Benefits and Drawbacks of Model-based Design

Arno Bergmann
Department of Electrical Engineering and Computer Science, Institute for Systems Engineering, Bochum University of Applied Sciences, Lennershofstr. 140, 44801 Bochum, Germany

* Corresponding author. E-mail: arno.bergmann@hs-bochum.de
Received: 18 April 2014; Accepted: 28 April 2014; Published online: 27 June 2014
DOI: 10.14416/j.ijast.2014.04.004

Abstract
The Model-based Design approach, as propagated by The MathWorks, is a state-of-the-art method in the fields of aerospace, defense and automotive developments. The obvious advantages of Model-based Design of a convenient, understandable graphical description of systems, continuous verification and validation at all stages of development as well as its inherent robustness against coding errors have made it a state-of-the-art method in fields such as automotive systems and aerospace and defense. Despite the vast number of success stories associated with this approach, Model-based Design is not a standard method throughout the entire industry, especially not for small and medium sized enterprises. Hence, consulting on the introduction of Model-based Design into development teams is a recurring task for the author. Presenting an industrial project, the development of a velocimeter (spatial frequency sensor system), benefits as well as obstacles corresponding to Model-based Design are introduced. The paper's object is giving detailed insight into the method based on first-hand experience. It will be concluded that Model-based Design is a favorable approach even for small and medium sized enterprises.

Keywords: *Model-based Design, Real-time workshop, Spatial filtering velocimetry, Embedded systems*

1 Introduction to Model-based Design

The MathWorks's method of Model-based Design for software engineering is solely based on a representation of the software interacting with its environment, called model. This model can be understood as a graphic implementation specification of the system components.

The key feature of this method is that the model can be run for simulation or system-testing purposes at any step of the development process. This means that the system behavior can be assessed right from the requirements phase of the project until series production without the need to change system description.

Figure 1 illustrates the workflow of Model-based Design: Starting with the system model, containing both descriptions of the system under construction as well as the corresponding environment, the code for the target platform is automatically generated, which may be C or C++ code for microcontrollers or digital signal processors as well as VHDL or Verilog for FPGA or ASIC designs.

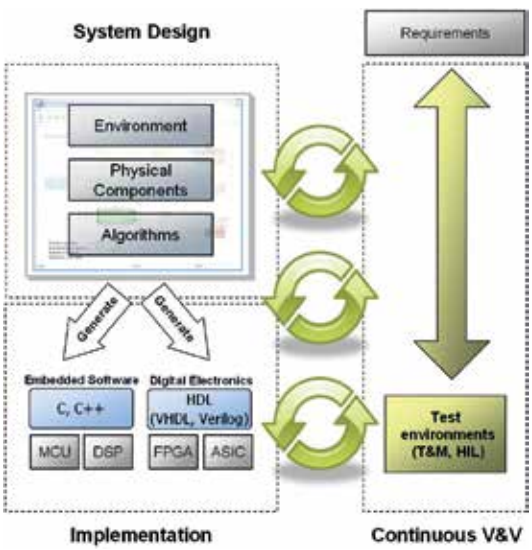

Figure 1: Model-based Design Workflow (The MathWorks).

The main advantage of this method is that the engineer never has to abandon the model in favor of another development tool. Thus, the need for an initial system sketch, a first mock-up, a prototype etc. is not given. All process phases (feasibility study, simulation, coding for prototypes and series, verification) are based on the same model.

The process of Model-based Design is known for quite some time now. However, it is hard for the potential user of this method to assess the suitability for his project at hand. The following describes the actual work with Model-based Design introducing an industry project and shows the consequences for the engineering process.

The project at hand has been funded by the European Union (EFRE Z0305MW02). The author contributed to this development by introducing Model-based Design to the project as well as developing the entire algorithm including the corresponding system software.

2 Application Example: COVIDIS Sensor System (FRABA group, Cologne)

COVIDIS is a camera based system for the measurement of translations in industrial environments as shown in Figure 2. Potential applications are steel mills, cutter controls, quality control of product lengths, automatic control of extrusion machines and so on.

