

# Rapid Partitioning, Automatic Assembly and Multi-core Simulation of Distributed Vehicle Systems

Sylvain Pagerit, Phil Sharer,  
Aymeric Rousseau  
Center for Transportation  
Research  
Argonne National Laboratory  
Lemont, USA  
[spagerit@anl.gov](mailto:spagerit@anl.gov),  
[psharer@anl.gov](mailto:psharer@anl.gov),  
[arousseau@anl.gov](mailto:arousseau@anl.gov)

Qinwei Sun, Mike  
Kropinski  
GM Powertrain  
General Motors, LLC  
Milford, USA  
[Qinwei.sun@gm.com](mailto:Qinwei.sun@gm.com),  
[michael.kropinski@gm.com](mailto:michael.kropinski@gm.com)

Neville Clark, James  
Torossian  
EST Embedded Systems  
Technology, P/L  
Sydney Australia  
[n.clark@esstek.com](mailto:n.clark@esstek.com)  
[j.torossian@esstek.com](mailto:j.torossian@esstek.com)

Graham Hellestrand  
Embedded Systems  
Technology, Inc.  
San Carlos, USA  
[g.hellestrand@esstek.com](mailto:g.hellestrand@esstek.com)

**Abstract**— The advent of heterogeneous specifications of complex systems utilizing both continuous (viz. physics or maths based models) and discrete domain models, communicating via accurate network models, enables a generalization of the usual model-based design methodology. The use of individual notations appropriate for sensor, actuator, plant, controller and accurate network modeling, enable both the natural specification of various models and the continued usage of legacy libraries containing models described in these various notations. The ability to choose fixed or variable-step solvers individually for the simulation of each continuous domain model in a system, enables designers to choose levels of accuracy of models, appropriate network communication models, and speeds of simulation.

In this paper, we describe how engineers can rapidly define the connection of hierarchies of modules contained in Simulink models, and their topography distributed across multi-core computers, for fast and accurate simulation.

**Keywords**—*Distributed simulation, partitioning and automatic model re-assembly, accurate network communication, vehicle systems, multi-core simulation, model based design and engineering*

## I. INTRODUCTION

Model-Based Design (MBD) process has many definitions [1, 2, 3]. The conventional definition is a math-based, frequently visual method for designing relatively complex systems comprised of sensor, actuator and plant modules (physical sub-systems), modules that control those physical sub-systems and the system model that contains them. The classical and decades old V process is the marker of this process which splits design from implementation and verification as a function of implementation (see Fig. 1). The MBD process is being used successfully in many motion-control, industrial, aerospace, and automotive applications. It provides an efficient methodology that includes four key elements in the development process: (i) modeling the plant and its control (from first principles or system identification), (ii) synthesizing and analyzing a controller for the plant, (iii) simulating the plant and controller together, and (iv) programming/deploying the controller. MBD integrates these

multiple phases and provides a common framework for communication throughout the entire design process.

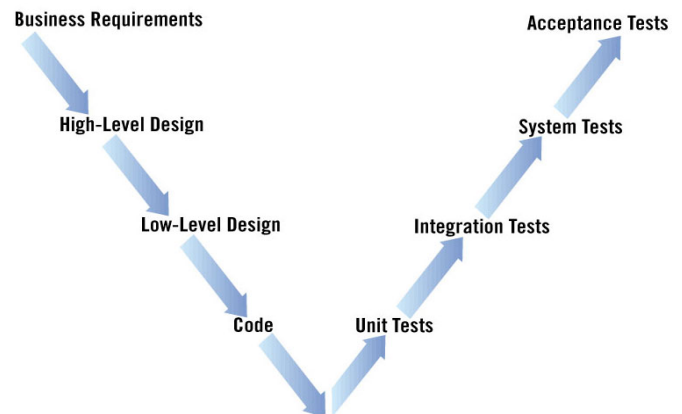


Fig. 1. V Diagram for the Conventional MBD Process

The MBD paradigm is significantly different from traditional design methodology. Rather than using complex structures and extensive software code, designers can formulate advanced functional characteristics by using continuous-time and discrete-time computational building blocks. These models and associated simulation support tools can provide rapid prototyping, virtual functional verification, and software testing and hardware/software validation. MBD is a process that enables faster, more cost-effective development of dynamic systems, including control systems, signal processing, and communications systems.

