2012 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
SYSTEMS ENGINEERING AND INTEGRATION (SE) MINI-SYMPOSIUM
AUGUST 14-16, MICHIGAN

Challenges Faced By System Integrators: From a System Simulation Approach

Karthikeyan Radhakrishnan
Ramesh Padmanaban
Ravindra Paike
Hari Vijay
LMS International
Troy, MI

ABSTRACT

In today's competitive market, OEMs are racing towards developing more efficient vehicles without sacrificing on its performance. In this process, they're forced to evaluate new technologies and designs in various subsystems. Most of the sub-systems today have become "intelligent", which means that the controllers have become quintessential for the system's behavior. Equally important are the physical behavior of the plant that needs to be controlled. These two independent groups have their own design and development cycle and the challenge for the companies have been in bridging the gap so as to identify potential failure modes. This paper discusses an Architecture-driven Model Based Development process that can address the challenges posed during the development. Three key enabling technologies – Imagine.Lab System Synthesis, Imagine.Lab SysDM & Imagine.Lab AMESim are leveraged in this process.

INTRODUCTION

Embedded systems in commercial and military vehicles are becoming increasingly complex in the functionality they support. Safety and security are very critical. Innovative approaches are needed to develop such systems efficiently without compromising on quality. A growing trend in development of complex embedded systems is the use of model-based development (MBD) techniques. MBD involves capturing the behavior of physical system as mathematical models that can be used for analysis, optimization, and verification and validation against desired functionality. MBD has proven effective in reducing development time and increasing product quality & reliability. MBD supported by CAE tools facilitates early V&V before the mechanical and electronic hardware become available.

A growing trend in model-based development of systems is that development has moved from being practiced by single or small group to large and globally distributed groups working in production environment. The processes, tools and work products continuously evolve during the development. Integration of different aspects/parts and re-usability of a work product becomes a challenge. MBD is focused on authoring, analyzing and simulating behavioral models. While this is essential for development, they prove incapable of addressing larger needs such as model & data sharing between plant and controller development cycles, variation management etc.

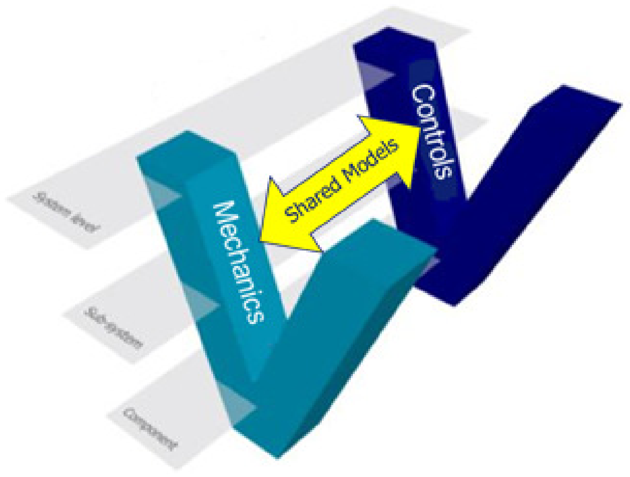


Figure 1: Concurrent Development Cycles

This leads to the idea of Architecture Driven Development (ADD), underpinning description of a system, with variant management built around this architecture and use of an effective tools/platform for model & data sharing, role based access control, intuitive retrieval, version tracking, etc. from a system simulation perspective.

In this paper we discuss the challenges faced during the development of large systems. Section 2 describes some of the challenges faced in the areas of model management, data management, knowledge management, variant management and system integration. Section 3 describes processes and tool chain that can help in addressing the challenges described in section 2. Section 4 describes a case study and finally concludes in section 5.

