L&T Technology Services

MODEL BASED SYSTEM ENGINEERING (MBSE)

For accelerating software development cycle

Manish Patil | Sujith Annamaneni

Table of Contents

Abstract 03

MBSE Overview 03

MBSE Development Cycle 04

Migration of Legacy Code to MBSE 07

MBSE Frameworks 07

Benefits of MBSE Approach 10

Conclusion 10

References 11

ABSTRACT

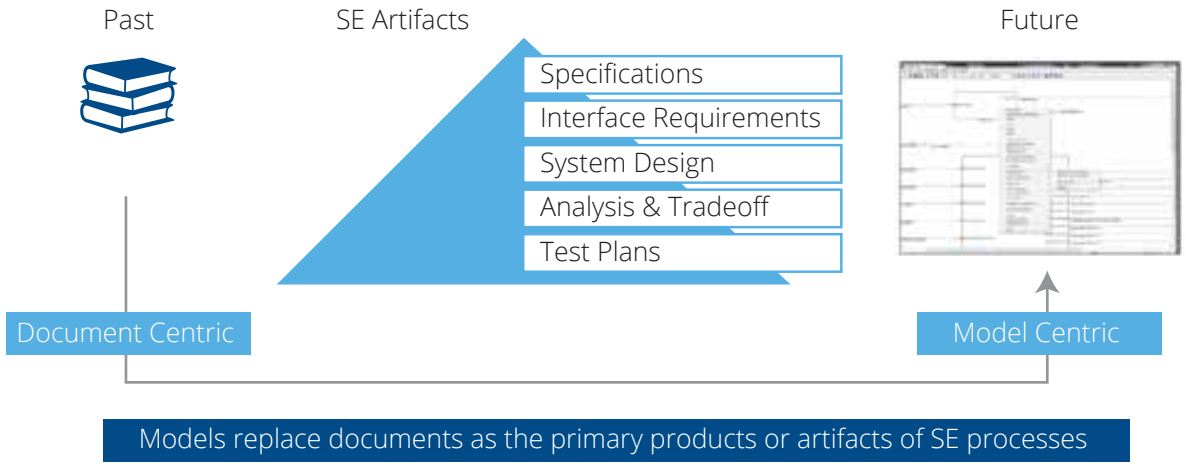
In today's vehicles, electronics account for about 40% of the total cost with a high emphasis placed on software and about 200 million lines of code which are developed in close collaboration with Service Providers, Tier 1s and OEMs. In today's automotive vehicle development programs, the development teams are placed globally, increasing the geographical diversity resulting in bigger challenges to maintain and ensure the consistent software co-ordination with different stakeholders. The main factors behind an increase in electronic content include increasingly stringent environmental regulations, advanced safety features, better drivability, increasing comfort and convenience, information and entertainment systems etc.

In the recent survey by Embedded Market Forecasters (EMF) on embedded product development it has been revealed that impediments in the software development process are mainly responsible for more than 80% of design delays and associated design complications. As the embedded systems getting more complex with increasing emphasis on software functionalities, it is becoming difficult to maintain quality, schedule and cost within budget with the traditional approaches of software development. In order to meet this challenge, automotive companies are moving towards a Model Based System Engineering (MBSE) approach.

This paper demonstrates the MBSE approach which plays a crucial role in the Software Development Life Cycle (SDLC) towards early verification and validation of the design resulting in reduced ambiguity and delays due to design iterations along with back-to-back testing. The benefits accrued with this approach are: reduction in software development time, effort taken and cost.

MBSE OVERVIEW

Model Based System Engineering (MBSE) is a paradigm shift in the software development methodology from traditional document-based specifications, design and manual coding with more towards the executable specifications in the form of models from which automatic code and test cases can be generated in single development environment. The following figure shows the transition approach to MBSE.



