Combining Model-Driven Development and Model-Based Design to rapidly develop industrial automation applications

Traditionally, systems and embedded software developers have had to use Model-Driven Development (MDD) and Model-Based Design (MBD) environments separately, relying on manual methods to pass information between them for system design, verification, implementation, and test. Now, a potent combination of these best-of-breed solutions can benefit users with an environment that covers the complete life cycle, as demonstrated by a recent project where the two tool environments together helped develop the control unit system and software for a state-of-the-art stretch blow-molding machine.
Developing complex systems and software for embedded or real-time industrial automation applications can be a daunting challenge for designers and engineers. Projects require that engineers, developers, and controls specialists maintain high-quality systems and software, meet demanding schedules, and manage complexity while simultaneously fulfilling safety, accuracy, clarity, traceability, and productivity objectives. Typically, this problem has been managed by selecting specific tool solutions to meet the unique needs of each stakeholder. Though effective, the single-tool philosophy may not be enough to fully meet the demands placed on it. Combining environments, however, can be a powerful strategy to solve the quandary.

Ideally, this approach allows users to harness the best aspects of each tool by combining their power to create an environment collectively better than what they could create individually. Such is the case for engineers that select a combined Telelogic Rhapsody UML/SysML-based MDD environment and The MathWorks Simulink MBD for dynamic systems tool. By combining these best-of-breed solutions, users benefit from an environment that covers the complete process, from requirements to specification, design, development, implementation, and test. Notably for industrial automation projects such as robotics or stretch blow-molding technologies (and aerospace/defense and automotive applications), engineers who realize the full potential of the MDD/MBD approach can develop robust systems and software in an easy-to-use modeling environment and meet rapid time-to-market goals.

Complementary solution

Most developers have strong preferences about the modeling tools they use to design new projects. With the days of static paper designs long gone, choosing cost-effective tools that enhance workflow and boost productivity are very important organizational considerations. Companies that select an MDD or MBD solution may not think to combine the two in a complementary fashion; in many cases, the search for a holistic tool environment is based on finding one standalone solution that solves all problems. Accessing a powerful synergy of tools, engineers can assure that all facets of the process are covered.

The Rhapsody MDD environment covers standard UML 2.0/SysML-based software and systems design. Simulink is the de facto standard for dynamic systems modeling, allowing block diagrams of complex dynamic (mathematical) algorithms to be captured and analyzed. The combination of powerful MDD and MBD environments form a hybrid-modeling environment capable of capturing requirements, designing systems and software architectures, and developing logical and mathematical algorithms while supporting multiple work flows. By providing a flexible way to combine the domain-specific UML, SysML, plant, and control modeling, users can work with the tool that best meets their needs (see Figure 1).

**MDD and MBD: The power of two**

The MDD/MBD combination offers a complete process coverage solution while allowing users to select the tool most appropriate for each activity in the design cycle. With MDD, users can access requirements capture and analyze functions, define systems and software architecture, and develop logical algorithms. Each function can be simulated and tested to ensure accuracy; customized documentation can be generated for enhanced communication, and the actual implementation can be realized through production code generation.

With MBD, users can create plant models that define the dynamic behavior of the physical elements the system will interact with and develop the mathematical algorithms that control these plant models. This can all be simulated in MBD to ensure proper results. While the control algorithms can generate code for use in prototyping and production, the plant models can generate code for use in hardware-in-the-loop testing.

**Requirements capture and analysis**

Using the MDD process, successful projects always begin by capturing requirements and then analyzing them for accuracy. With the MDD/MBD synergy, users can capture textual requirements in requirements management tools (DOORS, RequisitePro, and so on), or with requirements capture tools (Word, Excel, and PDFs) or by using the MDD environment’s SysML requirements diagram to capture textual requirements that can be managed within the MDD environment. Users can create traceability links between requirements and Rhapsody SysML/UML and Simulink model elements that realize or test the requirements. From there, requirements analysis, coverage analysis, and impact analysis can be completed. This includes identifying requirements yet to be addressed, finding design elements not justified by a requirement, finding all design elements affected by a requirement change providing the true cost of that change, and ultimately understanding what requirements may be influenced by a design change to ensure that fixing one thing does not break others. To assist in this process, the Rhapsody Gateway requirements traceability solution provides

![Figure 1](image-url)
a view of design elements that cover a requirement, including elements contained in the Simulink models (Figure 2).

Benefits of the MDD/MBD synergy based on work flow

The benefits of plugging the Rhapsody MDD environment into the Simulink MBD tool are multifold. First, the synergy enables the different disciplines to use best-in-class tools. Control engineers can work in Simulink and systems and software engineers can use Rhapsody to develop a project, or one engineer can use both tools while preserving the natural work flow of the user.