CHALLENGES

Following are some of the challenges in the development process,

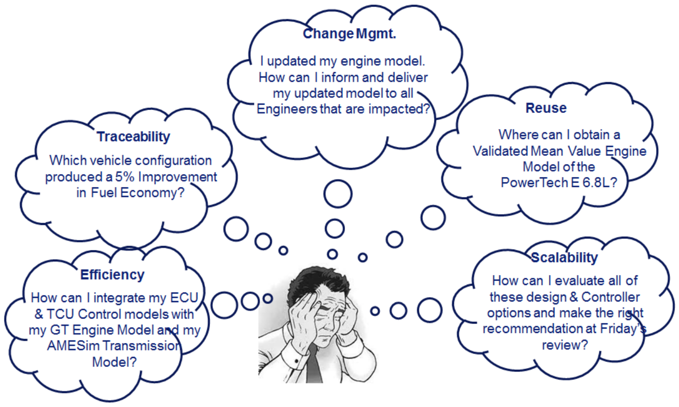

Change Mgmt.
I updated my engine model. How can I inform and deliver my updated model to all Engineers that are impacted?
Traceability
Which vehicle configuration produced a 5% Improvement in Fuel Economy?
Reuse
Where can I obtain a Validated Mean Value Engine Model of the PowerTech E 6.8L?
Efficiency
How can I integrate my ECU & TCU Control models with my GT Engine Model and my AMESim Transmission Model?
Scalability
How can I evaluate all of these design & Controller options and make the right recommendation at Friday's review?

Figure 2: Challenges

1. Change Management – Many of the modern systems are large and involve big and globally distributed teams. The final product keeps changing throughout the life cycle. While we know MBD is necessary ingredient in any development, managing the change in work product is a big challenge. The change can be caused due to multiple reasons. Once the changes are made then it is important to convey the changes to the appropriate set of users at the right time. Another dimension of the change management is keeping track of intricate dependencies between model and data files.

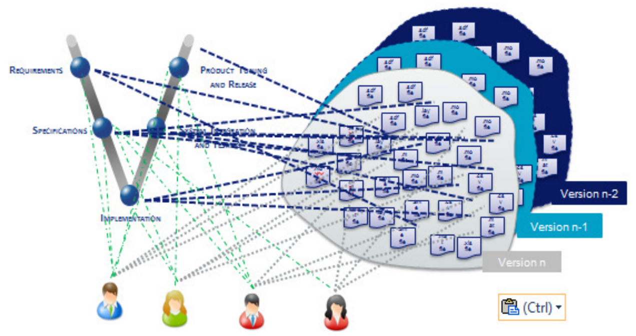


Figure 3: Change Management Challenge

2. Traceability and Re-use – As the complexity of engineered system grows thus grows the involvement of large, global and distributed teams of engineers. Another dimension of the traceability is keeping track of dependencies between model and data files. Distributed development makes it really challenging to trace and re-use core artifacts across various vehicle programs.

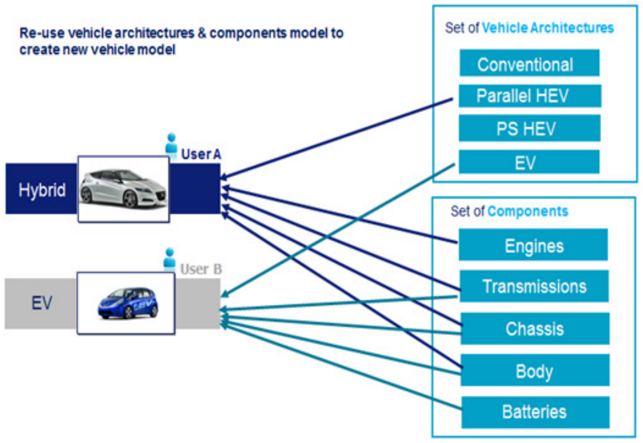


Figure 4: Traceability Challenge

3. Scalability and Share – Traditionally control system development used to happen after physical system development but to enable early Verification & Validation (V&V) in development phase and reduce the development time, most of the OEM's are frontloading the controls development through MBD. In recent times physical system and control system are developed concurrently in order to speed up the development time. In this case challenge for the company is to develop scalable models and to share these models across team. Early V&V leads in identifying design issues earlier in the development phase and increase the design confidence.

