

# APPLICATIONS OF MODEL-BASED SYSTEMS ENGINEERING METHODS TO VEHICLE AND SUBSYSTEM DESIGN AND OPTIMIZATION

Dr. Larry Michaels, Shane Halbach, Neeraj Shidore, Aymeric Rousseau  
Argonne National Laboratory  
Ground Vehicle Systems Engineering and Technology Symposium  
Systems Engineering Mini-Symposium  
Approved For Public Release

## **Summary**

This paper discusses the latest techniques in vehicle modeling and simulation to support ground vehicle performance and fuel economy studies, enable system design optimization, and facilitate detailed control system design. The Autonomie software package, developed at Argonne National Laboratory, is described with emphasis on its capabilities to support Model-in-the-Loop, Software-in-the-Loop (SIL), Component-in-the-Loop (CIL), and Hardware-in-the-Loop simulations.

Autonomie supports Model-Based Systems Engineering, which is growing in use as ground vehicles become more sophisticated and complex, with many more subsystems interacting within the vehicle and the environmental conditions in which the vehicles operate becoming more challenging and varied. With the advent of hybrid powertrains, the additional dimension of vehicle architecture has become one of the design variables that must be considered. This complexity results in the need for a simulation tool that is capable of incorporating systems of various levels of detail and fidelity, as well as handling scalable simulation tasks, from individual component models up to complete vehicles.

Autonomie automates the process of building system simulations comprising the models chosen by the user, and then executes those simulations, producing tabular and graphical results. This capability eliminates the need for many of the manual steps that are normally encountered when creating and running a system simulation model.

Specific applications are described to show the applicability of Autonomie to a wide range of problems. These applications highlight the use of Autonomie for SIL-based detailed control design using compiled production control software, ground vehicle performance and fuel economy optimization studies, and CIL testing at Argonne National Laboratory.

## **Introduction**

Model-Based Systems Engineering (MBSE) improves quality, shortens development time, lowers engineering cost, and reduces rework. Evaluating a system's performance, functionality, and robustness in a simulation environment avoids the time and expense of developing hardware

and software for each design iteration. Simulating the performance of a design can be straightforward (though sometimes tedious, depending on the complexity of the system being developed), with mathematical models for the hardware components of the system (plant models) and control algorithms for embedded controllers.

Model-Based Systems Engineering has been widely accepted in the automotive industry. Designing and studying the performance of a system in the virtual environment of a computer system simulation

- Provides a math-based environment for thorough multidisciplinary integration and testing prior to hardware builds;
- Permits sorting technologies quickly so as to reduce hardware build iterations;
- Promotes parallel and integrated virtual development of control systems and hardware;
- Delivers better-integrated initial designs that balance fuel economy, emissions, and drivability requirements;
- Provides methods and tools for comparing/evaluating technologies; and
- Facilitates efficient, seamless linking from research to production to maximize reuse of work products.

### **System Simulation**

Model-Based Systems Engineering calls for development of a complete system simulation, which includes plant models that represent the dynamic behavior of the physical system(s), models of the sensors and actuators that are used to connect the physical systems to their controllers, and models of the controller hardware and algorithm software. This system simulation environment is then used to perform various types of studies, including the following:

- Analysis of system performance and fuel economy;
- Analysis of variation and robustness; and
- Detailed control algorithm design

The structure of a system simulation is depicted in Figure 1:

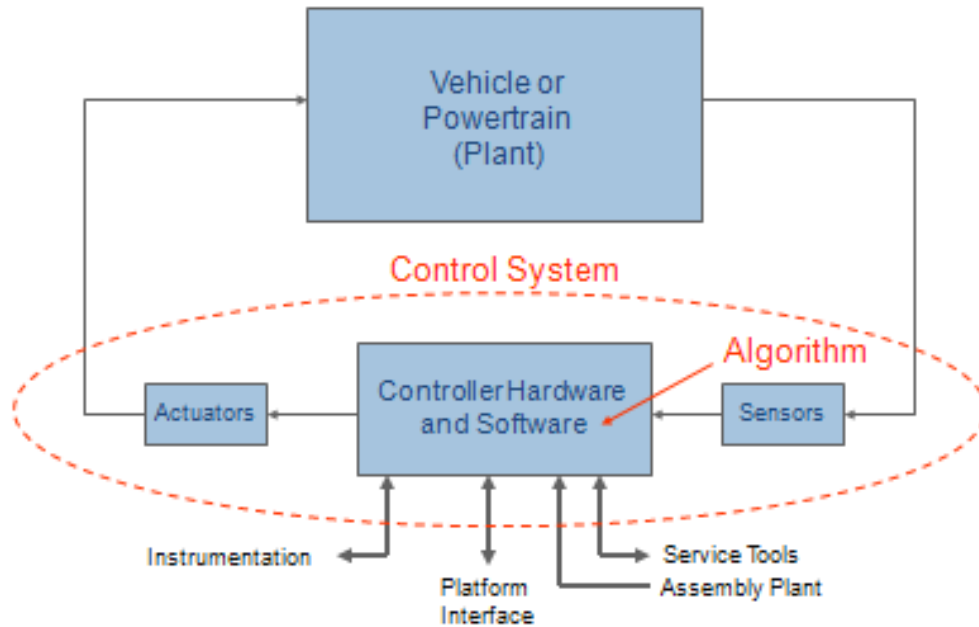


Figure 1 – System simulation

### **Challenges**

Developing a complete system simulation for a ground vehicle application can be a challenging task. Complex plant models, models of sensors and actuators, algorithm models, and production software modules must be connected. Some of the subsystem blocks have hundreds of inputs and outputs, which makes manually connecting them a time-consuming and error-prone process. To obtain a complete system model, the functionalities of all the control algorithms must be included—but in many cases, models of all the algorithms are not available and compiled controller production code must be used.