2.1 *Sensor system details*

The sensor system is a complete digital implementation of a Spatial Filtering Velocimeter (SFV), consisting of a LED illumination, a telecentric lens system, a line scan camera as and an FPGA/DSP-platform as depicted in Figure 3.

The concept of SFV is shown in Figure 4: The object's illuminated surface is projected on an optical grid (spatial filter). A photodetector acquires the light intensity behind the grid, therefore producing the spatial filter function.

Surface features are either mapped on an opaque or transparent section of the grid, thus contributing to the light intensity depending on their position. As a result, the light intensity (the amplitude of the spatial filter signal) is modulated by a frequency proportional to the object's velocity.

Figure 2: COVIDS sensor system (FRABA).

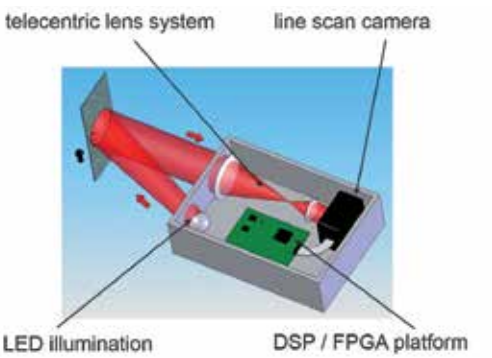

Figure 3: COVIDS sensor system in principle (FRABA).

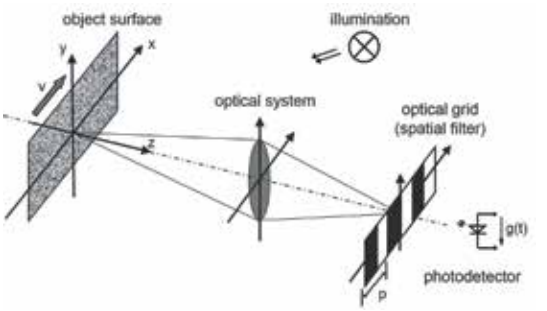

Figure 4: Principle of spatial filtering.

The main task of the sensor's software is estimating the main frequency of the spatial filter function. Auxiliary tasks comprehend camera control, scaling, object detection, etc.

Using a line scan camera the optical grid is replaced by a scalar product of the grid's transmission function and the camera image. This is shown in Figure 5: A sequence of line scan pictures (bottom) is being processed, the red line marking the current image. This image weighted by the rectangular transmission function is shown on the top, the sequence of the scalar products (sum of the values in the top graph)

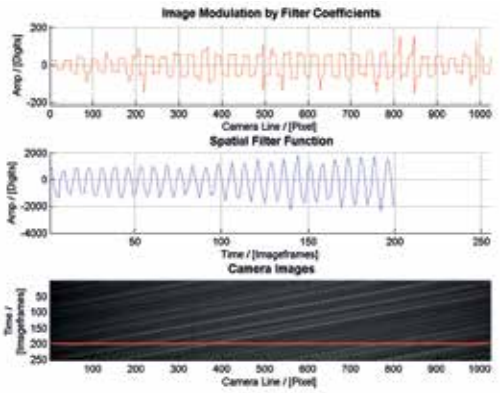

Figure 5: Calculation of an SFV signal.

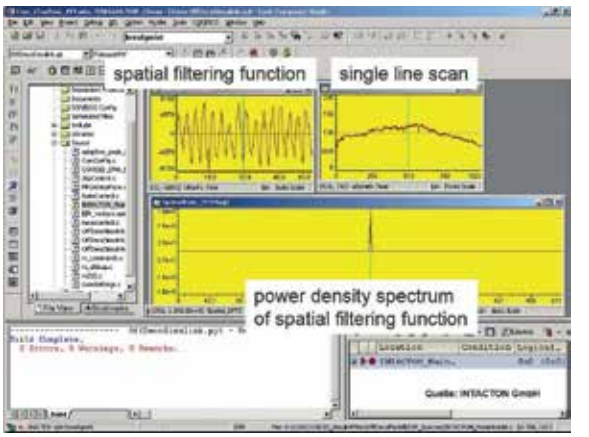

Figure 6: Implementation running in an IDE (FRABA).

can be seen in the middle, being the periodic spatial filter signal.