However, as systems have become more complex and systems of systems have entered the engineering domain Model Based Engineering (MBE) [4, 5] has evolved from MBD to address the new complexity. MBE is a top down model-based process driven by rapid prototyping and empiricism, and expects to operate in large state-spaces. Optimization, statistics and the rules of scientific experimentation are fundamental tools of MBE. The MBE process is briefly outlined (see Fig. 2): From **Requirements**, which include System Acceptance Tests, are derived **Executable Specifications** - parameterized continuous and

discrete domain models - which communicate with latency, and that correctly execute the Acceptance Tests; candidate **Architectures** are particular parameterizations of the Specifications and hence pass the Acceptance Tests; **Design** typically extracts control and plant from an architecture model and further tests are created to verify the correct operation of control and plant in the context of the network connections; and **Realization** is, the mapping of control functions to state machines or the generation software code for particular ECUs, executives and device drivers, appropriate for the plant that it controls and communication with other functions – spans of control and effect.

Specifications, Architectures and Designs are executable entities that are refined and elaborated throughout the model-based engineering process. Verification runs in parallel with elaboration wherein each level of the process must pass all of the test created at the higher levels as well as specific tests at its own level. The various stages of an MBE Process are shown in Fig. 2. A comparison with Fig. 1, the equivalent diagram for the MBD process, shows the difference in verification strategies.

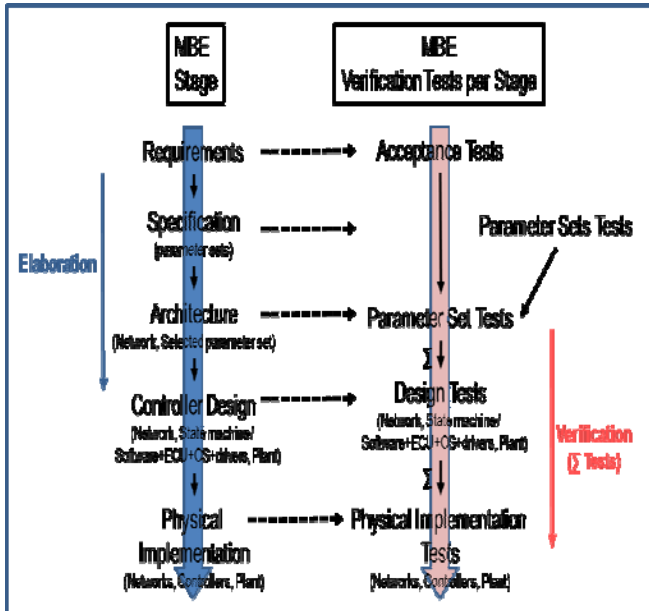


Fig. 2. MBE Process with its Integrated Elaboration & Verification

The implementation of MBD and MBE has been improved with the advent of systems integration tools such as Autonomie [6], which automates the assembly of multiple models into monolithic and/or distributed system models for analysis via simulation (Fig. 3). In MBD, the progressive availability of concurrent simulation products such as ESSE Systems Engineering Workbench [7], Cosimate [8], FMI [9] and MATLAB [10] (e.g.: via S-Function) has enabled selected models to be extracted from a parent system model and simulated in parallel with its parent.

The combination of these tools and models, together with the increase in computation capability delivered by powerful computers, have enabled engineers to increase the fidelity of plant models and implement more complex controls, with the

side-effect of increasing the size and complexity of control models.

In this paper, we describe how engineers can:

- insert accurate network models into monolithic Simulink models to ensure accuracy and better model physical systems,
- disjoint monolithic models and recombine them as distributed system models
- better control the distributed simulation computing load, leveraging computer resources such as multiple cores to reduce the simulation time.