For a system simulation to be effective and structured for the task at hand, users must be able to readily select the level of fidelity for each of the subsystem models. There also must be a mechanism to manage all of the artifacts of the system simulation, including the models themselves, initialization and calibration files to be used with the models, test cases and drive cycles used to exercise the system simulation, and results files. These artifacts also need to be configuration-managed and version-controlled.

### **Autonomie Description and Features**

The most important aspect of a tool that supports full MBSE is flexibility. A model is only useful and reusable if it can quickly be configured for any number of different usages. Therefore, it is not useful to version a full vehicle model, but rather to break that model down into smaller pieces, managing the individual models independently. The models can then be reused in the widest possible fashion, from research to production, because they can be easily recombined in infinite combination with any other model. Autonomie provides several key features to facilitate this flexibility.

The first feature is Autonomie's open architecture, which allows any configuration of groupings of models, with recursive levels of subsystems. This feature is essential. Groupings of models, along with initialization, scaling, and processing files as well as any tuned or overwritten parameter values, can be lumped together as a "system." This capability allows the systems to be abstracted and treated as black boxes; they can be independently validated, shared, configuration-managed, and swapped out in the vehicle model by users who are not familiar with the details of these systems, independently of context. This capability allows experts to design and optimize their individual systems to be reused by non-experts, naturally leading to better modelling throughout the MBSE process.

For example, a battery expert can create several battery plant and control models, varying from high-level, lookup-table models to very detailed, high-fidelity models (even using expert tools where necessary, such as AMESim, Saber, or GTPower). Using Autonomie, the battery expert could run the models through processes to test inputs and outputs, compare the models to test data from actual batteries, and tune the model parameters using optimization routines. He might also develop specialized systems to handle connections to hardware, such as Component –in-the-Loop (CIL) or Hardware-in-the-Loop (HIL). Once the expert was satisfied with the system (or set of systems), he could share it (i.e., via source control).

Similarly, other experts would develop their systems in parallel, e.g., an engine expert would develop a range of engine systems. Separate vehicle integration experts could then pick and choose among these systems to develop custom vehicle models for any application. So long as a tool such as Autonomie is used, the expert can quickly assemble high-level, technology-sorting vehicle models, low-level vehicle models to investigate detailed phenomena, distributed models for co-simulation, or anything in between. This approach is depicted in Figure 2.

