feasible to design a version of Autonomie that does not require MathWorks toolboxes, one would lose the flexibility needed for some powertrain configurations, especially HEVs.

### 3.2. Parameters Selection and Modifications

Autonomie provides the ability to select whether each parameter can be accessible from the GUI as shown in Figure 4. The tool uses an XML file to assign several properties to each parameter, including: viewable (yes/no) and locked (yes/no).

That simple and efficient process allows labeling agencies to quickly add or delete any parameter based on the evolution of the technologies and models. The protection of the XML files to avoid any unwanted modifications is discussed later.

### 3.3. Files Selection and Implementation

While it is also expected that some files, including initialization, models or controls will be initially frozen, the tool allows users to only view and/or modify the files allowed for a specific version of the labeling tool as shown in Figure 5. One might for example consider some component characteristics to be initially fixed while allowing others to be modified, although this would lead to significant issues since the entire vehicle would not be properly modeled. Another possibility would be to allow selection of component models in the case of specific technologies that could not be properly represented by the default model.

Similarly than for the parameters, a generic process has been implemented into Autonomie through the XML files to easily select only the files that can be viewed and/or modified by the users. The protection of the XML files to avoid any unwanted modifications is discussed later.

### 3.4. Ensuring Simulation Validity

Autonomie has been validated over the past decade for numerous powertrain configuration and vehicle
applications. One of the critical aspects of medium and heavy duty vehicle is their specific shifting algorithm, especially gear skipping for line haul applications. Figure 7 shows an example of validation performed on a conventional line haul vehicle [9].

![Figure 7: Gear Number Comparison of a Line Haul Vehicle](image)

For each vehicle simulation performed, users shall be able to view the main results of the simulation, including fuel consumption, how well the drive cycle was followed... But they should also be able to analyze their vehicle behavior since a combination of specific sets of parameters might lead to issues. Autonomie provides numerous tools as shown in Figure 8 to support a detailed analysis of the vehicle behavior, including:

- Generic post-processing (i.e., power, energy, efficiencies)
- Pre-defined plots for each component models
- Ability to quickly view any signal
- Ability to replay the simulation

![Figure 8: Autonomie Post-processing](image)

3.5. Handling Future Regulations

While it is most likely that the labeling process focuses initially on a simple version, future regulations will evolve in many different ways. One possibility is to allow use of component models developed in expert languages (i.e., engine model in GT-Power). Another one is the use of different steps of Model Based Design (i.e., HIL as currently used in Japan).

Autonomie already allows both of these evolutions. Figure 9 shows a current vehicle model solely based on expert languages: engine from GT-Power, transmission from AMESim and vehicle dynamics from TruckSim. While the development and implementation of such models is more complex, they will likely be necessary to properly represent future specific technologies. The building algorithm combined with the use of XML files describing the models allows such flexibility.

![Figure 9: Vehicle Model Using GT-Power, AMESim and TruckSim](image)

On another side, different steps of MBD have already been demonstrated with Autonomie [10] through project at Argonne and General Motors. These include Model in the Loop, Software in the Loop and Component-in the loop as shown in Figure 10.

![Figure 10: Engine-in-the-Loop Performed at Argonne](image)

3.6. Consistent Report Generation

To ensure results traceability and verification of the results, a generic report is generated. That report will then be provided to the labeling agency for analysis. The process implemented in Autonomie allows the definition of a specific report for regulation based on a GUI as shown in Figure 11.
3.7. Database Management

A database management is necessary to store and search all the results for specific applications, both for the manufacturers and the labeling agency. Figure 12 shows the generic process implemented into Autonomie to support database management.

While the communication with specific databases requires particular coding, the process remains the same and can be easily adapted.

3.8. Ensuring Simulation Integrity

In order to ensure the integrity of the simulation, it is preferable to protect as much information as possible, at least during the first stages of the labeling.

Autonomie is based on Matlab, Simulink and StateFlow. As a consequence, all the models, controls and initialization files are open. To prevent any modifications, Sfunctions have been developed for each component model and controller to ensure that no modifications can be performed. All the equations are still provided to the users through the documentation.

Similarly, processes (i.e., drive cycle or report selection) cannot be selected by the users when running labeling simulations. For example, when selecting line haul applications, the same drive cycle, post-processing and report will always be selected.

Finally, in order to prevent users from modifying anything (changing parameters, modifying locked files, turning off locking altogether, etc.) the XML will be encrypted, and thus will not be editable. Encrypted XML files will be automatically detected and unencrypted by the GUI (and the Matlab side) so that the rest of the code doesn’t know if the XML was encrypted or not.

4. Conclusion

Labeling of medium and heavy duty applications is necessary to maximize fuel displacement of transportation activities. In order to achieve that goal, modeling and simulation tools, along with other steps of Model Based Design, are necessary due to the complexity and diversity of the vehicles. Since the process used for labeling will evolve over time, the tool should meet numerous requirements, from flexibility to reusability and user-friendliness.

Autonomie has been initially developed as a plug & play software architecture to meet the different steps of Vehicle Development Process throughout the different steps of Model Based Design. The software architecture is very well suited to any type of labeling implementation as well as their future evolution.

5. References

4. www.gtisoft.com
5. www.amesim.com/
6. www.carsim.com
7. www.mathworks.com/products/simscape
6. Acknowledgment

This work was supported by DOE’s Vehicle Technology Office under the direction of Lee Slezak. The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.

7. Authors

Aymeric Rousseau
Program Manager
Argonne National Laboratory,
9700 S. Cass avenue,
Argonne, IL 60439, USA
Tel: +1-630-252-7261
Email: arousseau@anl.gov
Aymeric Rousseau received his Master of Science in Industrial System from EIGSI in La Rochelle, France in 1997. He is currently leading ANL’s Vehicle modeling and simulation group. He received an R&D 100 Award in 2004 and an FLC award in 2007.

Philip Sharer
Research Engineer
Argonne National Laboratory,
9700 S. Cass avenue,
Argonne, IL 60439, USA
Tel: +1-630-252-9739
Email: psharer@anl.gov
Phil Sharer received his Master of Science in Engineering from Purdue University Calumet in 2002. He has been a member of the Modeling and Simulation group at Argonne for 11 years. He is a developer on the Autonomie project and has authored two key features of Autonomie: the Autonomie Model Building and the Autonomie Process Runtime.

Shane Halbach
Research Engineer
Argonne National Laboratory,
9700 S. Cass avenue,
Argonne, IL 60439, USA
Tel: +1-630-252-9739
Email: shallbach@anl.gov
Shane Halbach is the lead software engineer on the Autonomie software platform, as well as the Powertrain Systems Analysis Toolkit (PSAT). He has been working in software for over 8 years, and has a Masters in Software Engineering from Penn State.