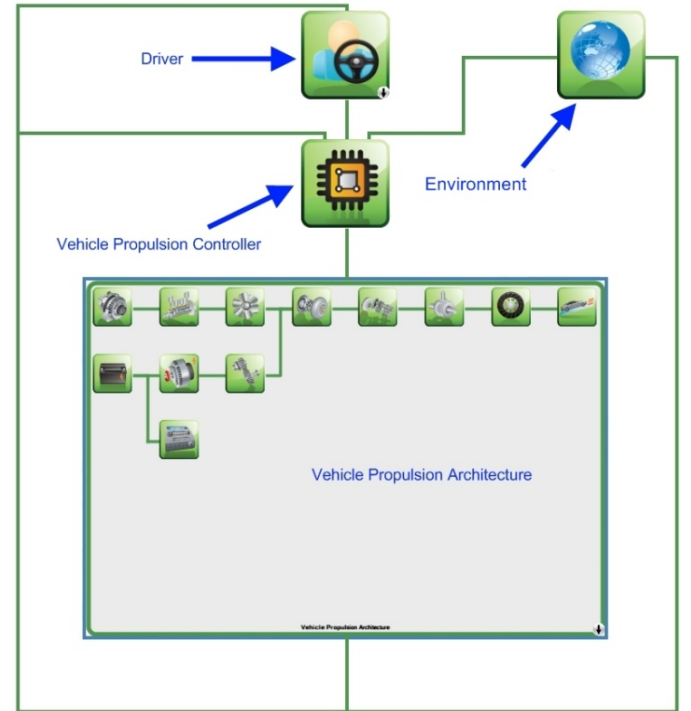


Fig. 3. Top-level vehicle layout.

## II. SYSTEM INTEGRATION TOPOLOGIES

Enabling models to communicate accurately with each other is the first part of their integration into complex system simulation. Most systems integration tools combine all the models under a single environment and simulate the whole system with a dedicated solver. In that topology, the models are connected via networks carrying signal or physical information (such as, signals or model connections in Simulink, messages in CAN). These networks can be organized to represent different hierarchies of communication. This ensure signals are only carried where they are needed and reduce the computational overhead (see Fig. 4).

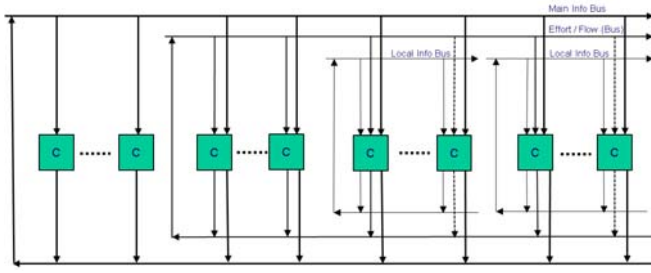


Fig. 4. Default system topology. Component models are connected together via hierarchical networks in a signal environment

However, as engineers integrate more compute-intensive models, that topology starts breaking down in term of performances. On one side, the solver has to run with the highest sampling rate required by the slowest model, leading to a lot more computation than needed for the faster models. On the other side, running all those models in the same solver thread means the calculation-intensive models also have to share the computing time of that thread with the faster models, slowing the whole simulation down.

A first step toward a solution to this performance issue is to start by separating the component models within the system architecture into sampling domains (Fig. 5).

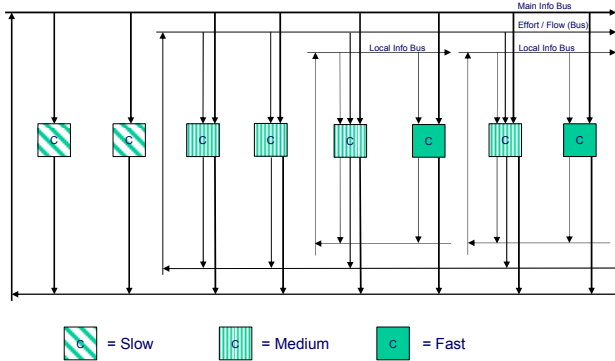


Fig. 5. Divide the models within the system into sampling domains.

Once those sampling domains are defined, the component models can be grouped by sampling requirement into modules which could be run in parallel and with their own solver properties to optimize the use of the computational power available on multi-core computers. Modules can also be organized according to a physical system architecture to preserve their domain representation (Fig. 6).

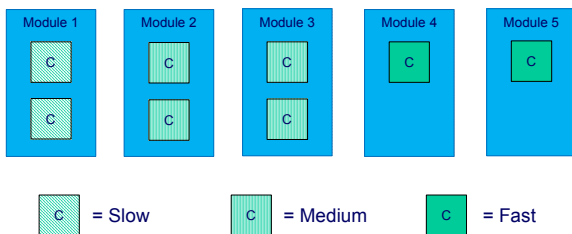


Fig. 6. Divide the models within the system into sampling domains.