4. Openness and Efficiency – To facilitate the testing of the system at different stages in the V cycle models needs to work with different tools. A significant challenge then it becomes in managing the models from various domain specialized tools across the enterprise.

5. Time and Resource intensive System Integration – In Industry, due to complexity of the large system and to reduce the development time, systems are often decomposed into logical units whose development is then distributed across multiple teams and integrated at a later stage. For example, in the automotive industry, the engine system development is distributed in terms of some actuators such as the Electronic Throttle, Fueling System, the torque production system etc. Specifications for the smaller components should be well developed otherwise this will lead to system integrator spending large amount of time bothering with low-level implementation details instead focusing on system behavior. System integration problems could be very expensive to resolve at later stage in the development.

Next section describes by adopting Architecture Driven Development (ADD) methodology/process challenges of,

a. Seamless interface between plant and controls group can be addressed.
b. Time and Resource intensive System Integration can be addressed.

ARCHITECTURE DRIVEN DEVELOPMENT (ADD) APPROACH

System architecture is essentially the classification of the system defined by the hierarchy of subsystems, the interfaces and the connectivity between subsystems. Architecture may be elaborated with many additional properties, such as functional behavior, associated requirements, data, documents, etc. which can be used at different stages in the development lifecycle for different purposes. However, the essence of the architecture will provide the base information that enables consistency through the development lifecycle, and addresses the challenges described in section 2 above.

The ADD approach usually consists of two major activities. *First* the system architecture needs to be defined and functional models (Executable Specifications) be developed that are consistent with the system architecture. *Second* the functional models should be integrated with the architecture in a plug and play mode for various types of simulations.

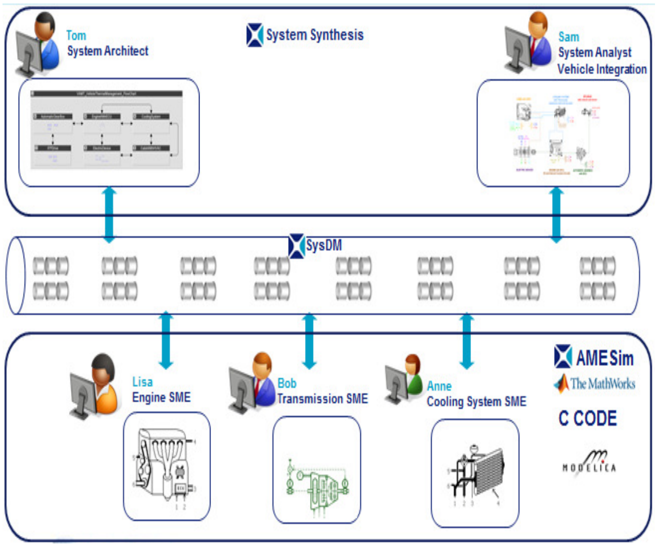


Figure 5: ADD Approach

The following sequence of steps is essential in implementing Architectural Driven Model Based Development:

1. Architecture Creation - The first key step is to develop the system architecture. The architecture is topology of the system describing the structural hierarchy of the subsystems/components, their interfaces and connections. Several stakeholders are involved in this step – System Architect, Domain specialist (Plant Modelers), Control engineers, Software engineers, System Analyst and Managers. Based on the end goal, of the type of simulations that would be performed by the System Analyst, a comprehensive set of interfaces (mechanical ports, thermal ports, control ports) are defined. This architecture can then be used to communicate the integration requirements to the teams who are responsible for individual components in the system.

2. Build Component Functional Model – Modular component model needs to be developed over a period of time to enable maximum re-use and have to be stored in a centralized repository which is accessible to all the engineers. For efficient reusability of those component models, the engineers will add standardized set of metadata attributes. The engineers should be able to perform the search on the repository based on these metadata attributes and system characteristics (Interfaces, Hierarchy etc.). To develop the

application model which will be consistent with system architecture, identify component models that match the architecture requirements such as interface, hierarchy, metadata criteria, etc.