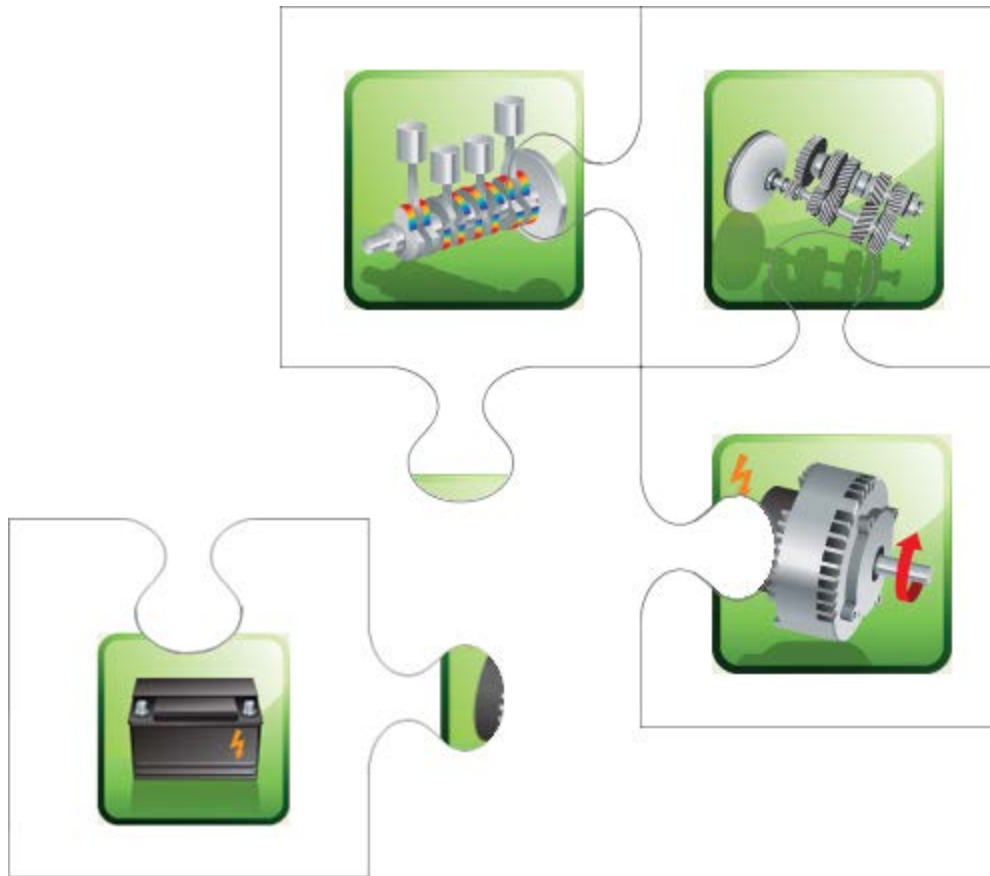


Figure 2 – Autonomie seamlessly integrates individual systems

Because Autonomie allows the flexibility to organize the systems in any way, an organization can be selected that most easily allows for the substitution of systems. Usually, this selection is achieved by echoing the structure of the actual vehicle. Separating the models into systems that represent components of the vehicle, and co-locating the appropriate controller models with the appropriate plant models, allows for the maximum reuse and “swappability” of the various systems.

For example, if an engine system is itself composed of a plant and a controller system, it can be applied to a range of uses. The entire engine system could be replaced with a system capable of running CIL. Alternatively, the plant system could be swapped for an HIL system running with a Simulink controller model. Or the plant system could be replaced with a more or less detailed system appropriate to the level of detail of the study; either one that further breaks down the models into additional, swappable subsystems, or perhaps one developed in an expert modeling tool. Finally, just the controller system could be swapped for a system containing compiled code for a Software-in-the-Loop (SIL) simulation (see Figure 3).

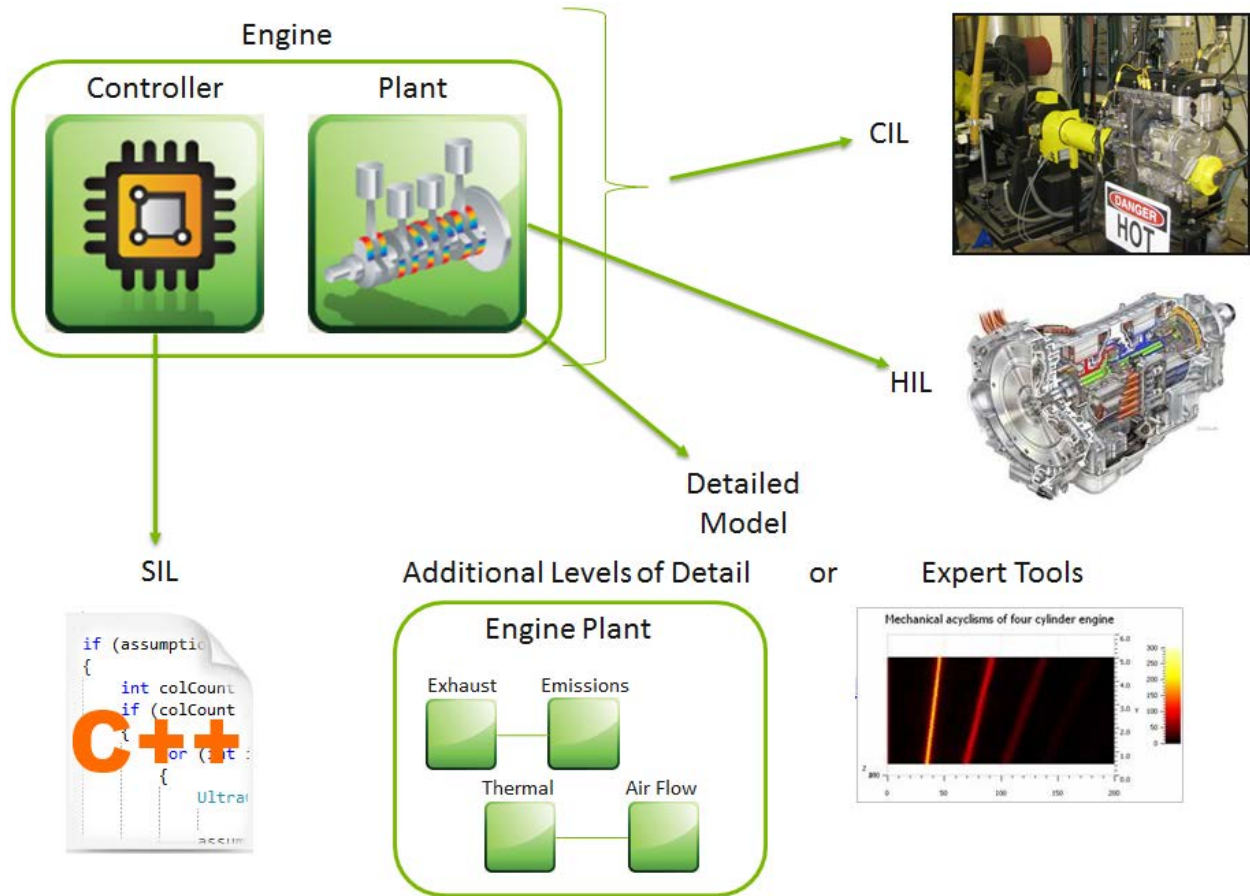


Figure 3 - Autonomie facilitates swappable subsystems for maximum model reusability

Of course, flexibility doesn't just happen; Autonomie provides three more interrelated key features that enable system plug-and-play: 1) automatic model connections, 2) compatibility checking, and 3) automatic model building.