Finally, the modules can be reconnected via timing accurate networks with different sampling rates so the modules themselves can be run with solvers at the optimum sampling rate to execute the models as quickly as possible without losing accuracy (Fig. 7).

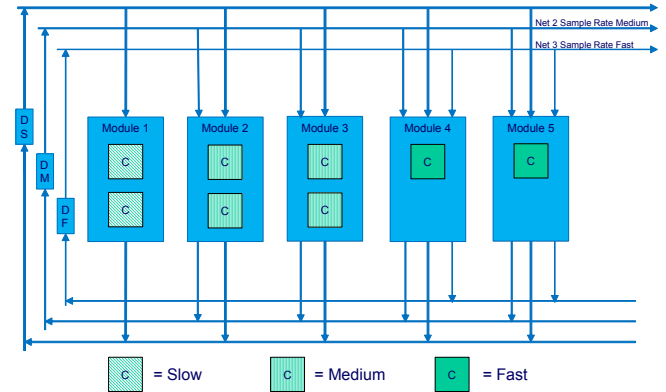


Fig. 7. Connect modules via buses (networks) with different sampling rates

### III. LINKING SYSTEM INTEGRATION WITH ACCURATE NETWORK MODELING AND FAST DISTRIBUTED SIMULATION: BEST OF THREE WORLDS

The implementation of an improved system topology as described above, together with appropriate choice of solvers, can be successful if the four sets of constraints described below can be met.

**First**, the network models connecting all the modules have the following attributes:

- Accuracy in timing, function and structure
- High simulation performance

**Second**, modules need the ability to use either:

- Variable rate solvers, or
- Fixed rate solvers

A significant advantage of using variable rate solvers in distributed simulations is that variable step solvers may improve both simulation speed and accuracy non-linearly. General Motors has demonstrated this advantage with a monolithic model that executed in 12 seconds with a fixed step solver. This model was disjointed and recomposed as a distributed two module system. Simulating with the same fixed rate solver achieved about a 6 second simulation execution. With a well-chosen variable rate solver, the simulation performance under the ESSE Systems Engineering Workbench executed in 2 seconds.

**Third**, a model communication capability which enables:

- Parallel communication networks that transfer information between modules at different sampling rate.
- Parallel execution of the network-connected modules at different sampling rate, and, if possible, on separate cores.

**Fourth**, this process will be highly effective when engineers have the flexibility to easily:

- Define or modify the system topology as wanted, i.e.: how many modules to run in parallel and which models to group together into modules
- Specify the different properties of the modules and networks (connections) used to communicate between them (for instance, sampling rate, name...)
- Define and validate the properties of all connections between the individual models and modules containing them
- Launch a process to automatically create all of the modules - with the models and connections that each contains – as well as all the necessary files to hand control to the ESSE Systems Engineering Workbench (see descriptions of the ESSE system below) for simulation.

When General Motors evaluated these requirements, they determined that there would be a highly desirable outcome in combining the systems integration capabilities of Autonomie, with the distributed communication and simulation capabilities of EST's ESSE Systems Engineering Workbench.

The ESSE Systems Engineering Workbench is built around an MBE process and features completely distributed scheduling which endows it with the potential to achieve maximum parallel simulation performance. Accurate modeling of the function, timing and structure of networks connecting models is a fundamental safety requirement of MBE and the ESSE Systems Engineering Workbench is unique in providing this capability.

EST's core technologies which satisfy the first three constraints, include:

- The ESSE Distributed Simulation Engine (DSE) that supports a scalable, massively parallel, distributed simulation capability that can concurrently execute many models and manage the interconnection and accurate message communication between modules.
- The ESSE System Network Fabric (SNF) technology from which all of its high performance, accurate network models are constructed. The types of SNF networks include CAN, CAN-MM, Signals, SyncSignals, FlexRay, UART, WiFi, DSRC, etc.
- The ESSE Simulation Desktop that enables user control of distributed simulations. Commands include: Start – starts the simulation of all models, Stop – stop the simulation and exits safely from all simulating processes, Pause – pause all models at the same simulation time, Resume – resume simulation of all models.

For the system integration part, Autonomie is designed as a framework which can accommodate any hierarchical system architecture and connection topology. These properties are defined in an abstracted vehicle file which enables Autonomie to build systems on demand by assembling the models and

their connections as defined by the user. This flexibility also enables the tool to take into account any topology reconfiguration defined for a simulation process, whether it is automated (e.g.: add ToWorkspace blocks on model outputs to log the signals), or user defined (e.g.: split models into multiple modules). Autonomie automatically handles the model and data compatibilities, as well as model to model connectivity to ensure all the initializations and connections are well defined.

