Model-based Approach to Development of Engineering Systems

Jozef Živčák, Tatiana Kelemenová*, Michal Kelemen and Vladislav Maxim

Technical University of Košice, Faculty of Mechanical Engineering, Letná 9, 042 00 Košice

BIOGRAPHICAL NOTES

Dr.h.c. mult. Prof. Ing. Jozef Živčák, PhD. Is a professor of biomedical engineering at the Technical University of Košice. He was born in 1958. He received his MS and PhD. degrees from Technical University of Košice in 1995. He is from 2009 Doctor Honoris Causa of Uzhorod National University. His research interests include human biomechanics, medical sensorics, medical thermography and rehabilitation technology. Today he is head of head of Department of Biomedical Engineering and Measurement. Since 1998 he is an expert witness in machine and electrical technology. Professor has more than 280 publications in home and foreign journals. He is an author and co-author of 9 monographies and 12 books.

Tatiana Kelemenová, doc. Ing. PhD. She received M.S. degree in mechanical engineering from Technical University of Košice, Slovakia in 1996 and Ph.D. degree in Mechatronics from Technical University of Košice, Slovakia in 2005.She is a assistant of the Department of Biomedical Engineering, Automation and Measurement at the Faculty of Mechanical Engineering at the Technical University of Košice, Slovakia. His research interests include engineering metrology, measurement systems, uncertainty of measurements, mechatronic systems, and measurement of non-electric quantities. He has authored more than 100 journal and conference papers on these topics.

Michal Kelemen, doc. Ing. PhD. IReceived M.S. degree in mechanical engineering from Technical University of Košice, Slovakia in 1998 and Ph.D. degree in Mechatronics from Technical University of Košice, Slovakia in 2002.He is an associated professor of the Department of Applied Mechanics and Mecharonics at the Faculty of Mechanical Engineering at the Technical University of Košice, Slovakia. He has been awarded the 1998 "Price of the VolksBank" for the best M.S. graduate and 2007 Price "Scienist of the year". His research interests include mechatronic systems, intelligent robotic systems, dust mass concentration measurement, measurement of non-electric quantities, and microcomputer systems. He has authored more than 180 journal and conference papers on these topics.

Vladislav Maxim, doc. Ing. PhD. Received M.S. degree in electrical engineering from Technical University of Košice, Slovakia in 1984 and Ph.D. degree in Power semiconductor systems from Technical University of Košice, Slovakia in 2002. He is an associated professor of the Department of Automation and Control and Communication Interfaces at the Faculty of Mechanical Engineering at the Technical University of Košice, Slovakia. He has been awarded the Werner von Siemens Excellence Award 2005. His research interests include automation, servosystems, electrical Drives design, electrical equipmed of the electric vehicles, switched reluctance motor, power electronic and electrical CAD systems. He has authored more than 80 journal and conference papers on these topics.

KEY WORDS
Model-based development, dSpace, HIL simulations, Matlab.

ABSTRACT
The control functions for most engineering systems (automotives, aeroplanes, medical devices and other devices involve large quantities of software. Because of the great complexity involved and the safety requirements, manufacturers must absolutely guarantee that the device software has exactly the functions that were specified and that the functions work as defined. Thus, the specifications themselves must contain unambiguous, comprehensive function descriptions. Model-based development, in which a function is described by graphical models in MATLAB®/Simulink®/Stateflow®, is a proven method of implementing these requirements.

1. Introduction
Model-Based Design is a process that enables faster, more cost-effective development of dynamic systems, including control systems, signal processing, and communications systems. In Model-Based Design, a system model is at the center of the development process, from requirements development, through design, implementation, and testing. The model is an executable specification that you continually refine throughout the development process. After model development, simulation shows whether the model works correctly.
When software and hardware implementation requirements are included, such as fixed-point and timing behavior, you can automatically generate code for embedded deployment and create test benches for system verification, saving time and avoiding the introduction of manually coded errors [1-5].
Model-Based Design (fig. 1) allows you to improve efficiency by:

- *using a common design environment across project teams,*
- *linking designs directly to requirements,*
- *integrating testing with design to continuously identify and correct errors,*
- *refining algorithms through multi-domain simulation,*
- *automatically generating embedded software code,*
- *developing and reusing test suites,*
- *automatically generating documentation,*
- *Reusing designs to deploy systems across multiple processors and hardware targets.*