Transition to MBSE

Model-based design can be invaluable in providing early design verification and a true executable specification. MBSE approach can be adopted during the following scenarios:

- **Complex Functionalities:** An increasing functional complexity with continuous improvements in features results in more complexity, posing a challenge to maintain and enhance the functionalities without disturbing the existing working behavior.
- **Stricter Regulations:** With the advent of demand towards ensuring stricter regulations, safety constraints (ISO26262) and higher quality, MBSE approach supports achievement of these objectives by providing

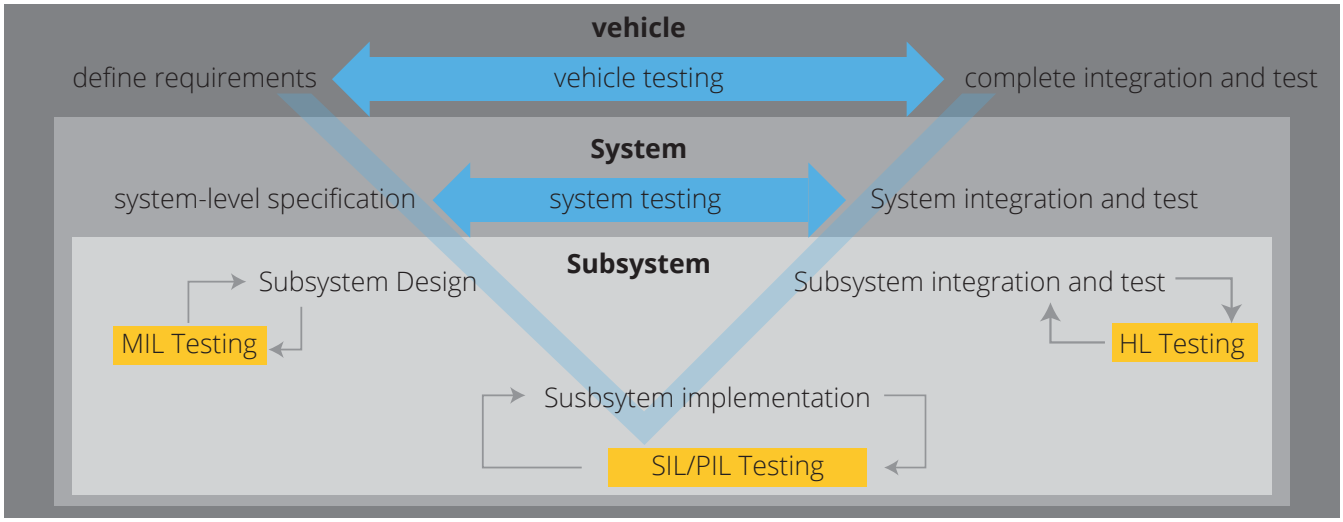
back-to-back testing and improving quality of software early in the development cycle.

- **Reduced Development Schedule and Cost:** Due to increased competition within the automotive OEMs, the time to market for new vehicles has reduced drastically, impacting on the overall product development life cycle with increasing impact to reduce the schedule and cost of vehicles.
- **Early Prototyping:** Expensive or impractical to build early prototypes or proof-of-concept development resulting in design cycles that are more predictable and give faster results

MBSE DEVELOPMENT CYCLE

MBSE approach provides a design environment that enables developers to use a model for data analysis, visualization, testing and validation. Once the model is developed and tested, using auto-code generation method, software for the production embedded design is automatically generated, thereby bringing uniformity in design and code, reducing code development time which will result in reducing overall costs as compared to the traditional development approach.

The MBSE SDLC is outlined in below



MBSE SDLC

The below figure represents a typical MBSE development cycle for capturing system level specifications (Plant Model), control algorithms (Control Model), auto-code generation and verification/validation methodologies.