3. System Integration – Final step is to generate the executable application model that can be simulated in a target native domain such as AMESim /Simulink. The simulation results can be used to confirm the functionality of the system.

The idea of ADD is to front load the system modeling activity by focusing on architecture construction with right set of interface definition to support various types of simulation downstream. The benefits of this approach are multiple,

a. Significantly reduce the System Integration time & effort.
b. Streamline the system development process to enable global distributed & concurrent development.
c. Modularize and have a few generic System Architectures from which multiple system simulation models can be generated and simulated. Any change in the architecture can be easily propagated to all the simulation models.

ENABLING TOOL CHAIN

The tool chain enabling Architecture Driven Model Based Development process is LMS Imagine Lab Platform, comprising of 3 solutions: IL System Synthesis, IL SysDM & IL AMESim.

Imagine.Lab System Synthesis

This solution is a "Tool-Neutral" Environment for simulation architecture and simulation configuration management. In this environment, a system analyst could

- Import System Architectures from AMESim, Simulink and SysML based Magic Draw.
- Populate the architecture with models or libraries from AMESim, Simulink, C and various native platforms etc.
- Create multiple System configurations to execute simulations in target platform of AMESim, Simulink or both.

Imagine.Lab SysDM

This solution is a Tool-Neutral collaborative framework for Model & Data Management to enhance System Simulation Process efficiency. Some of the key capabilities of this product are:

- Complete life cycle management of Model and Data
- Organize Model , Data, Architecture and other artifacts in a domain-relevant structure
- Role based access control to enable collaborative workflows
- Comprehensive Search & Retrieve capabilities to enable re-use of Models, Data, Architectures etc.

Imagine.Lab AMESim

This solution simplifies multi-domain integration thanks to its easy-to-use simulation platform. All an engineer needs to do is connect various validated components to simply and accurately predict multi-disciplinary system performance.

With extensive dedicated libraries, LMS Imagine.Lab AMESim actually saves enormous amounts of time by eliminating the need for extensive modeling. Thanks to application-specific simulation, engineers can assess a variety of subsystems in multiple physical domains. This way design and engineering teams can carefully balance product performance according to various brand-critical attributes to achieve the best possible design way before committing to expensive and time-consuming prototype testing. Since LMS Imagine.Lab AMESim actually frontloads system simulation early in the development cycle, it truly allows mission-critical design functionality to drive new product development.

It offers a complete 1D simulation suite to model and analyze multi-domain, intelligent systems and predict their multi-disciplinary performance. Model components are described using validated analytical models that represent the system's actual hydraulic, pneumatic, electric or mechanical behavior. To create a system simulation model, all the user has to do is use the various dedicated tools to access the required pre-defined components from validated libraries covering different physical domains. LMS Imagine.Lab AMESim can work with a variety of libraries to create a physics-based system model. Using libraries like the Hydraulic Component Design (HCD) and IFP-Engine, LMS Imagine.Lab AMESim software can accurately simulate intelligent system behavior long before detailed CAD geometry is available.

CASE STUDY

The objective of this case study is to showcase the application of Architecture Driven Model Based Development Process on a Hybrid Vehicle Energy Management.