In this project, a new process dedicated to launch the ESSE Systems Engineering Workbench was added, and its associated editor (Fig. 8) was integrated into the Autonomie Graphical User Interface, allowing the engineers to easily define:

- How many modules they want to run in parallel, as well as their properties, e.g.: solver type and sampling rate of each module.
- How many ESSE SNF networks should be available to interconnect these modules, as well as their type, sampling rates, communication rate and connections to modules.
- How to group the models specified in the selected vehicle into different modules. The user can see the original vehicle topology in a tree on the left hand side of Fig. 8, and just pick a model and drag and drop it into the module they choose.

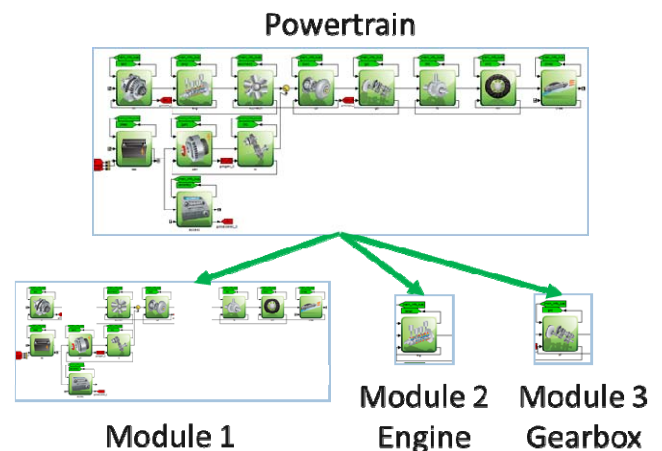


Fig. 8. Autonomie editor enables the user to easily define the module topology and the connection properties.

In the background, Autonomie automatically computes which communication network is best suited to ensure a synchronized connection between models in different modules, based on the modules solver properties and which networks are available to those modules. That information is then displayed to the user for review.

Once the model distribution topology and the network properties are defined, the user can launch the whole ESSE Simulation process in Autonomie by clicking on a single "Run" hyperlink.



By default, Autonomie builds the system (vehicle) model on demand by automatically adding and connecting all its selected sub-models. In this project, thanks to the customizable Autonomie framework, we easily integrated a different build process to add and connect models in separate modules. Special ESSE communication I/O interface blocks are also automatically inserted into the Simulink modules to enable the connection of each module to its appropriate instance and type of SNF network model. Autonomie then creates all the XML and batch files required to setup and launch those modules via the ESSE Systems Engineering Workbench.

Autonomie then launches the ESSE Distributed System Manager (DSM) which reads the XML and batch files to instantiate each process created from each Simulink module with its appropriate core affinity. The ESSE DSM is also responsible for connecting all networks to each module based on the information in the XML files generated by Autonomie. The ESSE System then launches the ESSE Distributed Simulation Engine and passes user control to the ESSE Simulation Desktop.

#### IV. EXAMPLE: MONOLITHIC MODEL PARTITIONED AND RECOMPOSED AS A 6-MODULE DISTRIBUTED SYSTEM

The example model has the structure of an automotive subsystem. But it has been developed to test various capabilities of Autonomie in model composition (Figure 9), and model partitioning and recomposition into a distributed system of 6 independent subsystems connected by ESSE SNF networks, both Signals and CAN (Figure 10).

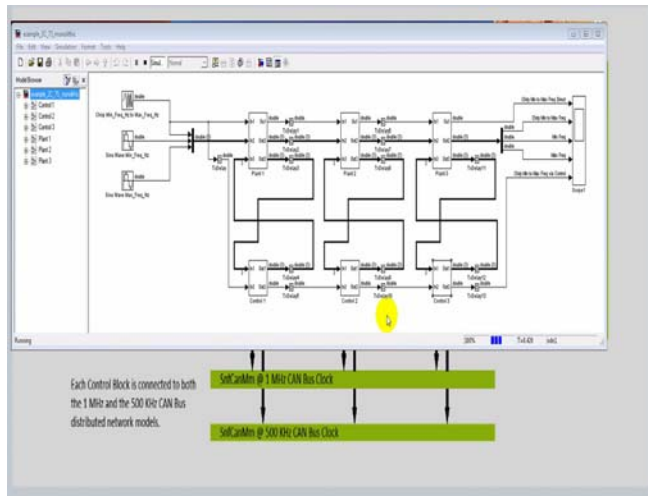


Fig. 9. Autonomie constructed Monolithic model with internal ESSE SNF SyncSignals network connections and connections to 2 ESSE SNF CAN Networks