With increasing model complexity comes an increasing number of connections between models. Connecting these models is time-consuming and error-prone, making the swapping of models a daunting task. Autonomie automates this task by routing the correct information to the correct model ports without user interaction. If the correct signal cannot be found for automatic routing, the user interface provides a simple way to select the proper signal and to save that information for later reuse.

This mechanism is dependent on Autonomie's robust compatibility checking, which ensures that all inputs are satisfied in the system by compatible signals. Without this second key feature, automatically connecting any possible system would be impossible. For example, if a very detailed plant model is selected without the corresponding level of detail in the controller model, then the plant model will most likely be missing the inputs required to exercise it. Or a detailed

engine system may require a correspondingly detailed transmission system to provide the correct feedback.

Autonomie puts it all together with the third key feature: automatic model building. Without this feature, all the rest is meaningless. The automatic model-building algorithm assembles the individual systems into a single integrated model without requiring the user to do this tedious work by hand.

Model-Based Systems Engineering is only as good as the models selected for the simulation. To this end, it is very important to test and validate models. Autonomie provides capabilities that facilitate this task too.

Autonomie allows systems to be simulated at any level, from individual plant models, to collections of systems representing entire components, to entire vehicles. Component-only and model test bench processes provide user interfaces to quickly exercise model inputs and check outputs before testing an integrated vehicle on a standard or user-created drive cycle. This capability allows multiple points of validation, as each level of integration can be tested. Therefore, it is possible to design and optimize individual models, sets of models, or entire vehicles.

Autonomie also provides the ability to import test data and compare them with simulation data, which is critical for model validation.

Just as Autonomie allows for the construction of a full vehicle model from its constituent parts, Autonomie also allows the constructing of processes to exercise the models. Depending on the model and what you want to do with it, different steps may be required to exercise it. For example, testing an individual model's inputs and outputs is different from testing a complete, self-contained system, and performing an SIL or HIL process would be different from running a drive cycle. Users can modify existing processes by adding steps or providing modifiers, or they can provide a process of their own.

Autonomie provides default processes and process modifiers to aid in design, such as parametric study and optimization routines, as well as distributed computing functionalities to aid in batch-running large sets of simulations. With the use of modern computing hardware, typically computationally expensive optimization problems such as multi-parameter tuning, component sizing, or Monte Carlo analysis are possible through Autonomie.

Finally, Autonomie provides advanced data analysis capabilities to quickly determine the results of a simulation by looking at raw numbers or plotting signals, either for individual runs or for

side-by-side comparisons of multiple runs of simulation and/or test data. This capability allows for efficient verification and validation of models as well as the quick identification of optimum parameter values.

### **Application Examples**

#### ***Application of Autonomie to Software-in-the-Loop (SIL) Simulations***

To build a complete system simulation, the functionalities of all components and subsystems must be included. In many cases, although plant models are already available or can be readily developed, models of many control algorithms are not available, and the only representation of the algorithm functionality is contained in existing software. In these cases, a full system simulation can usually be performed only when a complete software build has been created and loaded into the target embedded controller. Then the control algorithm software can be tested in an HIL system, where the controller with its application software is connected to a real-time computer system that runs a simulation of the plant model(s) representing the physical system to which the controller will eventually be connected.

However, it is now possible to use compiled production code in the virtual environment. In this so-called SIL technique, compiled production controller code is incorporated into the simulation environment, thus allowing the inclusion of algorithm functionality for which no simulation models exist. This approach allows a system simulation to be constructed even when models of all the control algorithms are not available, thus yielding what may be referred to as a virtual HIL simulation. The system simulation includes an SIL block that incorporates a compiled version of the production software used in the actual embedded controllers.

Figure 4 shows the SIL-based system simulation in the Autonomie environment.