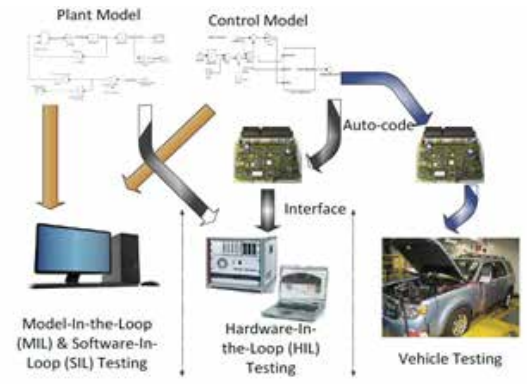


Figure 3: MBSE Development Cycle

The MBSE workflow includes the following steps:

- Create an executable specification consisting of algorithms, system model, and system level verification environment
- Verify the system model against functional requirements using simulation
- Algorithm and behavioral models are optimized and refined yielding a fully tested specification

- Perform floating point to fixed point conversion
- Incorporate AUTOSAR interface information within the model for AUTOSAR compliant auto-code generation
- Incorporate ISO26262 functional safety information within the model
- Automatically generate production software for embedded processors
- Verify the auto-code against model specifications

The below figure outlines the detailed model based development cycle with step-by-step approach.

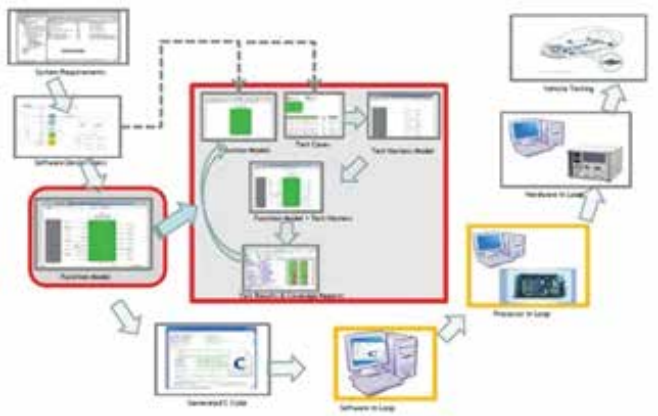


Figure 4: Step-by-Step MBSE Approach

The MBSE approach is being adopted for the SDLC support towards standards and guidelines-compliance during software development stages as outlined below:

- Standards and Guidelines Compliance

  In the Model based development, Model Advisor checks supports for the verification of the model for conditions and configuration settings of various standards as AUTOSAR, ISO26262, IEC61508, MAAB modeling guidelines and MISRA-C compliance. It produces a report that lists all the sub-optimal conditions and settings that is found and suggests better model configuration settings wherever appropriate.

- AUTOSAR Compliance

  Import and export of AUTOSAR Software Component descriptions and generation of AUTOSAR production code can be achieved in model based design. Model based design and auto-coding tools support AUTOSAR compliance using model configuration settings rather than AUTOSAR-specific blocks. As a result, a single Model can be used as a reference for simulation, rapid prototyping and production code generation in both AUTOSAR and non-AUTOSAR environments.

- ISO26262 Compliance

  The development of safety complaint systems within the automotive industry is characterized by demonstrating compliance with ISO26262, a standard for road vehicle functional safety. We can generate project-specific artifacts, including traceability matrices covering requirements, models, and generated code. Project and product-specific artifacts can be combined to produce needful documentation for achieving ISO 26262 certification.

- Verification and Validation

  a. In the Model Based Design approach, executable models can be tested and the test information can be re-used and applied later for testing code.

  b. Test cases can be automatically generated or easily adopted with the changes and coverage can be taken forward.

  c. Requirements-Based Testing, Decision, Condition, Modified Condition / Decision coverage can be covered with Model-In-the-Loop (MIL), Software-In-the-Loop (SIL), Processor-In-the-Loop (PIL) and Hardware-In-the-Loop (HIL) testing methods.