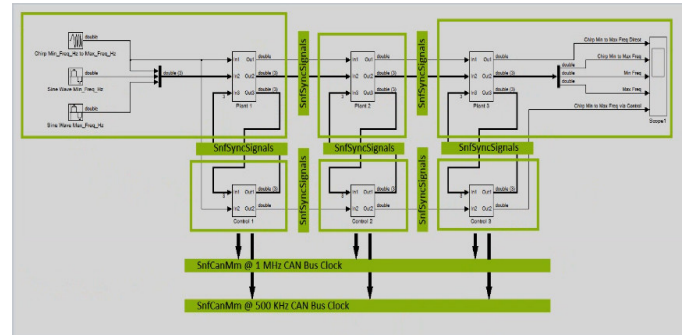


Fig. 10. Autonomie constructed Disjointed and Recomposed Distributed System with 6 subsystems connected by ESSE SNF SyncSignals networks and 2 ESSE SNF CAN networks

Each of the models was simulated with the ESSE Systems Engineering Workbench. Figure 11 shows the results of input signals being passed into and out of the Figure 9 model in the left-hand panel, and, in the right-hand panel, results of input signals being passed into and out of the Figure 10 model. The two panels show very similar signal wave patterns. And detailed point by point examination demonstrates that the two sets of waveforms are identical. In other words, both systems perform identically in regard to function and communication fidelity and timing accuracy under these inputs.

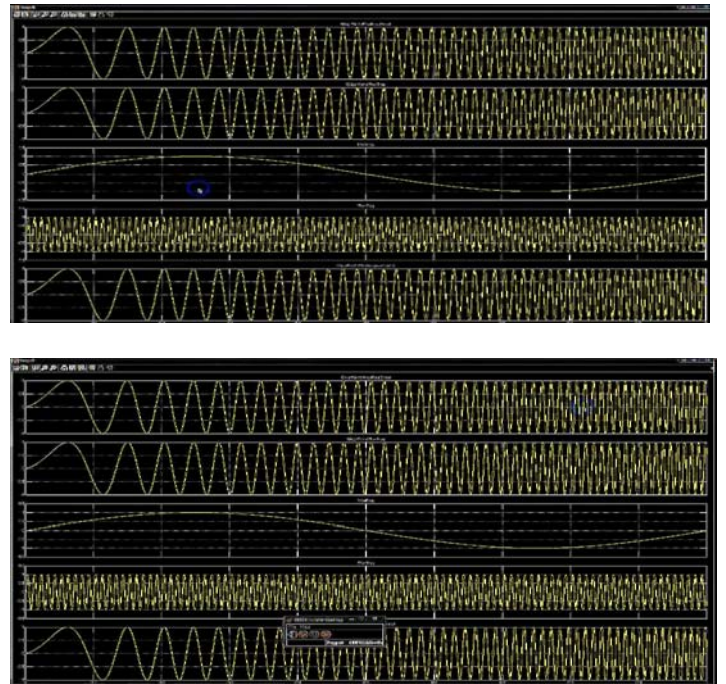


Fig. 11. Demonstration that the original monolithic 6 module system of Figure 9 (top plot) and the Disjointed and Recomposed Distributed System of 6 subsystems of Figure 10 (bottom plot) – each connected by ESSE SNF Sync Signals networks – behave identically. The wave forms are (from the top): Chirp – path thru plant, Chirp – path via plant & control, Sine wave – low freq., Sine wave – high freq., Chirp – path thru control. The object in the bottom middle of the bottom plot is the ESSE Simulation Desktop used to control multi-process, multi-core simulations.