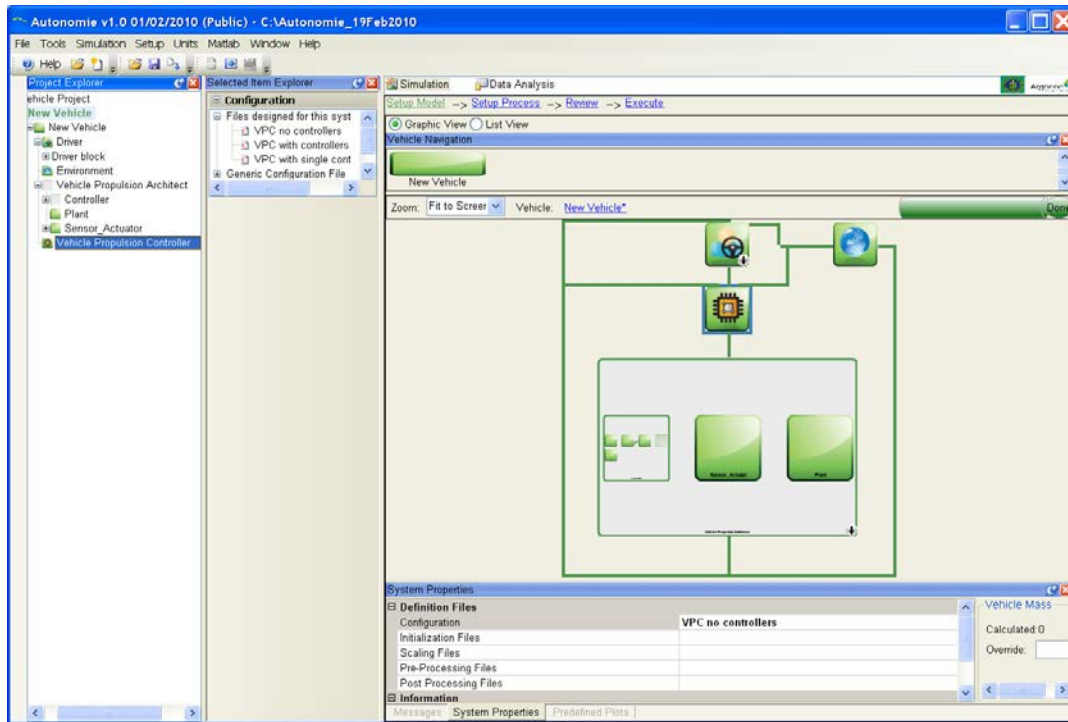


Figure 4 – Autonomie SIL-based system simulation

The controller model structure, shown in Figure 5, includes the following:

- A block emulating the functionality of the embedded-controller real-time operating system (RTOS), which schedules the execution of the tasks in the compiled software and any attached algorithm models (this RTOS block consists of a behavioral model of the actual RTOS used in the embedded controller, and is implemented in Simulink);
- A Controller Area Network communication block to supply serial data signals to the compiled software;
- An algorithm model block that contains any algorithms under development;
- A hardware I/O (HWIO) block that contains behavioral models of the HWIO functions in the controller; and
- An SIL block that permits inclusion of the compiled production code into the simulation model. An SIL block is required for each controller included in the system, thus allowing for multiple Embedded Control Units (ECUs) to be represented.

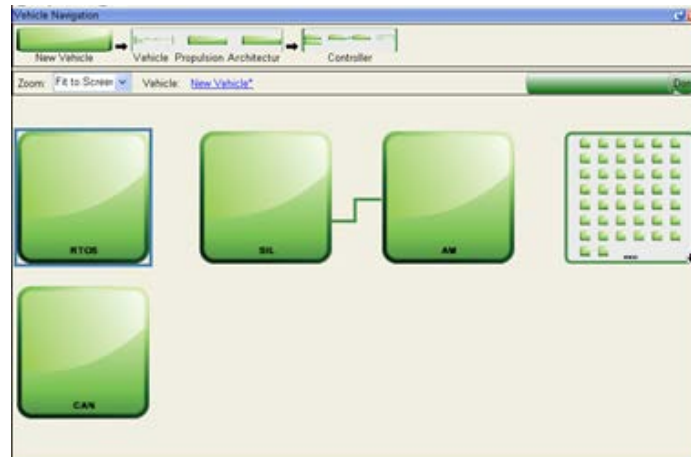


Figure 5 – Controller model structure

### ***Application of Autonomie to Component-in-the-Loop (CIL)***

It is difficult to obtain accurate models of certain phenomena (e.g., thermal behavior, emissions, after-treatment) that can run quickly enough for a system simulation tool. In such cases, real hardware components or subsystems are combined with real-time software simulation models of the rest of the powertrain for vehicle system analysis. This process is commonly known as component-in-the-loop (CIL) or component-hardware-in-the-loop [10]. Vehicle control strategies (energy management), which need feedback or inputs from such phenomena (e.g., cold start control), are evaluated using CIL. Another potential application of CIL is to evaluate prototype components in a vehicle system environment, where a mathematical model of the prototype model has yet to be developed because research and development of the component are ongoing [11].

The flexibility provided by Autonomie can be used to perform Model-Based Design Experiments (MBDE, a subset of MBSE) with CIL. Argonne National Laboratory has used the MBDE process for CIL in numerous experiments pertaining to advanced vehicle technologies, which needed a real battery [12] or a real engine [13].

In Autonomie, a separate configuration has been created for CIL experiments, which enables seamless transition between simulation and CIL. The generic component system configuration for any component is shown in Figure 6.

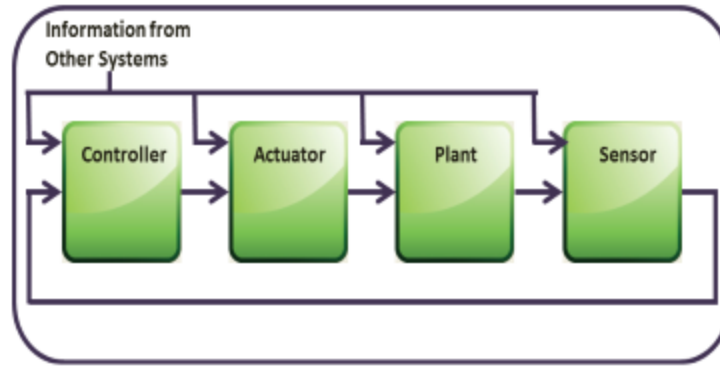


Figure 6 – Generic component configuration in Autonomie

This configuration is replaced with a configuration suitable for a CIL experiment, as shown in Figure 7.

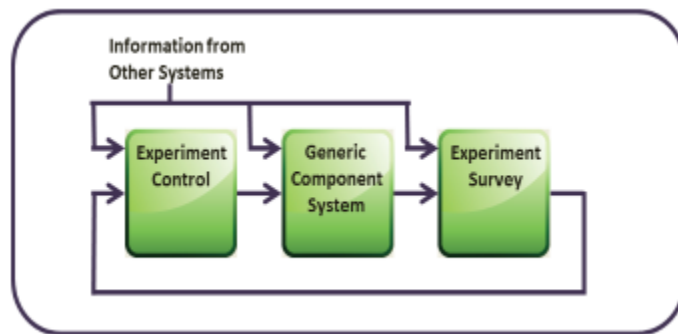


Figure 7 – Component configuration for component-in-the-loop experiments

The experiment control system has specific subsystems that enable the user to test the hardware component outside the vehicle environment and write low-level control code for the hardware component. The component system is essentially made up of I/O blocks that interface with the actual hardware component. The experiment survey monitors feedback signals and initiates proper action (including Estop) on the basis of the status of the feedback signal. Details of the CIL configuration are available in Reference [14].

Once the CIL system has been developed, the user can move from the simulation world to CIL by simply dragging the CIL configuration onto the appropriate component. The code can then be deployed to any real-time operating system for CIL experiments. Similarly, the user can get back to the simulation by simply dragging the default Autonomie component configuration (Figure 6) back onto the component.

This seamless transition between CIL and simulation enables an MBSE process, as shown in Figure 8.

Model-Based Systems Engineering was used to evaluate the fuel economy potential of a power-split PHEV, with a real engine (Engine-in-the-Loop [EIL]) to take emissions into account. Two parameters (delta SOC and wheel power threshold at which the engine turns ON) were varied in order to achieve the best possible fuel economy while maintaining emissions below a predefined standard. Figure 8 highlights the key steps in the MBSE process, which are possible because of the flexibility provided by Autonomie. The default Autonomie power-split PHEV vehicle was run with the real engine to identify control modifications needed to reduce emissions (torque shaping, engine warm-up). Once the controls were modified, a large simulation study was conducted with the modified controls to determine the values of delta\_SOC and wheel power threshold that provided the maximum fuel economy for the vehicle. Since only the engine configuration changes in the transition from CIL to simulation, it is relatively easy to change from CIL to simulation. Also, because the only difference between the CIL and simulation versions is the real/virtual engine, the model can be easily adjusted to match hardware behavior.

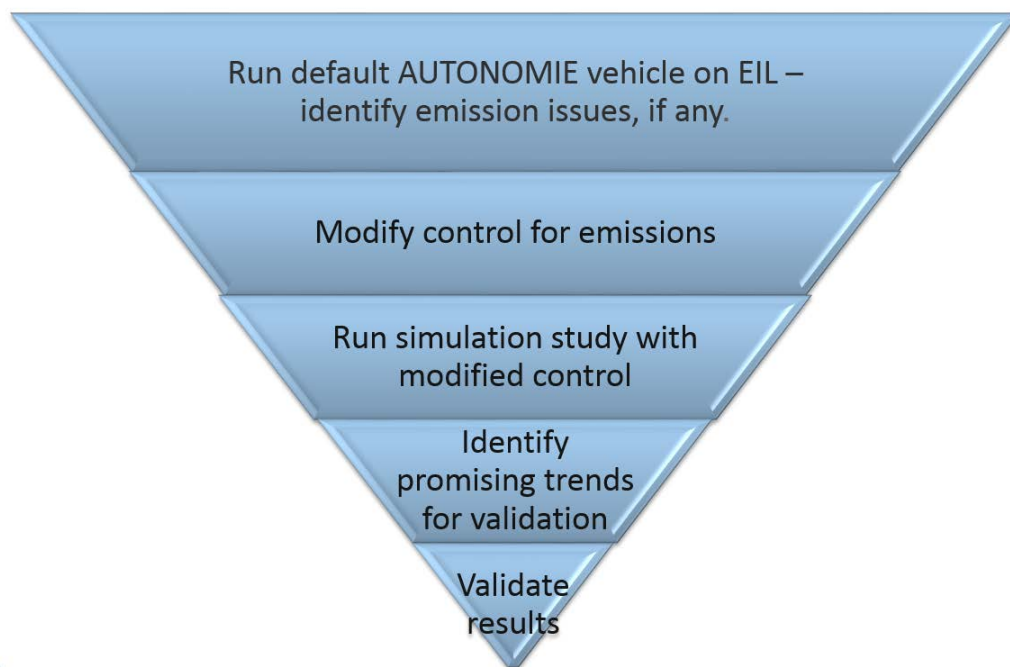


Figure 8 – MBSE process for an engine-in-the-loop experiment

Figure 9 shows results for fuel consumption as a function of delta SOC and engine-ON threshold. As can be seen from the figure, there is a minimum at a delta SOC of around 0.25 and an engine-ON threshold of around 12 kW, which gives the lowest fuel consumption or the highest fuel economy.