2.2 *Model implementation*

First step of the development process with Model-based Design is obviously creating a model. The model is simply mapping all functions into a signal-flow graph including the external interfaces into the target hardware.

This model can be executed for simulation purposes right from the first implementation. The basic system operation can be verified on a simple level, allowing plausibility checks on made assumptions.

For the feasibility phase of this project no effort has been spent on modelling the environment. The stimulus is given by an image sequence recorded with a frame grabber, which allows quick verification of initial assumptions.

Figure 5 of the previous section is such a result of the feasibility study. This illustrates one strength of Model-based Design: The feasibility study provides comprehensive results, including documentation for the next steps as well as short movie sequences for customer presentations.

2.3 *Target system connectivity*

After having developed a working system model the next project step requires usually a verification on a target hardware. The required code can be automatically generated from the model, so there is no need to test on the series hardware, the code can be brought onto a prototype system in the first step.

An interface allowing automatic generation of a software project in an integrated development environment (IDE) can be generated, which requires about five days' work. Having done that, the Code is automatically generated, included into the compiler's IDE, compiled, brought to the target and executed on a single button press. Figure 6 shows this for the described project with TI's Code Composer Studio, including graphic in-system evaluations.

To make this happen, the model code needs to be embedded into the so called "harness code" which provides the interfaces to the DSP periphery. Since the harness code is target specific, it has to be written by hand. The effort required depends solely on the number of peripherals used.

2.4 *Further automation*

In order to simplify the development steps a graphical user interface (GUI) has been created. Via this GUI, the following recurring development steps are automated:

- Simulation
  Execution of single test cases.
- Code generation
  Automatic code generation, compilation and execution on the target
- Profiling
  Automated execution time profiling of the model, which may be required to optimize certain model parts.

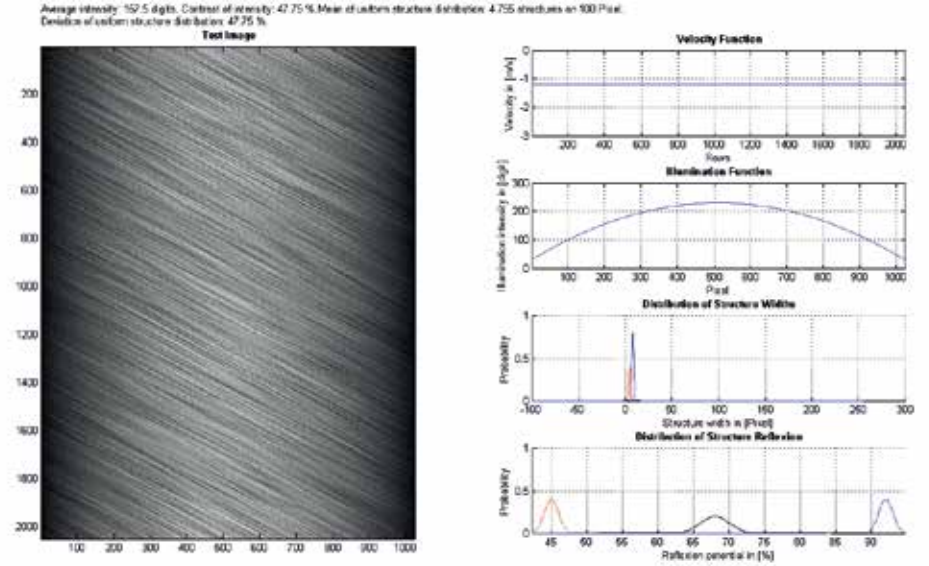

Figure 7: Implementation running in an IDE (FRABA).