- Model Reusability

  Model based development approach supports the creation of library blocks which can be developed for commonly used functions and can be included in Simulink library toolbox for re-use into different modules or features. Modification of library block at one place will imply the changes in all the models using these blocks.

- Retention of Legacy Code

  An important feature of model-based design approach is its ability to reuse existing pre-tested and deployed code as a part of the model. In this way, existing applications can be updated or enhanced without having to model the complete design. The legacy code can be encapsulated in the model using S-function providing the appropriate interfaces for inputs and outputs mapping. Thus ensuring the unified development environment with integration of legacy code into model based environment for verification of the algorithm behavior.

- Requirement Traceability

  The model can be linked with requirements in DOORS, Word or Excel format and traceability report can be generated directly from the model.

- Continuous Integration and Testing

  The below figure explains the automated steps adopting MBSE with continuous integration and testing framework using Jenkins.

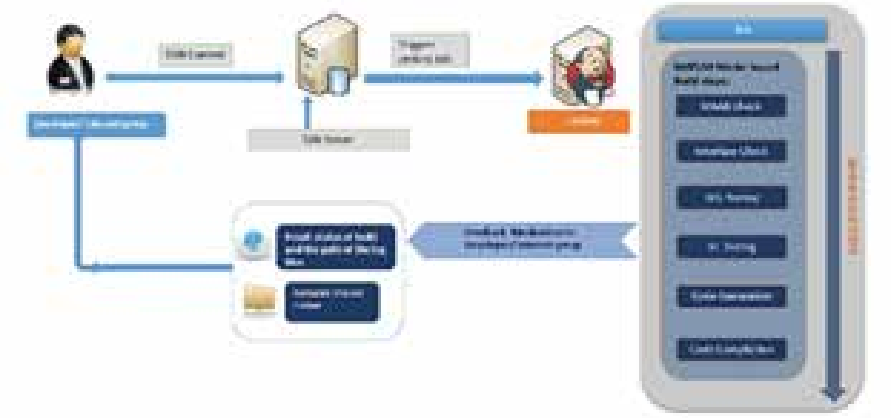


Figure 5: MBSE in Continuous Integration and Testing Framework

The continuous integration framework is adopted in the Model Based Design. As soon as the developer commits the changes to a subversion repository, continous integration framework process is initiated. Jenkins server detects the changes in the subversion repository and the pre-configured Jenkins job is triggered automatically. During the execution of Jenkins jobs, MBSE quality gates are verified for the following scenarios:

a. Modelling Guidelines Check: In this stage, the modelling guildelines are verified using Matlab for MAAB compliance and the modelling guidelines compliance report is generated.

b. Interface Check: In this stage, the inputs and outputs of the signals are compared in the Matlab model against the data dictionary and interface check report is generated.

c. Auto-coding Check: In this stage, the auto-code generation is invoked and verified for the successful code generation using defined configuration.

