2007-01-0777

Best Practices for Establishing a Model-Based Design Culture

Paul F. Smith, Sameer M. Prabhu, Jonathan H. Friedman
The MathWorks

ABSTRACT

The transition to Model-Based Design must be managed carefully, both to demonstrate short-term benefits and to establish a culture that enables the full realization of the theoretical benefits of this approach. In this paper we introduce the concepts of Model-Based Design, highlight some of its benefits, and then discuss in detail the 10 best practices for adopting a Model-Based Design culture across an organization. These best practices have been gleaned from successful and not-so-successful transformations to Model-Based Design at companies from a variety of different industries.

INTRODUCTION

The adoption of embedded systems continues to transform the automotive industry. This transformation arises from opportunities for improving performance, safety, and maintenance through the use of sophisticated, on-board, software-based electronic controls. In addition to this transformation of the passenger vehicle industry, a second wave of embedded systems adoption is occurring in the commercial vehicle industry. Here embedded systems are being used to control hydraulic systems that previously relied on mechanical controls to achieve improvements in machine productivity as well as safety and maintenance. In both industries, the increase in system complexity poses a significant challenge to the capabilities of traditional systems development processes to meet program timing, cost, and quality metrics. To address these challenges, engineers at major vehicle manufacturers are skipping over a generation of system design processes based on hand coding and using graphical models to design, analyze and implement the software that determines machine performance and behavior.

Using models ensures that a final product meets system requirements. Models enable engineering teams with different specializations to work together efficiently and to communicate between people working on various stages of the design process; to identify and fix errors early on in the development process; and to automatically generate robust, efficient, and high-quality software. From the unique perspective of the software tool vendor, a distillation of underlying principles leading to successful application of Model-Based Design is possible. These range from specific practices related to automatic code generation to organizational issues that must be addressed. We examine each in some detail. The ‘right’ combination of these suggestions, tailored to the surrounding corporate culture in which Model-Based Design is to be immersed, must be carefully selected by the engineering managers leading the transformation.

WHAT IS MODEL-BASED DESIGN?

In Model-Based Design, the development process centers around a system model — from requirements capture and design to implementation and test.

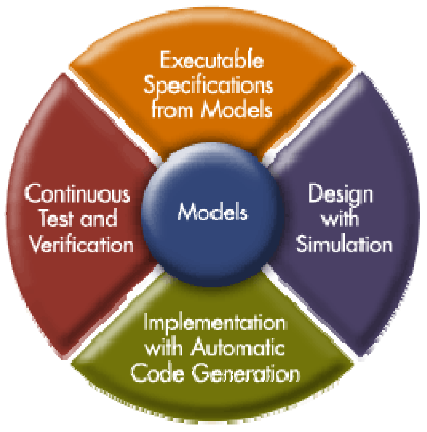
Executable Specifications from Models
Continuous Test and Verification
Models
Design with Simulation
Implementation with Automatic Code Generation

Figure 1: Model-Based Design.

This system model is the heart of an executable specification which is used and elaborated throughout the design flow. The executable specification can also include inputs and expected outputs or acceptance criteria, and the application environment, as well as links or references to the requirements.[1] The goal of the executable specification is to unambiguously communicate the design goals, as well as to allow feasibility and compatibility analysis of the requirements via simulations.

When software and hardware implementation requirements are included, such as fixed-point and timing behavior, code can automatically be generated for embedded deployment and test benches created for system verification, saving time and avoiding the introduction of hand-coding errors.

With Model-Based Design, engineers improve efficiency by:

- Using a common design environment across among project teams
- Linking designs directly to requirements
- Integrating testing with design to continuously identify and correct errors
- Refining algorithms through multidomain simulation
- Automatically generating embedded software code and synthesizable HDL code
- Developing and reusing test suites
- Automatically generating documentation
- Reusing designs to deploy systems across multiple processors and hardware targets

ADOPTION OF MODEL-BASED DESIGN

Why do companies adopt Model-Based Design? Sometimes there is a top-down management mandate driven by strategic plans to deploy a common set of tools and processes. Other times there is a grass roots initiative by engineers who used modeling at the university and are looking for tools to address their needs in their current jobs. At other times, Model-Based Design is an enabling technology to a broader initiative such as Six Sigma or Systems Engineering. Whatever the impetus to make the initial leap into Model-Based Design, the effort is sustained because of the payback that companies see. The payback comes in a variety of forms:

- Efficiency gains, such as a reduced number of hours required to complete a given project [1],[2]
- Reduced time to market [3]
- Improved quality [4]
- Reduced dependence on physical prototypes [5]

Additionally, engineers often just have more fun doing their job when they have the right set of tools.

MAKING THE TRANSITION TO MODEL-BASED DESIGN – 10 BEST PRACTICES

The transition – What is important to consider?

Wisdom is the sum of learning through the ages.

Smart organizations learn from their own mistakes. Wise ones learn from others' mistakes. Working with major corporations and governmental agencies over the years, we have seen many successes and a few mistakes as these organizations transitioned to Model-Based Design. We have gathered the following best practices together so our readers can learn from others and avoid the common pitfalls encountered when evolving to a Model-Based Design culture.

When considering the transition to Model-Based Design as a way of developing embedded computing systems, it is important to consider culture, tools, processes, organization, and strategy.

First, an organization must establish a culture that values the development and use of models and simulation as a core engineering activity.

Tools must be readily accessible and widely available to enable the productivity enhancements possible with model-based design.

Tools are used within the boundaries of defined processes. Each corporate or government agency has varying levels of formality in the design process driven by culture, regulation, best practices, the latest trends, and the people working within it. When adopting model-based design, as with any technology, effective and consistent surrounding processes must be established.

Traditional organizational models may need to be adapted or abandoned in favor of ones that support Model-Based Design workflows. Deliveries of work products between internal groups and the boundaries between designer and implementer get blurred with Model-Based Design. An effective organizational structure with visionary leadership is essential to success.

The transition to Model-Based Design must be driven by a strategy with clearly defined goals and supporting metrics. There should be a logical sequence to the evolution, which will differ from organization to organization. The strategy should build upon current strengths and shore up weaknesses in the design process. Top management needs to understand the strategy and hold the engineering staff accountable to deliver against the plan.

Best Practice #1: Identify the problem you are trying to solve

Before any process improvement can take place, it is necessary to have a deep understanding of where the relative strengths and weaknesses are in the current organization or processes. There must be metrics in place to support this understanding. For organizations that do not have a metrics program in place, Best Practice #0 should be to establish one. A doctor would not prescribe medications to fix an illness until the nature of the illness and the patient's medical history are well understood. The same is true when adopting model-based design. There are numerous sub-elements within the modern embedded systems development process that could be operating inefficiently. Examples of such sub-elements include:

- Schedule Predictability – Can the organization predict and deliver to a schedule?

- Software Quality – Are there excessive defects produced by the current development process? At what stage of development are they introduced? Are there sections of the code or algorithm that tend to produce more defects?

- Time to market – Are products being produced in time to meet the rapidly shifting demands of today’s consumers?

- Productivity – Are the LOC, person, and time period (or other measure) up to industry standards?

- Defect Tracking and Configuration Management – What level of sophistication exists to manage and track the work output of the development organization?

- Reuse – Are the work products reusable?

- Product documentation – Is there a system in place to publish documentation on the work products of the development organization?

- Rapid Prototyping – Do processes exist to prototype new functionality?

- Validation and Verification – Are bugs being caught and removed before product delivery? At what cost?

Each organization adopting model-based design needs to decide where the current weaknesses in their processes are, what the cost to the organization is as a result of these weaknesses, and consequently what problem to tackle first.

When deciding what problem to solve first, it is also important to understand how long it will take to solve that problem. Executive management may have a myopic view of the return on investment of the tools and technologies they authorize. Pick something to fix that will demonstrate to the sponsors that their investment is producing results.

Best Practice #2: “Rule of Two”

To extract the minimally acceptable return on an investment made in tools, training, and organizational change needed to justify the move to Model-Based Design, the models produced must be used for at least two different purposes. For example, models could be used to validate requirements through simulation and to automatically generate documentation. Alternatively, models could be used to define and develop control algorithms that are used during rapid prototyping and also to automatically generate code for production controllers.

The choices made for the two process elements should be chosen carefully, using Best Practice #1 above. The goal is to achieve a maximized return on investment. Model reuse across different design stages is important to achieving this maximum return.

Organizations that pursue a single use of models *can* be successful, but tend to find the burden of adopting the new technology overwhelming. They may fail to reap the benefits of their investment of time and capital.

Best Practice #3: Use models to generate production code

Organizations increasingly use software to achieve hardware commonality – that is, to keep the basic hardware the same, but make it behave differently by changing the embedded system software that controls the hardware. This allows manufacturers to achieve economies of scale on the hardware side, but at the same time allows the product to be customized for multiple customer needs by simply changing the embedded software, in effect satisfying two seemingly contradictory objectives of product customization versus mass production. It is no surprise then that embedded system software is fast replacing mechanical hardware as the key determinant of product performance characteristics. Also, it is the embedded system software that controls the major systems on a vehicle, and thus in effect it is the software that makes the vehicle “move”. Given the importance of embedded software, it is only natural to extend the “Rule of Two” mentioned earlier to cover the fact that one use of a model should be to automatically generate embedded software.

Using models ensures that a design meets the associated system requirements and also allows errors to be found early in the design process when it costs less to fix them. The latest advances in code generation technology allow production-ready embedded software to be generated automatically from a model.[11] When the embedded software is automatically generated from a model it allows the reuse of the entire testing infrastructure built to test the model. It thereby allows the verification of the generated software and also errors, if they can be found and fixed early on. Also, if requirements change or design changes have to be made, they are made to the model. The software can be then regenerated very quickly and efficiently from the model. This significantly shortens the software turnaround time for responding to requirements and/or design changes.

Generating embedded software automatically from models does require a cultural shift in an organization. Historically, control designers built models for their designs, and software engineers developed embedded software by hand based on these model specifications. With automatically generated embedded software, the software engineer's focus shifts from spending time rewriting the algorithmic code every time the design and/or requirement changes to spending more time on integrating algorithmic code with the rest of the embedded system, as well as on specifying and setting up the infrastructure for the same. This is a significant shift, and it requires that the metrics used to measure software engineers' productivity also change accordingly. In addition, models facilitate a closer working relationship between controls and software engineers, and break down the traditional barriers between these job functions. The organizational leadership has to encourage and facilitate this in order to realize the full benefits of Model-Based Design.

Best Practice #4: Models are the sole source of truth

In addition to the benefits accrued from following the "Rule of Two" and using models for multiple purposes in the development process, a significant benefit of Model-Based Design is that models can be reused over multiple programs. Models then become the organization's intellectual property (IP) and reusing them over multiple programs allows the IP to be leveraged significantly to ensure that the program exceeds quality and schedule metrics. However, these benefits can only be realized if the model does indeed contain the true IP.

A logical extension of this and of Best Practice #3 is that the model should be the sole source of truth regarding the embedded system. The embedded software automatically generated from the model is what makes the vehicle "move". If an algorithmic error is found or if some last minute requirement changes are made when going through final verification and testing on pre-production vehicles, it is indeed very tempting to "fix" the embedded software itself to avoid having to go through the automatic embedded software generation and integration process all over again. However, if the software engineer acts on this temptation, it quickly leads to the model and software getting out of sync, and the model does not contain the true IP. Thus, if the model is then reused in a different program, the algorithmic errors or inconsistent requirements then have to be dealt with all over again on that program as well.

This raises organizational implications similar to those discussed with Best Practice #3. Controls and software engineers have to work together to ensure that any last minute changes are indeed propagated back to the model. The organizational leadership should support and encourage this, and emphasize the need and benefits for doing so. If not, it is indeed very easy for controls and software engineers to diverge and thereby prevent the realization of all the benefits afforded by Model-Based Design. In effect the organization has to have the discipline and rigor to use and enforce models as the sole design language for embedded software.

Best Practices #5: Use the transition as a learning opportunity

Transitioning from a conventional development process to Model-Based Design can sometimes seem daunting. However, it is essential that organizations follow Best Practice #1 and use this transition to learn about themselves and the issues critical to their long-term success. This introspection leads to a clear fundamental understanding of the current process, its strengths and weaknesses, and also a deeper understanding of the core competencies of the organization. Armed with this in-depth information, an organization can better tackle the task of transitioning to Model-Based Design.

A common temptation in such a transition is to "outsource" the transition to Model-Based Design to a third party. While outsourcing in itself is not bad, it needs to be judicious and targeted at areas that are not core competencies of the organization. If the conversion of existing IP to a model-based environment is outsourced, the organization loses the opportunity to re-examine this IP, to question which parts of the IP should really be transferred over, or to fix any errors and bugs that might exist in this IP, etc. Thus, a significant opportunity for improving product quality can easily be squandered due to ineffective outsourcing. The organization should certainly get help from third parties, but that help should be focused on creating tools and utilities that allow easier transition to Model-Based Design, recommendations for better processes for Model-Based Design, etc.

Another common temptation is to "flip the switch" and transition completely over to Model-Based Design in order to realize its benefits. This is fraught with significant pitfalls. First, there are several elements of the current development process or product that are likely to be efficient, or well established, and do not need to be changed during the first pass. Throwing these out will cause needless rework in adapting to Model-Based Design. A smarter way would be to judiciously select the elements of the process and product that need to change and to leave the rest as is. An example would be to identify problem areas in the current process or product and focus Model-Based Design on these areas only in the first iteration. Second, it is important to recognize that there are several elements of the current development process or product that would probably never be changed over to Model-Based Design. As an example, low-level device drivers, or control features that are to be phased out in upcoming generations, should not be transitioned to Model-Based Design. Taking a deeper look at the current development process to identify appropriate areas for Model-Based Design causes minimal disruption to the development process and at the same time yields the maximal benefit.

When deciding which parts of the product IP to model, completely new control features should be tackled first. Next, move on to problem areas in the design – things that are constantly changing or causing issues. Finally, slowly migrate more stable portions of the control system into the background as time permits.

Best Practice #6: Focus on design instead of coding

Sometimes, when Model-Based Design is introduced, software engineers are immediately concerned that their jobs are in jeopardy. However, this fear evaporates as the design team sees that software engineering is still occurring as part of Model-Based Design. However, the traditional role of the software engineer shifts from the combined activity of coding for implementation to one of architecting the software at the beginning of the process and elaborating the executable specification to enable code generation closer to the implementation phase.

For example, software engineers must develop a flexible architecture that allows legacy code to be integrated with new features that are developed using Model-Based Design. Additionally, software engineers must add elaborations to the model to ensure that the generated code will work on the target processor, such as when an algorithm model is created with floating-point values and the targeted processor is fixed point.

Best Practice #7: Integrate the development process

Model-Based Design thrives where there is a supporting infrastructure that reinforces the best practices outlined in this paper. In other words, to be successful, Model-Based Design needs to be part of the mainline product development process, not an overlay.

Often existing processes and metrics need to be modified to incorporate Model-Based Design. For example, style guides need to be developed and enforced through reviews at appropriate project milestones. Configuration management of the Model-Based Design artifacts needs to be addressed. In a code-centric process, the code is usually the primary or sole artifact under control. In Model-Based Design, controlled artifacts could include test vectors, expected outputs, model components, the version of the modeling environment used, etc. [6] Similarly, requirements management and verification processes and tools need to be integrated with Model-Based Design tools. Traditionally, software requirements are not verified until there is a prototype implementation available. However, when Model-Based Design is introduced, simulation verification can be added prior to hardware verification.

Detailed and ongoing training and competency development plans must be established. These include generic product training and customized process-specific guidelines and rules.

Lastly, and perhaps most importantly, the metrics used to evaluate a project's status must be updated to account for the shift to Model-Based Design.[8] For example, under a traditional development process, one might measure the number of lines of code generated per day to assess the "health" of a project. When all the lines of code can be generated "at the push of a button," this metric loses its meaning.[8]

Best Practice #8: Designate a champion who has influence and budgetary control

Organizations that are most successful in implementing the transition to Model-Based Design have selected a strong, experienced and highly respected champion. Sometimes this person needs to be a consensus builder, and sometimes a benevolent dictator.

This person typically has seniority and a proven track record of orchestrating organizational change. The migration will be feared by some and welcomed by others. The champion needs to be able to channel the energies of those welcoming the new way of doing business and to calm the fears of (or possibly remove) those that are not.

The champion should be well respected by peers, subordinates, and superiors. If there is overt or covert undermining of the transition due to the political landscape surrounding the champion, the entire transition may be placed at risk for reasons having nothing to do with the technology or process.

This person should have either direct budgetary controls or the support of executive sponsors who do. The investment in people, capital, and training is a function of the magnitude of the overall Model-Based Design deployment and expected return on investment (ROI). These investments can be substantial for a large organization, and budgetary discretion will be an important attribute of the champion.

Best Practices #9: Have a long-term vision

The migration to Model-Based Design takes time. For a moderate-sized embedded systems design and development organization in the transportation, construction equipment or aerospace-defense industry who is starting from scratch, the time scale for change is measured in years, not months. ROI can first be achieved after the first models are built, simulated, and defects removed from the design *before* going to code or hardware. The full benefits can only be achieved over time.

It is quite common for organizations trying to catch up to this trend to "hope" that they can transform their development organization to one based on the concepts and technology of Model-Based Design in three to six months. This is quite unrealistic. Perhaps a small outfit with a limited product line and 5-20 developers can make the switch in this short time frame. Teams performing primarily research can also make the leap more quickly, because they tend to be less encumbered with legacy processes (but the corresponding ROI tends to be lower.) Larger organizations tend to be heavily encumbered with legacy process elements, and the time to "glue" any commercial tool chain into their processes takes a considerable amount of time. Some factors to consider include how to marry Model-Based Design with the legacy systems and process for:

- Defect tracking
- Configuration management

- Documentation and publishing
- Tuning, trimming, or calibration
- Project management
- Home brewed embedded operating systems or custom hardware targets
- Metrics programs (how you measure productivity will change [8])

(Note: This is not meant to be an exhaustive list.)

Be patient and think for the long term. Have a master plan and keep grinding away at it. Celebrate small successes along the way and change the plan as needed as conditions change. Applying other best practices listed above, carefully choose the sequence in which Model-Based Design concepts will be applied to demonstrate value up the chain of command (and hence to those making the resource commitments necessary to sustain the transition).

There is an immediate benefit most engineering organizations report as they begin the transition: their people have more fun doing their job. Most people – especially engineers – like working with powerful tools. This leads to higher retention rates and increased job satisfaction.

Best Practice #10: Partner with tools suppliers

As mentioned at the beginning of this paper, smart organizations learn from their own mistakes, and wise organizations learn from the mistakes of others. As organizations go through the process of transitioning to Model-Based Design, tool suppliers are intimately involved in this transition due to their critical role in supporting the transition. Thus, they accumulate critical experience with a variety of organizational situations and factors that have to be dealt with in managing a successful (and sometimes unsuccessful) transition. Using this accumulated experience for managing the transition of your own organization to Model-Based Design can significantly reduce the steepness of the proverbial learning curve. Doing so then allows an organization to realize the benefits of Model-Based Design sooner than they would have otherwise. It thus causes the return-on-investment (ROI) breakeven point to be reached earlier. Not only is the initial ramp up on the learning curve reduced, but learning the best practices the tool supplier has gleaned from other experience early on in your organization reduces the effort required to sustain Model-Based Design, as shown graphically in Figure 2. Thus, partnering closely with your suppliers and involving them closely in your transition plan allows you to leverage their experiences, avoid common pitfalls, and negotiate a successful transition.

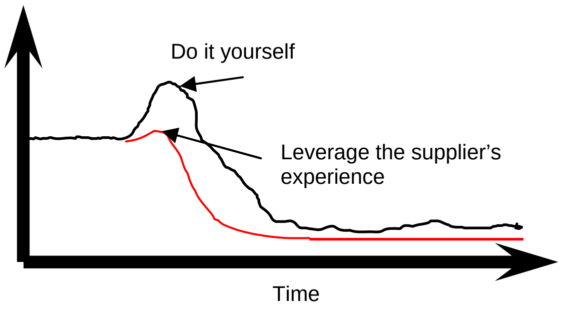

Figure 2: Learning curve effects in transitioning to Model-Based Design.

CONCLUSION

The application of Model-Based Design has been a well-established, well-documented, and highly refined practice for developing embedded control systems like those used for automotive powertrain or chassis controls. Through many years of observing and participating in the transition of large and small, commercial and governmental, transportation, communications, and other embedded controls development organizations, these best practices have been distilled and are now being shared. These are not hard-and-fast rules, and they may not apply in every situation. Each organization must take a step back to look at itself in the context of the transition to Model-Based Design. Only then can a judicious application of these best practices result in maximum return on the investment in people and technology required to modernize the embedded controls development processes in use.

REFERENCES

1. Peter J. Schubert, Lev Vitkin, and Frank Winters, “Executable Specs: What Makes One, and How are They Used?” 2006 SAE World Congress, Detroit, MI, April 2006, 2006-01-1357.

2. Jeff Thate, Larry Kendrick, and Siva Nadarajah, “Caterpillar Automatic Code Generation,” SAE Paper 2004-01-0894.

3. https://tagteamdbserver.mathworks.com/ttserverroot/Download/20542_91205v00_Nissan_userstory.pdf

4. Bill Potter, “Achieving Six Sigma Software Quality Through the Use of Automatic Code Generation,” 2005 MathWorks International Aerospace and Defense Conference: www.mathworks.com/industries/aerospace/miadc05/presentations/potter.pdf

5. https://tagteamdbserver.mathworks.com/ttserverroot/Download/27372_91334v00_Ford_us.pdf

6. Gavin Walker, Jon Friedman, and Rob Aberg, “Configuration Management Within Model-Based Design,” SAE Paper 07AE-328.

7. The MathWorks, Inc. – Automotive Technical Literature: www.mathworks.com/industries/auto/technicalliterature.html

8. Arvind Hosagrahara and Paul Smith, “Measuring Productivity and Quality in Model-Based Design,” SAE Paper 2005-01-1357.

9. Jon Friedman and Jason Ghidella, “Using Model-Based Design for Automotive Systems Engineering — Requirements Analysis of the Power Window Example,” SAE 2006-01-1217.

10. Tom Erkkinen, “Safety-Critical Software Development Using Automatic Production Code Generation,” SAE Paper 07AE-168.

11. Grantley Hodge, Jian Ye, and Walt Stuart, “Multi-Target Modeling for Embedded Software Development for Automotive Applications,” SAE Paper 2004-01-0269.

12. Proceedings from The MathWorks International Automotive Conferences:

- www.mathworks.com/company/events/programs_de/iac2004/iac_confirm.html
- www.mathworks.com/industries/auto/iac/presentations.html
- www.mathworks.com/industries/auto/iac06/presentations.html

CONTACTS

Paul Smith — Consulting Services Group, The MathWorks, Inc.
Paul.Smith@mathworks.com
www.mathworks.com

Dr. Sameer Prabhu — Applications Engineering Group, The MathWorks, Inc.
Sameer.Prabhu@mathworks.com

Dr. Jon Friedman — Automotive Marketing Group, The MathWorks, Inc.
Jon.Friedman@mathworks.com