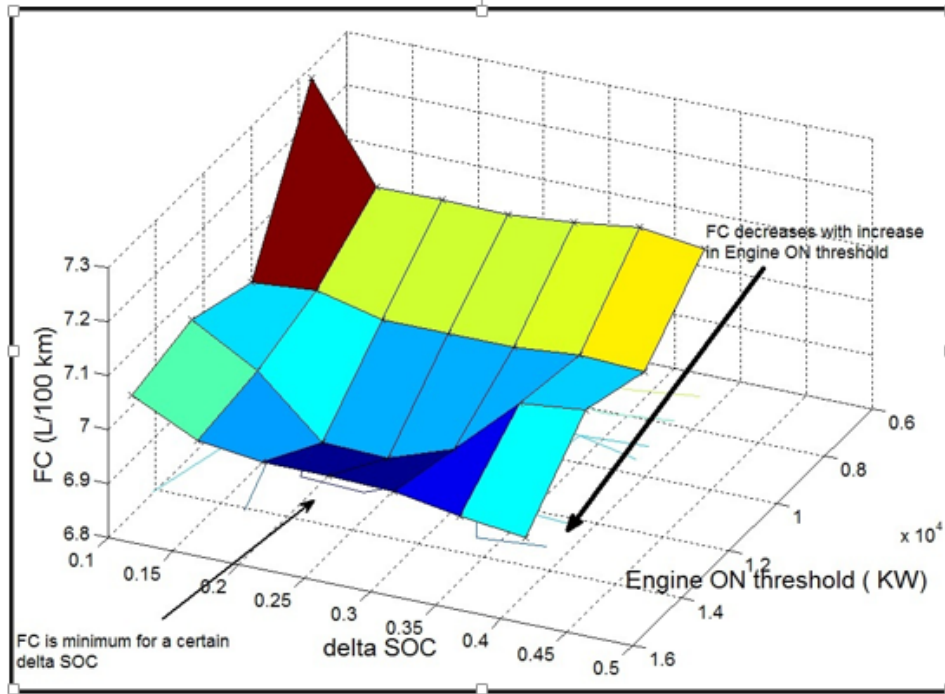


Figure 9 – Fuel consumption as a function of delta SOC and engine-ON threshold

Certain simulation results (including the minimum and certain results around it) were chosen for final validation with hardware (EIL), and to make sure that the emissions for the minimum were still within predefined limits (Figure 10). This is the final step of the MBSE process.

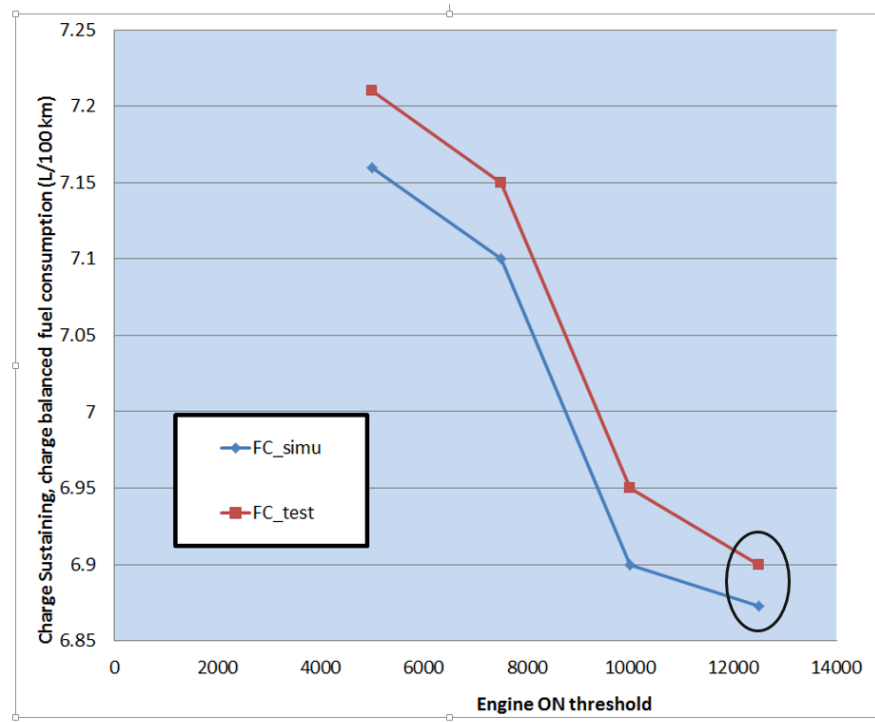


Figure 10 – Validation of simulation results with EIL (last step of MBSE process)

The MBSE approach, highlighted above for the EIL experiment, is possible because of the seamless transition between simulation and CIL. It greatly reduces the number of tests done on hardware, since most of the design matrix is explored in simulation.

## **Summary and Conclusions**

### ***Performance and Fuel Economy Studies***

Autonomie's automated model-building and simulation process capabilities are used extensively for vehicle performance and fuel economy studies. It is easy and straightforward to set up simulation runs that will allow the user to perform comparative analyses of different vehicle architectures and component technologies, in order to do a trade-off study of multiple designs. Autonomie is also used to run component sizing studies and determine, for example, the best battery size and capacity for a Hybrid-Electric Vehicle.

The ability to readily configure vehicles for simulation runs, and then perform those runs by running predefined or customized processes, greatly reduces the burden on the user to set up and execute simulation runs of dozens or hundreds of vehicles in order to obtain the desired analysis results.