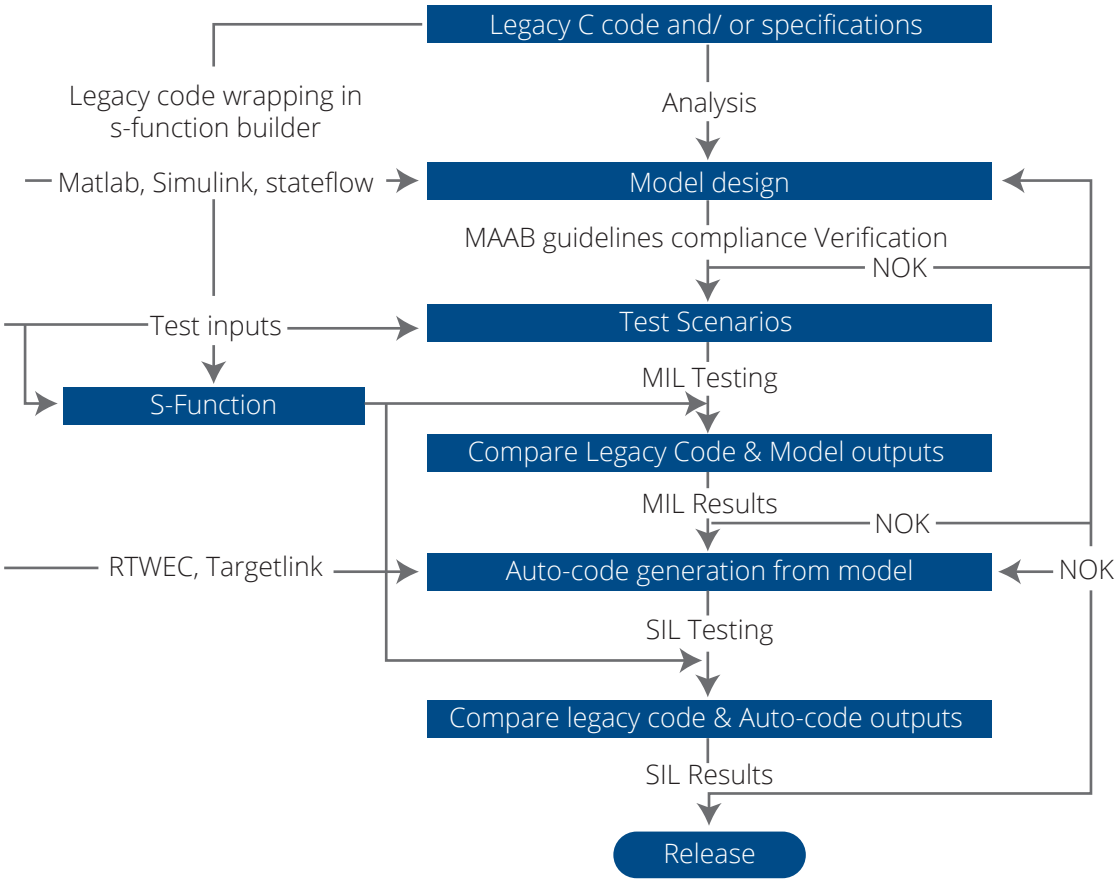
d. Compilation: In this stage, the auto-code compilation is carried out and checked for any errors.

e. Unit Testing: In this stage, the unit testing in MIL (Model-in-the-Loop) and SIL (Software-in-the-Loop) environments is carried out against the test cases. The tests are run in Matlab and Tessy environments and reports are generated for the coverage based on statement (C0), branch (C1) and MCDC configuration.

During the execution of the above stages, an email will be sent to the developer on whether passed or failed, along with a link of the execution status for each stage for a detailed report. For the errors reported, the developer performs an analysis and rework for the reported issues. Then the changes are re-committed to the working copy or subversion branch which triggers the Jenkins job and the process is repeated.

MIGRATION OF LEGACY CODE TO MBSE

The figure outlines the overall step-by-step approach for migration of legacy code to MBSE along with intermediate quality gates towards modelling guidelines, MIL and SIL testing methods.



Flowchart For Migrating Legacy Code to MBSE

MBSE FRAMEWORKS

We have extensively worked for the development of automation frameworks using Matlab scripting language for the auto-code generation and MIL / SIL testing.

i. Auto-code Generation Framework:

This framework incorporates the following aspects:

- Configuration settings for auto-code generation
- Customized library block checking
- Modelling Guidelines checker
- Code Generation utilities using Embedded Coder
- Interface checks between model and data dictionary to ensure correct usage of memory, boundary values and data types

The below figure outlines a typical Auto-code Generation Framework:

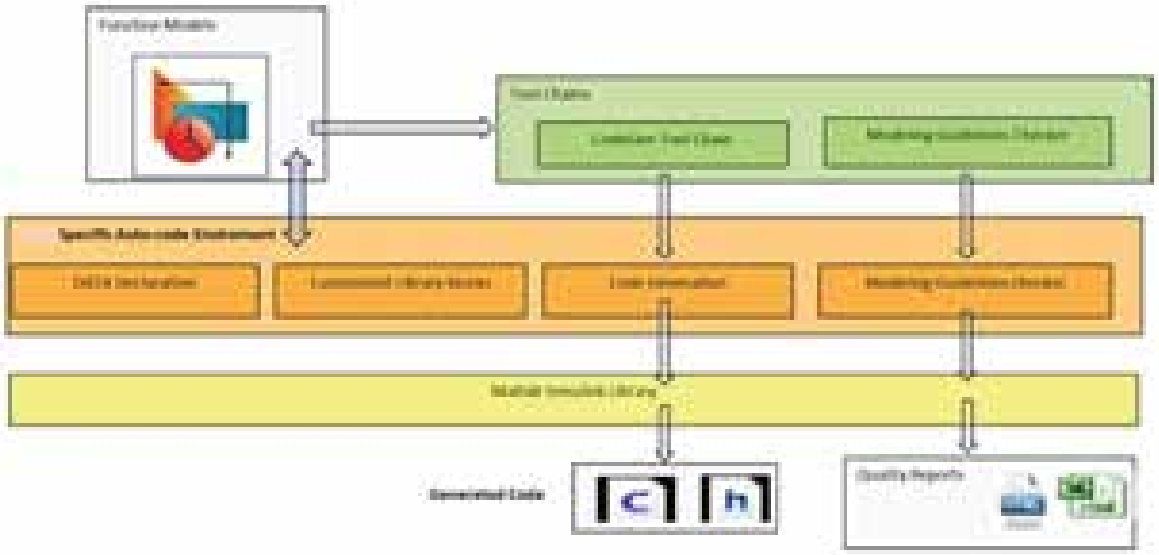


Figure 7: Auto-code Generation Framework

ii. Unit Testing Framework:

This framework incorporates the following aspects:

- Generation of automatic test vectors for boundary values and equivalence testing in reference with the data dictionary
- Creation of MIL and SIL test framework with incorporation of model and auto-code under test automatically
- Apply test scenarios automatically and validate the results of MIL and SIL tests
- Generation of model and code coverage reports