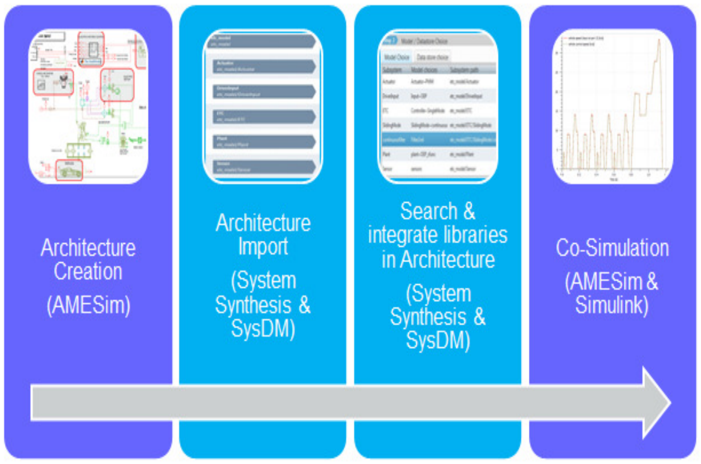

Architecture Creation
(AMESim)
Architecture Import
(System Synthesis & SysDM)
Search & integrate libraries in Architecture
(System Synthesis & SysDM)
Co-Simulation
(AMESim & Simulink)

Figure 6: ADD Process followed for Hybrid VEM

1. Architecture Creation: In this case study, the architecture was constructed in IL AMESim (shown in Figure 7).

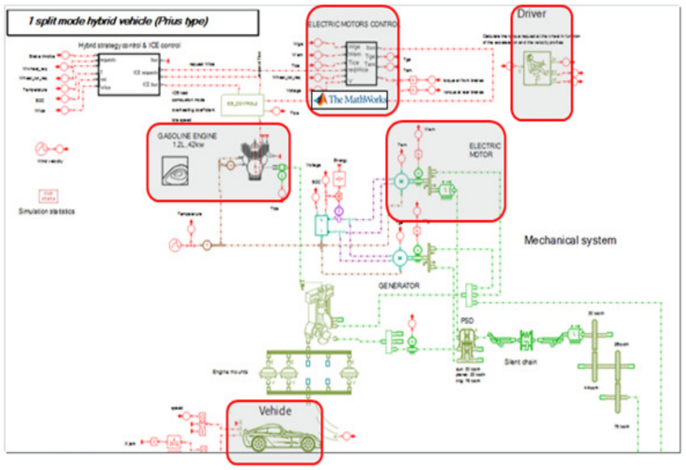


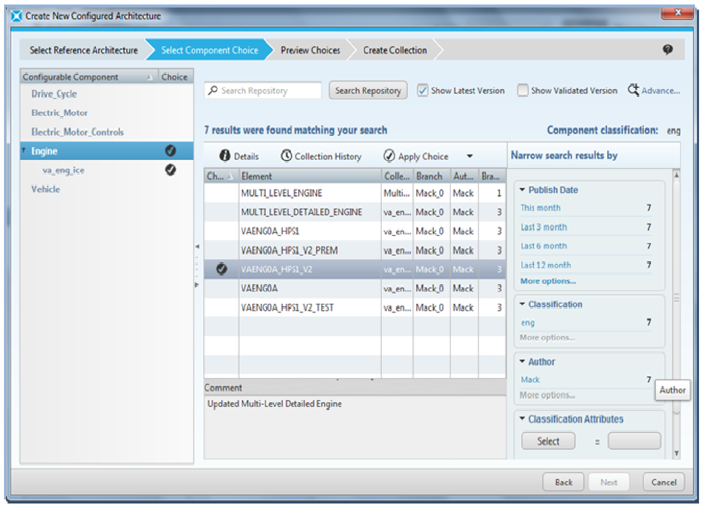
Figure 7: Architecture constructed in AMESim

During this step, the system architect interacts with subsystem experts (Engine plant, Electric Motors (plant & controls), hybrid Vehicle System Expert) to decide on the logical decomposition of the system, interfaces definitions (ports, port types, connections, parameters exchanged etc). In the sketch mode, the graphical view of the architecture was constructed and saved as an .ame file. At this stage, five blocks were marked as configurable components. These were Engine, Electric Motor, Electric Motor Controls, Driver and Vehicle.

2. Architecture Import: In this step the System Architect imported the AMESim based architecture (from previous step) into System Synthesis through a Configuration wizard. This wizard guides the user to select the AMESim Architecture file, Preview the configurable components and initialize the Reference Architecture. This Reference Architecture is then published into a central repository for sharing with other team members. During this publish process the user adds attributes (like Vehicle Platform name, Program Name, Life Cycle State, etc.). These attributes are meta-information associated with reference architecture, which helps during the Search & Retrieve process.