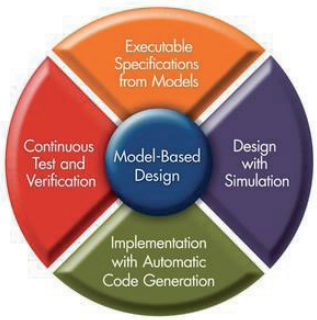

Fig. 1: *Model based design philosophy [5].*

Model-Based Design helps engineers achieve certification to safety standards by supporting requirements traceability, verification, and documentation. These capabilities span multiple design stages. For example, requirements linked to model are inserted as comments in generated code. Qualification kits, available for several verification tools, can reduce the amount of manual review needed.
It is also increasingly common for organizations to adopt Model-Based Design on large programs spanning multiple organizations. This allows system-level performance to be assessed and integration issues to be uncovered much earlier in the design process.
When detailed models from multiple organizations are combined, resulting models can contain hundreds of thousands of blocks. Modeling tools have evolved to meet these challenges with improved support for large-scale modeling, including support for composite models from other model files and support for signal buses.
When organizations adopt Model-Based Design, they improve product quality and reduce development time by 50% or more. It also causes that product will be cheaper and more competitive on market [1-5].

2. Model Based Design Process Steps
In generally, there are six steps to modeling any system:

- *Defining the System*
- *Identifying System Components*
- *Modeling the System with Equations*

■ *Building the model*
■ *Running the Simulation*
■ *Validating the Simulation Results*

We perform the first three steps of this process outside of the software environment before we begin building our model. Mainly, these first three steps are important and many people make mistakes in these steps. Finally, when system is modeled through the equations, next building of the model is more or less routine operation. It is important to say that every model is not perfect and every time we have to neglect any points and simplify system description. Overall process requires the experiences. When we will try to make absolutely prefect model, very complicated model and slowly simulation will be as the result of them [2].

Defining of the system - the first step in modeling a dynamic system is to fully define the system. If we are modeling a large system that can be broken into parts, we should model each subcomponent on its own. Then, after building each component, we can integrate them into a complete model of the system.

Identifying System Components - the second step in the modeling process is to identify the system components. Three types of components define a system: parameters (system values that remain constant unless you change them), states (variables in the system that change over time), and signals (input and output values that change dynamically during a simulation).

Modeling the System with Equations - the third step in modeling a system is to formulate the mathematical equations that describe the system. For each subsystem, use the list of system components that we identified to describe the system mathematically. Model may include: algebraic equations, logical equations, differential equations, for continuous systems and difference equations, for discrete systems etc.

Building the Simulink Block Diagram - after we have defined the mathematical equations that describe each subsystem, we can begin building a block diagram of our model for example in MATLAB/Simulink. Build the block diagram for each of our subcomponents separately. After we have modeled each subcomponent, we can then integrate them into a complete model of the system.

Running the Simulation - after we build the Simulink block diagram, we can simulate the model and analyze the results. Simulink allows us to interactively define system inputs, simulate the model, and observe changes in behavior. This allows us to quickly evaluate your model.

Validating the Simulation Results - finally, we must validate that our model accurately represents the physical characteristics of the dynamic system. We can use the linearization and trimming tools available from the MATLAB command line, plus the many tools in MATLAB and its application toolboxes to analyze and validate our model.

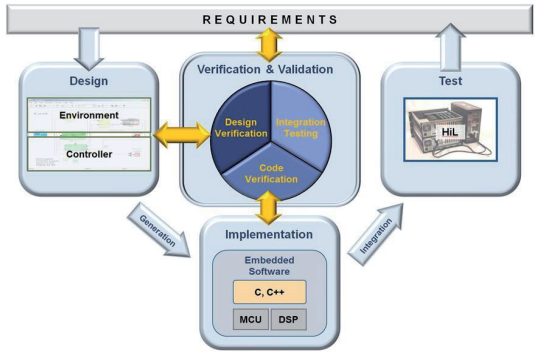

Fig. 2: *Early verification as part of Model-Based Design streamlines embedded control design with modeling, simulation, and automatic code generation [4].*

Model-Based Design with MATLAB and Simulink is an efficient and cost-effective way to develop complex embedded systems in aerospace, automotive, communications, and other industries. It enables system- and component-level design and simulation, automatic code generation, and continuous test and verification. [2].

Plant models provide another perspective on the system. Modeling the non-software parts of the system gives engineers another view into system behavior. Engineers can often learn more about system dynamics through simulation than from the real system because simulation provides details on force, torque, current, and other values that are difficult or impossible to measure on the actual hardware [4].

Creating plant models requires engineering effort, but this effort is often overestimated, while the value provided by plant modeling is underestimated. When developing plant models, it is a best practice to start at a high level of abstraction and add details as needed. Choosing a level of abstraction that is just detailed enough to produce the needed results saves modeling effort as well as

simulation time (see Figure 2) [2, 4].