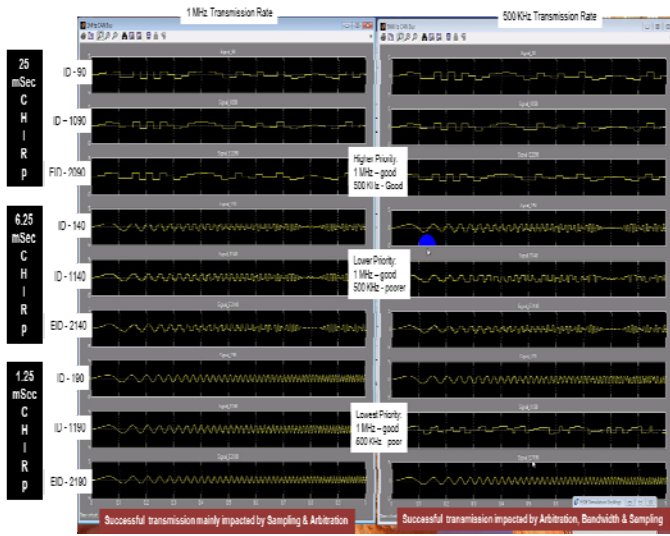


Fig. 12. Demonstration of the correct behaviour – in function and timing – of a 1 MHz CAN network (left) and 500 KHz CAN network (right) operating in each of the models of Figure 9 and Figure 10. Since the CAN networks of the Monolithic and Distributed System models produced identical behaviours only one set of traces is published in this paper.

The trace of CAN message transmission of various priority are shown in the left-hand panel of Figure 12 for the 1 MHz CAN network and the right-hand panel for the 500 KHz CAN network. Since the CAN network behaviours were identical for both the monolithic and distributed systems, only one set of results is presented in this paper. An interesting feature of the modelled CAN network is that it has the accuracy of corresponding physical CAN network.

The maximum performance of the ESSE SNF SyncSignals network is approximately 2 million messages per second. And that of the ESSE CAN network is similar.

The simulation performances of the monolithic and distributed system models were compared using wall-clock timing. The distributed system model was simulated using 7 cores of an Intel x86 i7 (4 physical cores each with 2 threads, each core-thread has the capability of about 0.6 of a physical core). The monolithic model was simulated using 1 core of the same computer. The distributed system, with each of the 6 modules presenting similar computational demand, simulated ~3.2x faster than the monolithic system using the same computational demand per module.

## V. REAL WORLD PERFORMANCES

The improved topology and distributed simulation mentioned in this paper has been tested and evaluated at General Motors using production level models, and combining systems of either Simulink-only models or a combination of Simulink and Amesim models. The combination of Autonomie tool and the tools and models of the ESSE Systems Engineering Workbench allowed a quick and easy way to partition component models in modules and redefine their connection topology as needed to improve the simulation time while maintaining communication accuracy (TABLE I.).

TABLE I. SIMULATION IMPROVEMENTS

Simulation types	Simulation time		
	Decomposition	Original time	New Time
Vehicle 1, Simulink Only	GM System Model	16 times real time	1.07 times real time

## VI. DISCUSSION

In using the integrated ESSE Systems Engineering Workbench integrated with the Autonomie System integration capabilities on GM models, the GM simulation speed target was to achieve 1:1 ratio (simulation time vs real-time). GM engineers were able to try different system partition strategies to get a ‘good’ distributed simulation model in short time. For example, a monolithic model that executed 16 times slower than real-time, was subject to experimental partitioning and recomposition. There were several hundred ways to partition and recompose this model. One of the authors (Sun), obtained a model that reduced the execution time from 16 to only 1.07 time slower than real-time in less than 3 hours. This is a highly significant achievement and demonstrates the productivity of the ESSE-Autonomie integration in an industrial situation.

Physical vehicle plant and control systems are distributed systems by nature. They are typically represented as monolithic models due to concern for the accuracy of continuous plant models during simulation. One can argue a distributed continuous plant model will not compute exactly the same response as its corresponding monolithic model. This argument is likely true. However, the objective of modeling and simulation of physical systems is to use a math model to produce an acceptable system response to a prescribed set of inputs, based on some accuracy criteria. Both monolithic and distributed system models can be constructed using these constraints and both may have behaviours that approximate to the physical system within the same error tolerance. If so, both are equally fit for use. Arguing about the quality difference between two models having the same accuracy tolerance is meritless; arguing about the performance of two such models has value. Both arguments are decidable empirically and dispassionately.