3. Search & Integrate Libraries in Architecture: In this step the System Analyst uses a process wizard for integration.
   a. The first step in this wizard is to search for the right set of validated Hybrid Vehicle Reference Architectures from central repository. At this point the user has a choice to use the attributes like Vehicle Platform name, program name, etc… to converge on the right architecture. The user has access to all the versions of the architecture and could pick the right one.

   b. At this point, the list of configurable components (Engine, Electric Motor, Electric Motor Controls, driver & vehicle) in the selected architecture (shown in Figure 8). For each of these configurable components, the user searches for available super-component libraries in the Central Server and selects a choice. For Electric Motor Controls a Simulink model was re-used from the server. The remaining configurable components were populated with AMESim super-component libraries. In the case of Engine component the user had a choice to configure the combustion model inside the engine. This is an example of multi-level configuration capability.



7 results were found matching your search

Component classification: eng

Ch...	Element	Colle...	Branch	Aut...	Bra...
	MULTI_LEVEL_ENGINE	Multi...	Mack_0	Mack	1
	MULTI_LEVEL_DETAILED_ENGINE	va_en...	Mack_0	Mack	3
	VAENG0A_HPS1	va_en...	Mack_0	Mack	3
	VAENG0A_HPS1_V2_PREM	va_en...	Mack_0	Mack	3
✓	VAENG0A_HPS1_V2	va_en...	Mack_0	Mack	3
	VAENG0A	va_en...	Mack_0	Mack	3
	VAENG0A_HPS1_V2_TEST	va_en...	Mack_0	Mack	3

Comment

Updated Multi-Level Detailed Engine

Figure 8: System Synthesis Wizard for search & assign Super-Component Libraries for each configurable component

c. After populating those five configurable components in the architecture, this set is saved as one vehicle configuration. The user creates multiple configurations by changing the choice in the above step. For example, if configuration A has a Mean Value Engine Model (Version N) as a choice the configuration B could be a detailed Engine model (Version M). The user could create various vehicle configurations and publish this information in the server & database.

4. Co-Simulation: In System Synthesis, the system analyst could select multiple vehicle configurations and "apply" the configuration. This step opens the vehicle configuration in the Native Tool (AMESim in this case) and all the choices are automatically set (for those 5 configurable components). The user could click on run to simulate. Another option is to create a Simulation Run Set for batch simulation of all the configurations. The user has the option to select post processing script at the end of batch runs. In this case the Electric Motor Controller was a Simulink model and hence a co-simulation was triggered.

SUMMARY/CONCLUSIONS

In this paper we discussed the challenges faced during System integration, from a system simulation point of view. Architecture Driven Development process is a key enabler in addressing most of the challenges. The enabling technology for this process is showcased as Imagine Lab Platform with IL System Synthesis, IL SysDM and IL AMESim as the 3 main products. The application of Architecture Driven Development with Imagine Lab platform is shown through Hybrid Vehicle Energy Management case study.

The value addition through Architecture Driven Development process and Imagine Lab platform offering is the following:

1. Architecture Driven Development methodology front loads System Simulation process with significant focus on Architecture Creation. The hierarchical breakdown of System / Sub-System / Components with rich set of interface definition enables distributed development.

2. Publishing the vehicle architecture with rich set of interface definition is a very convenient way to communicate the interface definitions to each sub-system & component developers. Each developer would know exactly what I/O needs to be present in their models / libraries.

3. The search & integrate process enables re-usability of core component & sub-system models / libraries across multiple vehicle programs.

4. Traceability is enabled through this process and the proposed tools. The System Analyst knows exactly which version of the Mean Value Engine Model was used for a specific Vehicle Program at different stages of the program.

5. The last step of System Integration is very efficient due to consistent interface definitions.