System behavior is defined not only by the embedded control software, but also by the electronic and mechanical components, including the connected sensors and actuators. Early simulations in which the architecture is executed provide more insight when they are performed in a closed loop with plant or environment models. System-level optimization requires multi-domain simulations. It is impossible to optimize today's sophisticated systems by tuning one parameter at a time. To deliver maximum energy efficiency and highest performance at minimal material cost, engineers must optimize the system as a whole, and not just the embedded software [1, 2, 4].

Our research is focused on mechatronics systems and robotic devices. Example of built multi-domain simulink model of in-pipe micromachine is shown on figure3. As it is shown, model contains many nonlinearities and discontinuities. The model has been built for optimization of machine and for control design.

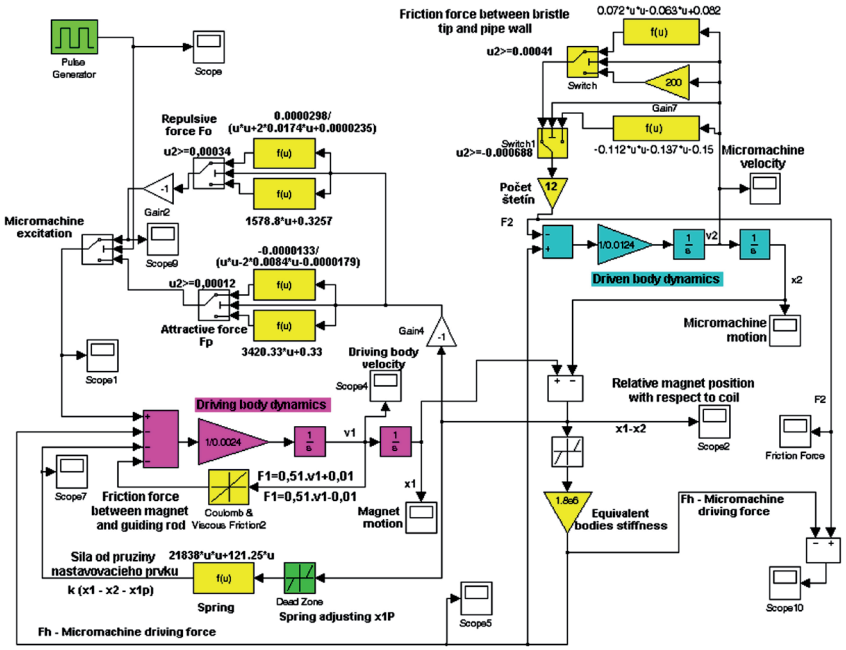

Fig. 3: *Example of multi-domain simulation model of in-pipe machine in Matlab/Simulink.*

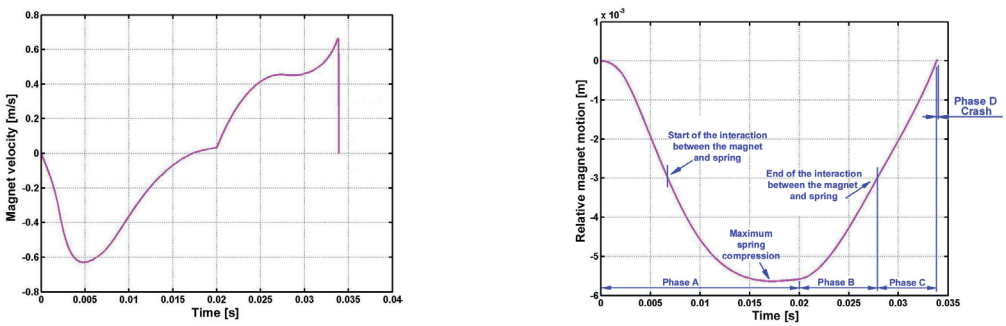

Fig. 4: *(a) Micromachine velocity in time. (b) Relative magnet motion with respect to driven body.*

The main purpose of this model is to study micromachine behaviour in various situations. The main purpose of this model is to study micromachine behaviour in various situations. This model also can give answer (fig. 4) to the many questions (How should be values of input variables? Where

is the weakness of the micromachine?, How is the velocity of machine? etc.)

3. Simulation Modes

There are different simulation modes (fig. 5) for the model (model-in-the-loop (MIL) simulation), the host implementation (software-in-the-loop (SIL) simulation) and the target implementation (processor-in-the-loop (PIL) simulation).

These methods (fig. 5) avoid work-intensive iterations in later development phases, and save time and money by:

- *Verifying at an early stage, by means of model simulation, that the model and requirement are correct*
- *Verifying that the code and the mode are consistent, and that the code correctly represents the model's functionality, by simulating the generated code on the host PC*
- *Verifying seamless traceability for documenting the software development*
- *Allowing resource requirements to be estimated at an early stage by simulating the code on the appropriate evaluation hardware*

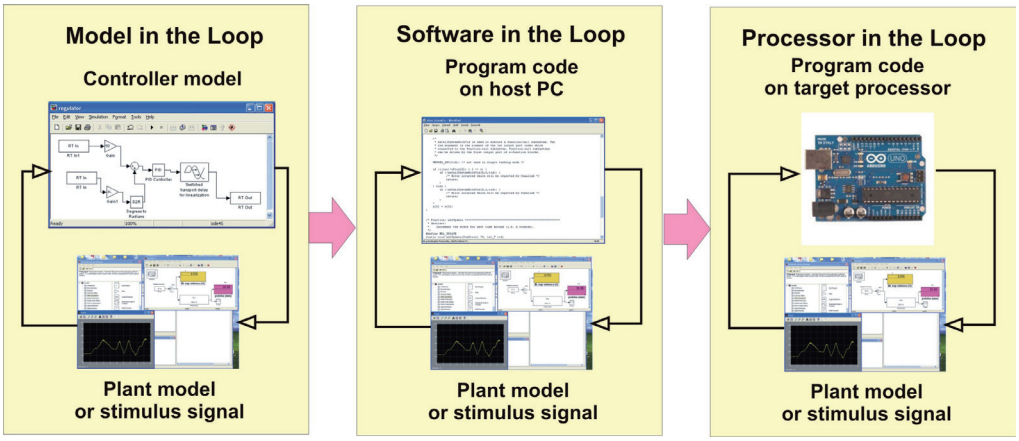

Fig. 5: *(a) Add a descriptive label of the figure here. (b) Add a descriptive label of the figure here. (c) Add a descriptive label of the figure here.*

4. Hardware-in-the-loop Simulation

Not only is the number of electronic control units (ECUs) in modern vehicles constantly increasing, the software of the ECUs is also becoming more complex. Both make testing a central task within the development of automotive electronics. Testing ECUs in real vehicles is time-consuming and costly, and comes very late in the automotive development process. It is therefore increasingly being replaced by laboratory tests using hardware-in-the-loop (HIL) simulation (fig. 6).

Time to market is speeding up, especially in automotive electronics. 90% of automotive innovations are currently connected with new electronics. Test drives can scarcely cope with the volume of systematic testing needed, especially just before start of production. The growing number of recall campaigns is a clear indication of this. It is little wonder that testing and error finding have become key tasks in the development process.

ECU testing typically is done using hardware-in-the-loop simulation. The ECU (prototype) is con-

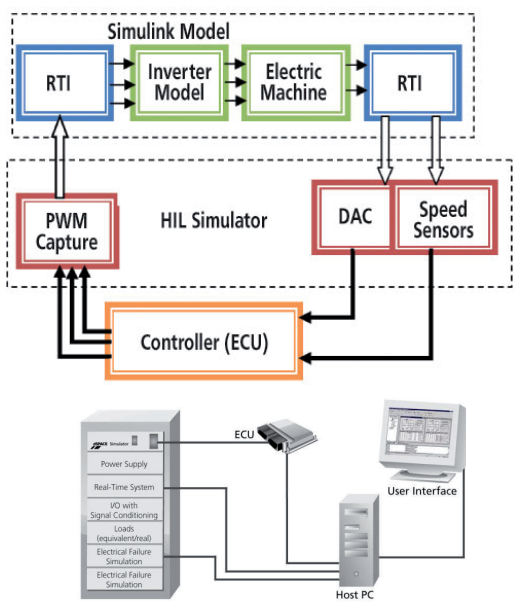

Fig. 6: *(a) The Components Required for High-Speed Simulation of the Electric Machine (b) Typical architecture for Hardware-in-the-loop HIL simulations [7, 8]*

nected to a real-time simulation system simulating the plant (engine, vehicle dynamics, transmission, etc.) or even the whole vehicle.

As HIL has become a standard method for testing ECUs and control strategies during the whole development cycle (i.e., not only after availability of the final ECUs), different needs of different users have to be addressed by the various test systems [7, 8].

The dSPACE software components are standardized and can be integrated in any dSPACE simulator. The tight integration of dSPACE software and the modeling tool MATLAB®/Simulink® from The MathWorks provides a powerful development environment.

dSPACE Simulator's graphical user interface provides a convenient and flexible environment. Simulated driving cycles, data acquisition, instrumentation, monitoring, test automation and all other tasks are executed graphically within dSPACE Simulator. The hardware requirements, however, vary immensely depending on the HIL application. For example, function tests typically are executed with simulators that have a fixed (super)set of I/O, and adaptations to the ECU are most often made in the cable harness. In contrast, acceptance tests call for flexible and combinable simulator setups [7, 8].

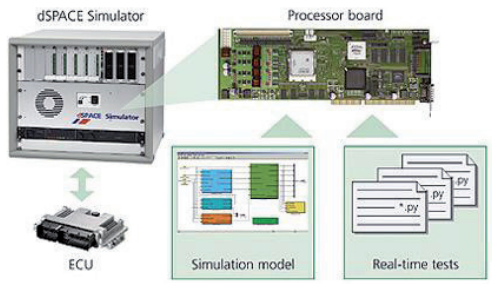

Fig. 7: *dSpace system for HIL simulation [2]*

5. Conclusion

Model-based design comprises four elements: modeling of desired behavior or reference designs; design exploration and refinement through simulation; implementation with code generation; and continuous test and verification throughout the development process. These elements address the design and verification issues inherent in today's electronic systems by letting engineers progress systematically from specification to implementation to verification, leveraging executable system models that unambiguously specify functional and physical requirements.

In traditional design flows, there can be no systematic testing of the entire system until it has been fully implemented. As a result, bugs remain hidden until late in the system development, when fixing them is significantly more costly and disruptive.

In contrast, model-based design enables system test and debug from the earliest stage of development, when most flaws are introduced. Models can be validated early on through simulation and verified continuously as the component models are refined with additional implementation detail. C, HDL and Spice implementations, as well as ESL models, can be incorporated to support existing workflows, design reuse and final integration testing. Components from different design teams can be integrated as they become available, ensuring that any changes do not degrade system performance and that bugs can be quickly isolated to the offending component [6-9].

6. Acknowledge

The authors would like to thank to Slovak Grant Agency – project APVV-0091-11, VEGA 1/1205/12. This contribution is also the result of the project implementation: Centre for research of control of technical, environmental and human risks for permanent development of production and products in mechanical engineering (ITMS:26220120060) supported by the Research & Development Operational Programme funded by the ERDF.

7. References

[1] dSpace GmbH. Model-Based Development of Safety-Critical Software: Safe and Efficient Translation of "Sicherheitskritische Software entwickeln" Published at: MEDengineering, 06/2012. available online at http://www.dspace.com.

[2] The MathWorks, Inc., Model-Based Design. Documentation Center. Cited 07-11-2013. Available online. http://www.mathworks.com/help/simulink/gs/model-based-design.html

[3] Otterbach, R.: Automotive Solutions, Systems and Applications. dSPACE GmbH. 2013. Available online. http://www.dspace.com.

[4] Sandmann, G., Schlosser, J., Maximizing the benefits of Model-Based Design through early verification. Embedded Computing Design. An OpenSystems Media publication. Available online. Page last refreshed: Mon, 08 Jul 2013

05:04:12. http://embedded-computing.com/articles/maximizing-benefits-model-based-design-early-verification/#

[5] MathWorks, Inc., Evolution of model-based design in aerospace. IML Group PLC. Available online. http://www.epdtonthenet.net/article/41911/Evolution-of-model-based-design-in-aerospace.aspx

[6] Karnofsky, K., Putting the system in electronic system design. EETimes Newsletter 2/4/2008, UBM Tech. Also available online. Cited 07-11-2013. http://www.eetimes.com/document.asp?doc_id=1271606

[7] Köhl, S, Jegminat, D., How to Do Hardware-in-the-Loop Simulation Right. dSPACE GmbH. SAE International. 2005. 2005 SAE World Congress Detroit, Michigan, April 11-14, 2005. SAE Technical paper series. Reprinted From: Controller System Software Testing and Validation (SP-1928). ISSN 0148-7191.

[8] Dhaliwal, A., Nagaraj, S., Jogi, S., Hardware-in-the-Loop Testing for Hybrid Vehicles. dSPACE GmbH, Evaluation Engineering, November 2009, NP Communications, LLC. dSPACE, 50131 Pontiac Trail, Wixom, MI 48393. Also available online. Cited 07-11-2013. http://www.evaluationengineering.com/articles/200911/hardware-in-the-loop-testing-for-hybrid-vehicles.php

[9] Trebuňa, F., Šimčák, F., Handbook of experimental mechanics. 1st edition. Košice : TU of Kosice, Fac. Of Mech. Eng. 2007. 1526 pages. ISBN 970-80-8073-816-7.