The below figure outlines the unit testing

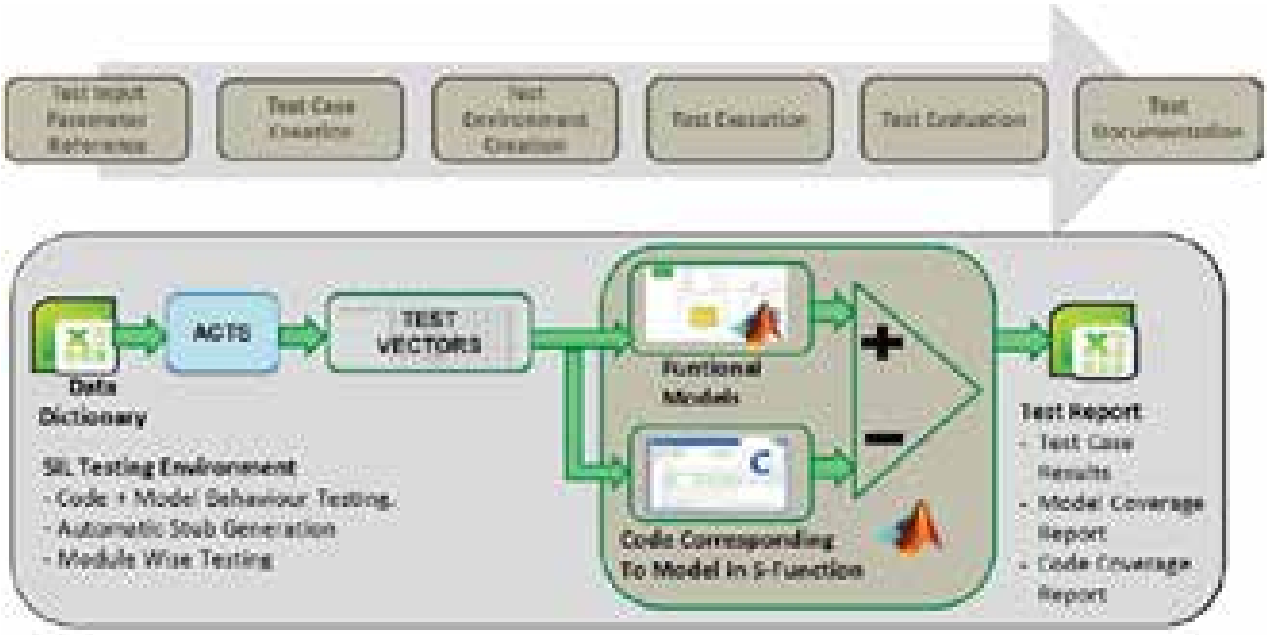


Figure 8: Unit Testing Framework

This framework supports:

- Detection and reporting of violations between model and code with respect to the interface requirements
- Identification and reporting of the precision errors between model and code
- Detection and report of incompatible behavior between model and code

iii. HIL Testing Environment:

Hardware-in-the-loop (HIL) simulation is a technique that is used in the development and test of complex real-time embedded systems. HIL simulation provides an effective platform by ensuring complexity of the plant is under the control of the test platform. The complexity of the plant under control is included in test and development by adding a mathematical representation of all related dynamic systems. These mathematical representations are referred to as the "plant simulation". The embedded system to be tested interacts with this plant simulation. The below figure outlines a typical HIL testing environment being incorporated in MBSE.

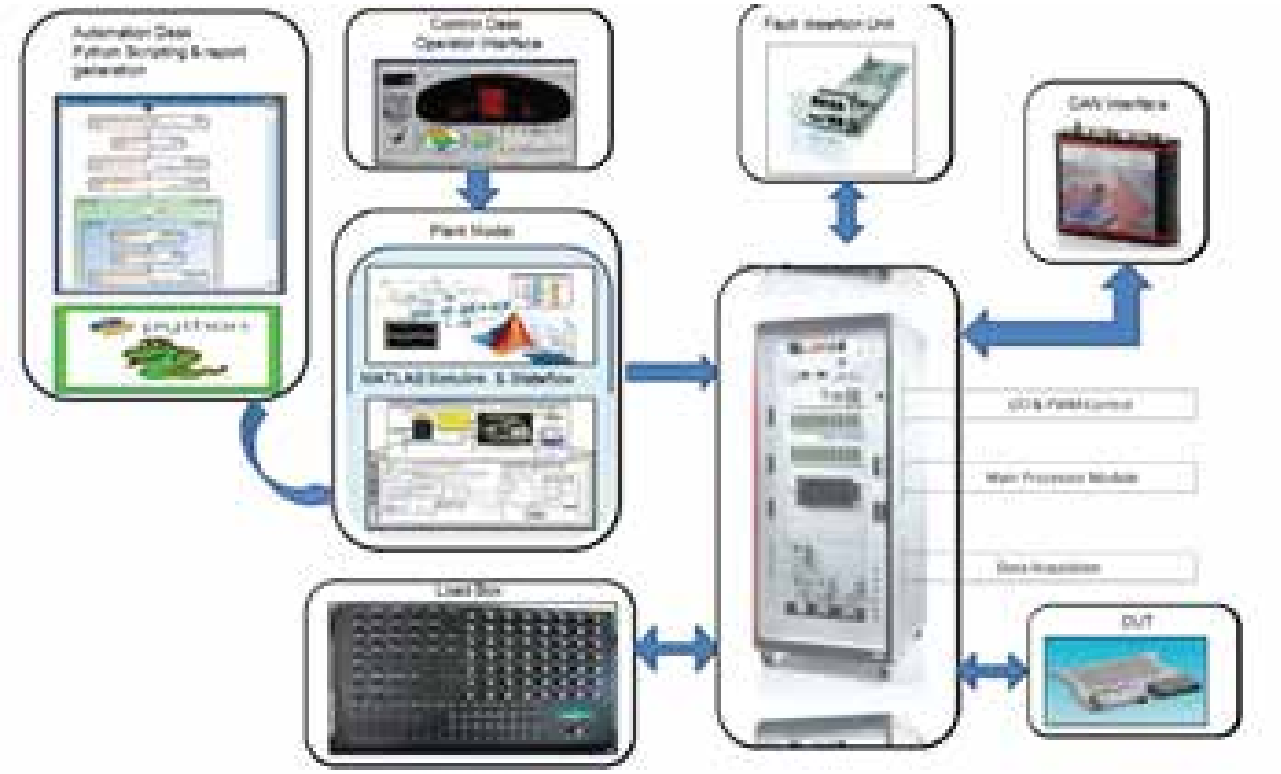


Figure 9: HIL Testing Environment

Benefits of MBSE Approach

Organizations that adopt Model-Based Design or approach realize their savings range from 20-60%, when in comparison to traditional methods. The bulk of these savings come from better requirement analysis combined with early and continuous testing and verification. As requirements and designs are simulated using models, defects are uncovered much earlier in the development process.

Model Based System Engineering technologies including simulation modeling, rapid prototyping, hardware-in-the-loop testing and automatic code generation offer better design results and considerable savings to OEM owing to the advantage of early detection through simulation.

Additionally, MBSE creates a structure for software reuse that permits established designs to be effectively and reliably upgraded in a more simplistic and cost effective manner. Model-Based Design for embedded software development lowers costs by identifying defects early in the development process and reduces the total number of latent defects thereby helping companies deliver higher quality systems at lower costs and in lesser time frames.

CONCLUSION

Model-based design offers developers a distinct advantage over traditional product development techniques. The ability to perform design validation and verification at the onset of a project as well as the ability to test system segments using rapid prototyping and automatic code generation (production code) provides a distinct advantage to the developer.

From a design and project management viewpoint, the ability to validate at different points during the design cycle enables management to more accurately forecast duration and costs. Model-based design also brings the concept of “design re-use” into the design cycle creating a framework for future savings.

REFERENCES

- Jerry Krasner, "Model-Based Design and Beyond: Solutions for Today's Embedded Systems Requirements", EMBEDDED MARKET FORECASTERS, American Technology International. January 2004. www.mathworks.com/tagteam/17878_91218v00_Krasner_Report.pdf
- Measuring Return on Investment of Model-Based Design By Joy Lin, Aerospace Industry Marketing Manager, MathWorks, https://www.mathworks.com/solutions/model-based-design/mbd-roi-video/Measuring_ROI_of_MBD.pdf
- Autosar. Automotive Open System Architecture, www.autosar.org
- MathWorks, www.mathworks.com
- Road vehicles — Functional safety — Part 6: Product development at the software level https://www.iso.org/obp/ui/#iso:std:iso:26262:-6:ed-1:v1:en

ABOUT L&T TECHNOLOGY SERVICES

L&T Technology Services is a subsidiary of Larsen & Toubro with a focus in the engineering services space, partnering with a large number of Fortune 500 companies globally. We offer design and development solutions through the entire product development chain, across various industries such as Industrial Products, Medical Devices, Transportation, Telecom and Hi-tech and the Process Industry. We also offer solutions in the areas of Mechanical Engineering Services, Embedded Systems & Applications, Engineering Process Services, Product Lifecycle Management, Engineering Analytics, Power Electronics and Machine-to-Machine and the Internet-of-Things (IoT).

L&T Technology Services

www.lnttechservices.com