An interesting note. When monolithic models support signal interconnections between their internal modules to have physically appropriate latencies – rather than zero latencies as is typical in continuous domain models. And those modules are disjointed from their parent model and recomposed with the original interconnections replaced by appropriate ESSE SNF Signals networks, having, respectively, the same *physical* latencies. Then both the monolithic and distributed models will behave identically during simulation, except that the distributed model will likely execute considerably faster than the original monolithic model.

The combination of distributed system models with *physical* accuracy in function, communication and timing, and high performance in simulation, makes a strong claim for them to be standard reference models.

## VII. CONCLUSION

With the increase in standard computing power and the advent of Model Based Design (MBD) and Model Based Engineering (MBE), system simulations have become a lot more complex and now often require the combinations of heterogeneous and multi-domain models. This level of model interconnection has been improved with system integration tools which automate the compatibility checking as well as the setup and assembly of all the models into monolithic or distributed systems. However, this level of complexity and integration often leads to sub-optimum usage of the computing resources available or sub-optimum connectivity between the systems. In our case, we had a simulation example running 16 times slower than real time.

After analysing the vehicle system simulation, it became clear that a better distribution and control of the models over the computer resources was needed. Reorganizing the system topology into distributed systems would enable a nearer optimum selection of independent solver options and a better spread of the computing loads. This would also require a way to accurately network all the model connections.

In this paper, we demonstrated how we successfully designed such a framework by combining the system integration features of Autonomie with the distributed simulation engine and network features of the ESSE Systems Engineering Workbench.

Autonomie enables us to quickly redefine the computing load topology of a system simulation by graphically designing the model distributions in modules, each of which are run in parallel and can be setup with independent solver options. Once that distribution is defined, Autonomie automatically builds each module and launches the ESSE Workbench which execute them concurrently to better leverage the computer resources, and ensure accurate connections between each modules in timing, structure and functions.

This framework enabled us to speed up our system simulation by a factor of almost 16, close to real time, and with

the same behaviour as the monolithic model with physical latencies.

## CONTACT INFORMATION

For Autonomie, contact Aymeric Rousseau: [arousseau@anl.gov](mailto:arousseau@anl.gov), +1-630-252-7261.

For Embedded Systems Technology, contact Graham Hellestrand: [g.hellestrand@esetek.com](mailto:g.hellestrand@esetek.com), +1-650-488-4571.

## REFERENCES

- [1] Smith, P., Prabhu, S., and Friedman, J., "Best Practices for Establishing a Model-Based Design Culture", SAE Technical Paper 2007-01-0777, 2007, doi:10.4271/2007-01-0777.
- [2] Post, D. "Product Development with Virtual Prototypes", Computing Edge, 22-25, IEEE Computer Society, March 2015.
- [3] Winters, F.J., Mielenz, C., and Hellestrand, G.R., "Design Process Changes Enabling Rapid Development", Proc. 30<sup>th</sup> Convergence, 2004-21-0085, Oct. 2004, 613-624.
- [4] Hellestrand, G.R., "Engineering Safe Autonomous Mobile Systems of Systems: Using Specification (Model) based Systems Architecture and Engineering", IEEE 7<sup>th</sup> International Systems Conference, Orlando, Florida, April 2013, 599-605.
- [5] Abdallah, A., Feron, E.M., Hellestrand, G.R., Koopmen, P., and Wolf, M. "Hardware/Software Codesign of Aerospace and Automotive Systems", Proc. IEEE, Vol. 98, No. 4, April 2010, 584-602.
- [6] Argonne National Laboratory, Autonomie (Version 14.0), Computer Software, Argonne, IL, [www.autonomie.net](http://www.autonomie.net), 2014.
- [7] Embedded Systems Technology, ESSE Systems Engineering Workbench, [www.esetek.com/automotive-systems.html](http://www.esetek.com/automotive-systems.html).
- [8] ChiasTek, CosiMate (Version 2013.07), Computer Software, Toulouse, France, 2013, <http://www.cosimate.com>
- [9] MODELISAR, Functional Mock-up Interface, [www.fmi-standard.org/](http://www.fmi-standard.org/).
- [10] MathWorks, MATLAB, Computer Software, Configuring Models for Targets with Multicore Processors, [www.mathworks.com/help/simulink/ug/\\_bs1315v.html..](http://www.mathworks.com/help/simulink/ug/_bs1315v.html..)