### ***Design Optimization***

In addition to being able to run parametric studies, Autonomie also supports the integration of advanced optimization algorithms that enable a user to perform multi-objective optimization studies to determine the best vehicle architecture and component specifications to meet overall design requirements that cover a variety of operational scenarios. These optimization runs can also take into account the variation that occurs in vehicle operation because of manufacturing variances, environmental changes, and product aging. Optimizing the system across product life-cycle variations enhances the development of robust designs.

### ***Control Design Applications***

The methods provided by Autonomie enable a control system design capability that was heretofore very difficult to achieve. Control algorithm development in the context of a complete system simulation provides many advantages that result in high-quality, robust control systems. Engineers are able to account for interactions among the many functions, subsystems, and controllers that comprise modern automotive controls, and to study the effects of variation on the system's robustness. The approach developed cooperatively between GM and Argonne National Laboratory overcomes the challenges and difficulties of constructing a complete system simulation.

The Autonomie software developed at Argonne facilitates management of the models, including automatic interconnection of the subsystems. Autonomie provides a complete environment for assembling, building, running, and analyzing complete vehicle (or vehicle subsystem) models. The SIL capability developed at GM allows the user to include a complete representation of production control algorithm software, thus permitting the simulation to include the functionality of algorithms for which no simulation models exist. The SIL facility provides complete flexibility in scheduling and executing the production code tasks, as well as mixing and matching them with algorithm models represented in Simulink. This flexibility permits the engineer to run the same tests and experiments that are normally performed on an HIL simulator, and provides the capability afforded by a rapid controller prototyping system running in bypass mode.

Autonomie is based on a MATLAB/Simulink platform, which may encounter computer performance limitations for very large models. The SIL feature avoids many of these performance issues by including compiled C code for all production control algorithms except the one under development. Even so, there may be situations where simulation speed is adversely impacted by the model size. In such cases, co-simulation technology may be employed to distribute the computing workload among multiple processors or computers.

---

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.

## REFERENCES

1. Michaels, L. (Argonne National Laboratory), Kropinski, M. (GM), "Development of Production Control Algorithms for Hybrid Electric Vehicles by Using System Simulation," SAE 2012-01-9008, SAE Convergence Conference, October 2012.
2. Michaels, L., et al. (Argonne National Laboratory), Kropinski, M., et al. (GM), "Model-Based Systems Engineering and Control System Development via Virtual Hardware-in-the-Loop Simulation," SAE 2010-01-2325, SAE Convergence Conference, October 2010.
3. Argonne National Laboratory, Autonomie Documentation, Argonne, IL, 2009.
4. Argonne National Laboratory, Autonomie Presentation, <http://www.transportation.anl.gov/pdfs/MC/578.pdf>, 2009.
5. Smith, P., Prabhu, S., Friedman, J. (The MathWorks), "Best Practices for Establishing a Model-Based Design Culture," SAE 2007-01-0777, SAE World Congress, 2007.
6. Tung, J. (The MathWorks), "From Specification and Design to Implementation and Test," Embedded World 2004 Conference, Automotive Digest, April 2004, [http://www.mathworks.com/company/newsletters/auto\\_digest/april04/embed\\_world2004.html](http://www.mathworks.com/company/newsletters/auto_digest/april04/embed_world2004.html).
7. Matthews, O., Michaels, L. (GM), "GM Powertrain Automatic Code Generation Process," Presentation at MathWorks International Automotive Conference, June 2005.
8. Vijayagopal, R., et al. (Argonne National Laboratory), "Automated Model Based Design Process to Evaluate Advanced Component Technologies," SAE 2010-01-0936, SAE World Congress, 2010.
9. FreedomCAR and Fuel Partnership, United States Council for Automotive Research (USCAR), United States Department of Energy, [http://www.uscar.org/guest/view\\_partnership.php?partnership\\_id=1](http://www.uscar.org/guest/view_partnership.php?partnership_id=1).
10. Shidore, N., Lohse-Busch, H., Smith, R.W., Bohn, T. (Argonne National Laboratory), "Component and Subsystem Evaluation in a Systems Context Using Hardware in the Loop," Presentation at IEEE Vehicle Power and Propulsion Conference, 2007.
11. Shidore, N., Ciatti, S., et al. (Argonne National Laboratory), "Evaluation of the Fuel Economy Impacts of Low Temperature Combustion (LTC) using Engine-in-the-Loop," Presentation at Department of Energy's 2013 Hydrogen Program and Vehicle Technologies Annual Merit Review, May 2013.
12. Shidore, N. Bohn, T. (Argonne National Laboratory), "Evaluation of Cold Temperature Performance of the JCS-VL41M PHEV Battery Using Battery HIL," SAE 2008-01-1333, SAE World Congress, April 2008.
13. Shidore, N., Ickes, A., Wallner, T., Rousseau, A. (Argonne National Laboratory), "Evaluation of Ethanol Blends for PHEVs Using Engine in the Loop," Presentation at the IEEE Vehicle Power and Propulsion Conference, 2011.
14. [http://www.autonomie.net/overview/presentations\\_CIL.html](http://www.autonomie.net/overview/presentations_CIL.html).

## CONTACT INFORMATION

ARGONNE NATIONAL LABORATORY

Aymeric Rousseau

(630) 252-7261

Email: [arousseau@anl.gov](mailto:arousseau@anl.gov)