The natural work flow of the Simulink user is preserved with the ability to access Rhapsody as a block in the Simulink model, enabling engineers to model the architecture, behavior, and logic of blocks representing electronic or embedded systems using a full UML/SysML environment. Additionally, Rhapsody designs can be tested against plant models with a true cosimulation that supports the full debugging and analysis capabilities of both tools. Co-code generation is supported as well, enabling combined code for rapid prototyping, hardware-in-the-loop testing, and production applications.

Testing a module design with a plant module is easy to accomplish in this environment. Users can model the physical plant in Simulink, then design the electronic module in Rhapsody. Connecting the module design to the plant model and then testing the module design with accurate plant models assure engineers developing industrial control and automation applications that the behavior and functionality of the design is correct before building the actual electronic unit.

A very powerful feature is the ability to integrate mathematical algorithms into Rhapsody, a common demand of engineers designing industrial control and automation designs. This supports work flows in which systems engineers use MDD to create systems specifications, and defines the system architecture, high-level behavior, and the areas where software developers use MDD to develop their applications. Meanwhile, control engineers use MBD to design complex control algorithms. This also supports the process when one engineer uses both tools.

Developing stretch blow-molding machine controllers

Working in combined MDD and MBD environment offers a powerful solution, but requires that users adhere to a work flow within the tool environments that allows each to capitalize on its respective strengths. By allocating different functions to each tool, users can benefit from a seamless work flow in their application. In a recent project where the two tool environments were combined to help develop the control unit system and software for a state-of-the-art stretch blow-molding machine, engineers, software developers, and controls specialists determined that mapping the project to a clearly defined work flow was critical to the project’s success. As the project moved forward, designers determined that the best work flow process was to assign the functionality piece to the MDD environment and utilize MBD tools to determine performance.

The stretch blow-molding machine, which included not just the molding operation but also postproduction loading and other automated features, required flexibility for numerous variants, meaning that the control system needed to be flexible enough to operate flawlessly on several different machine configurations. The stretch blow-molding process requires that the plastic is first molded into a preform using an injection-molding process. A preform typically resembles a thick-walled test tube and is produced with the necks of the bottles molded into one end including the cap threads, usually manufactured from a PET plastic material.

The advantage of utilizing this process is that processors obtain a more robust, clear, higher-quality finished container that can be filled with pressurized contents such as soft drinks. Manufacturers can produce bottles faster than traditional operations using stretch blow molding, and with this application, in a lights-out production facility. To do this, however, the stretch blow-molding machine relied heavily on the controller unit to provide the precision, control, and reliability necessary to obtain repeatable results from the molding operation.

When combining the MDD environment and MBD tool chain for this industrial application, the best practice is to use the MDD environment for requirements and functional analysis of the design. In this case, the blow-molding operation was described, the common function traits were determined, and functionality was identified. Next, the MDD tool was used for functional analysis, whereby the common function traits were grouped into use cases. These use cases were then analyzed in detail so as to identify the underlying functionality of the system. Also, the functionality was checked for completeness, correctness, and to assure the entire system was unambiguous. Specifically for the stretch blow-molding machine application this meant analyzing the separate
axis for the system, which included loading, unloading, molding, cutting, démodé and mode, transfer, and other functions. Lastly, the MDD environment determined the project’s architectural design using actors in the tool to describe interactions between the functions and utilizing the Harmony SE process. In the architectural design phase, the designers can allocate the identified functionality to the architecture and check the allocation strategies before implementation into the black box or white box view. Figure 3 shows the integrated systems/software development process.

For more detailed design work, engineers turned to the MBD tool. Here, components from the architectural design phase were executed with an emphasis on developing system control algorithms. From this point, software development was possible, with performance requirements such as speed and other nonlinear behavior such as timing analysis performed on the system architectural design with Simulink. From the outside, this may appear to be a linear step-by-step process, but the MDD/MBD tool chain is really an iterative process, meaning that the tools allow engineers to loop back to areas where analysis shows allocations require a redesign in either tool environment. As the controller was developed using an MDD and MBD process, the software development utilized the model to form the basis for subsequent software and hardware design and implementation.

Using this process, engineers, developers, and controls specialists alike benefit from a MDD/MBD integration that interacts where differences between requirements and performance appear, thereby finding potential issues early in the design phase and correcting them in the models, driving both productivity and quality in the finished stretch blow-molding machine controller unit. With a shared database and information sharing plus a strategic modeling function overlap in the respective tools, the most potent aspects of each were able to combine to form a very powerful industrial automation design environment.

**Multi-axis robotic application**

Another work flow where MDD and MBD tool environments can be used together is in the development and integration of simulation, mathematical algorithms, and state-based control algorithms. This can be illustrated using an example based on the development of a single axis in a multi-joint on a robot.