- SIL-testing
  The main verification of the system has been done by a software-in-the-loop (SIL) technique. The current project required a large number of test cases which could not be run and assessed one by one. To achieve a high test coverage, the SIL environment can be conFigured to generate a large number of artificial linescan sequences based on parameter sets and execute these repeatedly with different model versions. An example of a single test case is given in Figure 7.

3 Conclusions

Applying Model-based Design to the described project has been a major factor of success. However, the price for improving the development efficiency is introducing a complex method and corresponding tool to the team as well as the high licensing costs for the actual tools.

The benefits as well as the obstacles of Model-based Design to this project are described in detail in the following sections.

3.1 *Benefits of Model-based Design*

Consistent documentation and implementation: The model description is a very appropriate source for the system documentation. Since it is also the source of the actual code, documentation an implementation can easily be kept consistent.

Coding errors: Automatic code generation completely eliminates coding errors, making implementation specification (model) and actual implementation consistent as well.

Documentation effort: As stated before, the model's graphic representation make it appropriate for documentation purposes and significantly reduce documentation costs.

Communication improvements: The model can easily be understood be each and every project member, allowing simple but effective presentations for discussions of technical details or outsourcing of code development into other teams.

Continuous verification and validation (V&V): V&V is a key bonus of this method, allowing tests at every project stage. Problems can be tackled in the earliest possible stage, reducing the costs for late and potentially risky product changes.

Possibility of major changes at late project phases: Due to the block oriented and therefore modular workflow as well as the automatic code generation, large functional blocks may be replaced at a late project stage. This is a main advantage if concurrent algorithms are available, e.g. autocorrelation and fast-fourier transform.

Code re-use: The simulation blocks can be included into an own library without extra effort, because they automatically exist as modules. Therefore, re-use of the solutions is very simple.

No coding effort: Target specific implementation requires a lot of effort in development and debugging and may require an expert engineer. With automatic code generation this effort is reduced to implementing the harness code.

Less debugging: Extensive simulations reduce the need for in-system debugging.

3.2 *Obstacles of Model-based Design*

Platform independence: In case code execution time has to be optimized, device-specific libraries need to be included. These hinder porting onto another target.

Difference between prototype and series code: The first prototype code, including large floating-point operations as well as completely generic code, took a hundred times longer to execute on the target than the series code. This was a major obstacle for the described progress because it took much longer for first prototype verifications than planned.

Implementation of harness code: Implementing a harness code requires an initial training.

Adapting the toolchain to the target hardware: The initial adaption of a new IDE requires detailed insights into the code generation process producing significant project costs and delay.

Implementation of special functions: Model blocks provided by Simulink are limited in their possibilities. Therefore it may be necessary to generate proprietary blocks by handwritten C-Code, which defeats the Model-based approach to a certain amount.

Model implementation: Implementing a model is a unique task which requires a significant amount of practice and experience.

4 Conclusions

Model-based Design is the method of choice for embedded system development, as long as the functions to implement are sufficiently complex to benefit from the advantages of continuous V&V and detailed simulations. The obstacles of this method can be handled even by small and medium-sized enterprises and can be addressed during the project as long as the extra-effort for initial training is accounted for in the planning phase.

References

[1] Y. Aizu and T. Asakura, *Spatial Filtering Velocimetry*, Berlin Heidelberg New York: Springer Verlag, 2006.

[2] B. Degener, "Implementation of a fully automated test environment in MATLAB/ SIMULINK for SFV systems," Diploma Thesis, University of applied sciences Gelsenkirchen, 2006. (orig: "Erstellung einer vollautomatischen Testumgebung in MATLAB/ SIMULINK für Ortsfrequenzfiltersensoren," Diplomarbeit, Fachhochschule Gelsenkirchen, 2006.)

[3] INTACTON GmbH, "Data sheet: Optical length and velocity sensor COVIDIS 08/2007," *INTACTON GmbH*, 2007.

[4] THE MATHWORKS TRAINING SERVICES, "Real-Time Workshop Embedded Coder for Production Code Generation," *The MathWorks Training Services*, 2005.