In the first stage of development, control stability analysis is carried out using mathematical models of the motor system (modeled in the Laplace or Z domain) and the control algorithm, typically PID or nonlinear adaptive control algorithms for different parts of the motor operating curve. An MBD tool environment is the natural tool to model and analyze these algorithms. More advanced algorithms may include stiffness modeling for more precise path planning of flexible robot arms.

The next stage of development is to build the overall control software for the robot, which can be achieved using the Rhaposdy MDD environment’s typical use cases work flow (such as Add Path Manually and Teach Path) to understand and scope behavior, sequence diagrams, and class and structure diagrams with embedded behavior realized by state charts. This model can be used to test the interaction of safety features and general control behavior driven by the interfaces.
These models then need to be brought together, partly because the MDD model will be missing some essential features for robot control and because engineers will want to simulate and test the integration of the software with a set of motor models. Currently, support is available to import code from the Simulink MBD tool into the Rhapsody MDD environment in a seamless manner. In this particular example, code generated from the MBD tool must be integrated in two places.

The most important is the integration of mathematically complex control algorithms that depend on specialized libraries that support specific integration and differentiation algorithms. These must be embedded directly into the MDD model and will be tightly integrated with the final code. If the robot is very flexible, the stiffness modeling and path-planning algorithms may also be modeled in the MBD tool and then integrated into the MDD model.

This process allows developers to test the path-planning and control algorithms in a tool such as the MBD tool. Then, engineers can integrate the generated code into a tool, which gives users model-specific system behavior in the MDD environment and generates code for the target easily.

The second place where MBD derived code should be integrated is the actual simulation code that supports the motor and load modeling. This element sits outside the main piece of software as it is modeling the environment, but integrates with the main code body through set interfaces (that is, velocity demand, actual angular velocity, and actual angular position) these being typical motor control system interfaces. This enables the software representing the control system to be tested with the motor model to see if they work correctly. This is carried out on the host, thereby testing interfaces and general behavior.

Finally, the software model is taken from the MDD environment and then placed onto a target processor. This is easily achieved due to the way that code from the MDD environment is structured, having a behavioral layer and a factory layer that allows integration of the behavior with the specific operating systems and target application. With the large number of standard Real-Time Operating System (RTOS) compilers, operating systems, and targets supported by the MDD tool environment, engineers and developers can benefit from a much faster design cycle.

**Two is better than one**

The benefits achieved from plugging MBD into MDD are powerful. By adding architectural capabilities to Simulink algorithms users can ensure the algorithms will interact properly with other elements in the design. Furthermore, it is possible to understand how software design decisions such as scheduling and sequencing will affect the integrity of the algorithms. In addition to creating a combined model that can be executed to ensure proper behavior, the actual production code is tested as a complete system in the software environment, thus making certain the customer receives a robust deliverable.

Working in a MDD/MBD tool environment is a powerful synergy that provides engineers a truly best-in-class solution. Both tools working together achieve complete process coverage from requirements capture, architecture design, logical and mathematical algorithm design implementation, and test. With architecture definitions, a focus on UML and SysML standards, and features such as Dynamic Model/Code Association, the Rhapsody environment extends tremendous MDD advantages to users incorporating the combined MBD Simulink synergy.

Dr. Graham Bleakley is a principal consultant with Telelogic Systems and Software Modeling Business Unit, where he currently works with a number of European aerospace/defense companies on projects such as Eurofighter, PAAMS, and Meteor, aiding in the definition and application of MBD strategies using UML 2.0 and SysML. Before joining Telelogic Graham worked as a research engineer, designing and building robots for safety-critical applications using MBD techniques. He has 15 years of experience in the fields of modeling, MB, and systems engineering. Graham has a B. Eng Hons in Mechanical Engineering and a PhD in Systems Design and Modeling.

Richard F. Boldt is the senior director of Rhapsody product marketing at Telelogic, and has served at the company since 1994 in roles as an application engineer, consultant, and product manager. He has more than 15 years of experience in the embedded systems and real-time industries, with more than a decade of experience focused on MDD and modeling environment technologies, as well as systems specification, software development, and hardware design experience in the automotive and aerospace/defense markets. Richard has a BS in Electrical Engineering from Rensselaer Polytechnic Institute.

Dr. Hans-Peter Hoffman is director and chief methodologist for systems design at Telelogic and has 22 years of experience in the design and development of complex systems in the aerospace/defense and automotive industries. As director of the simulation department of the missile division at MBB, Germany (now EADS) he developed a methodology-based approach for modeling and analysis of flight control systems, especially via software-in-the-loop simulation. A renowned specialist in real-time systems design, Hans-Peter delivers senior-level consulting, mentoring, and training to embedded systems developers. Hans-Peter has a PhD in Physics (Fluid Dynamics) from the University of Hamburg, Germany.

To learn more, contact the authors: