Functional Development with Modelica: A Use-Case Analysis

Stefan-Alexander Schneider* stefan-alexander.schneider@bmw.de
Tobias Hofmann† hofmann.tobias@me.com

* BMW AG, 80788 München, Germany
† TU München, Germany

Abstract

This contribution deals about the development steps of an embedded controller. The activities of the role function developer are explained for the simple example traffic light controller. The method of virtual integration is explained to establish short feedback loops.

Keywords: embedded systems; simulation; modeling; short feedback loops; co-simulation; virtual integration; Vee-Model; systems engineering

1 Introduction

The behavior of a dynamic system is in general too complex to treat by theory or formulas. Several simulation methods have been established for analyzing such systems. The virtual integration method is conducted on a model to gain knowledge about the (intended) real system behavior. This abstraction typically allows to focus on the main properties of the studied multi-domain system and their effects. These components require specific domain solvers for mechanical, electrical, etc. components. In this context, the term co-simulation has been established. The virtual integration is based on co-simulation and described in [7, 10]. There is a rather huge literature on the Vee-Model and systems engineering, see e.g. [1, 6, 12, 9, 4]. For more general introduction see, e.g., [5, 15, 16].

In the following, we demonstrate how to develop a control algorithm for an embedded controller designing the entire system - both the plant and the control components - with the modeling language Modelica. This approach allows us the modeling and simulation of the entire system, and thus the validation of the design decisions in an early phase of the development.

2 Model Example

Traffic is in general a good example for dynamic systems. The planning of traffic flow includes among others the avoiding of traffic jams and the optimization of traffic flows. No wonder that traffic planing is a current political issue as the article *Guck mal, wer da fährt* in the *Süddeutsche Zeitung* of May 15th 2012 shows. According to this article, the traffic of a city like Munich is controlled by more than 1.000 traffic lights. All these traffic lights serve to control the traffic and arrange for all traffic participants in some sense optimal traffic flow and an acceptable (system) behaviour.

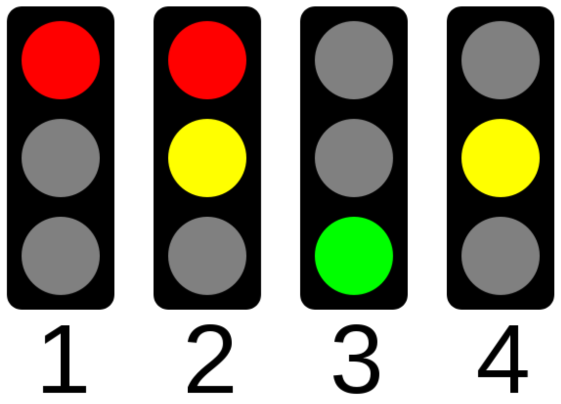

Figure 1: The typical sequence of coloured lights, see table 1.

Figure 1 explains the typical European sequence of coloured lights, see, e.g., [14].

traffic light	meaning
red light	do not cross
red and yellow light	prepare to cross
green light	cross
yellow light	if safe to do so, stop

Table 1: The typical sequence of coloured lights and their meanings.

A signal timing plan is a graphical representation of the traffic light phases for the correspondings traffic lights, similar to a so-called GANTT chart, see also Figure 2.

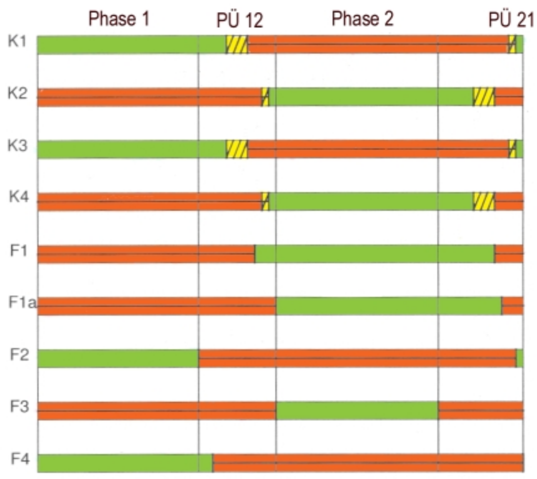

Figure 2: A typical signal timing plan is a graphical representation of the traffic light phases similar to a GANTT chart.

Traffic engineering programs like LISA+, see Figure 3, facilitate a planning processand are especially developed for intersections with a large number of signal groups and traffic lights, see, e.g., see [13].

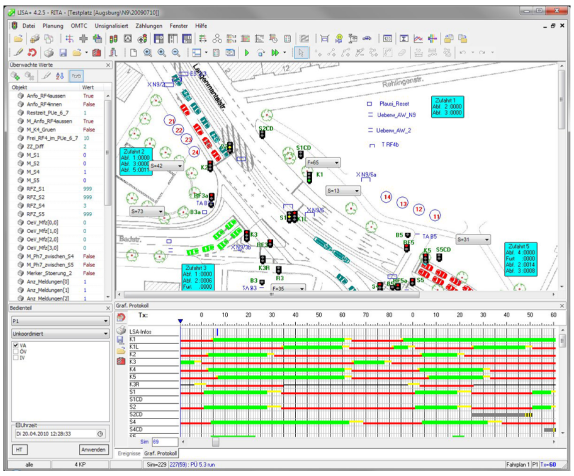

Figure 3: The GUI of the software package LISA+ for the planning of a specific traffic scenario.

Although the analysis of such systems of traffic lights contains a number of interessting (non-linear) mathematical taks, we simplify the considered task to a single two crossing road intersection. The main reason for this is that we can better study the phases of the development process for such a simple example.

In this report we therefore restrict to the following model example: a simple intersection of two roads with four traffic lights, see Figure 4. According to the wind rose, the lanes are denoted by *North*, *East*, *South*, and *West*.

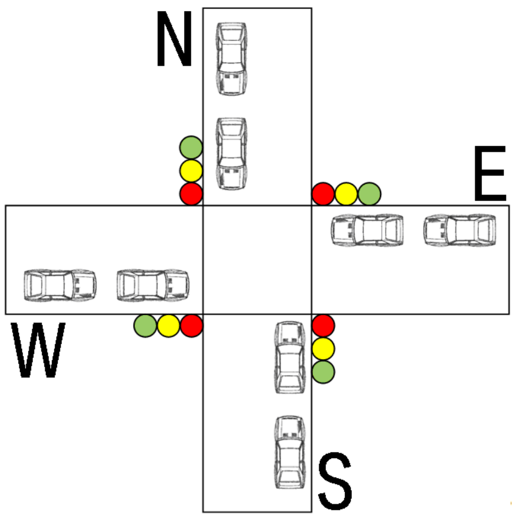

Figure 4: A simple road junction serves as for this paper sufficiently model example where the road crosses a north-south direction with a road in east-west direction.

We describe and study in the following sections a workflow with its development phases for an embedded control system for the traffic lights, using among other the environment design, modeling and simulation language Modelica and its modeling and simulation tool Dymola.

3 The Development Phases

The development phases of the Vee-model that are considered in this paper are, see Figure 4, [17]: system level requirements, system design, module design, module implementation, module integration and test and finally system integration and test on an embedded controller.

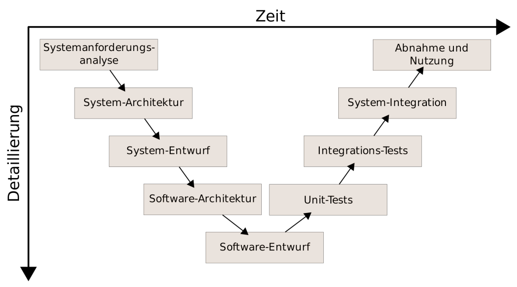

Figure 5: The Vee-model and its development phases.

3.1 System Level Requirements

In this development phase we formulate the requirements to the system and with that to the controller to be developed.

For this purpose, we define the *waiting time* W_i of a single vehicle crossing as the time from arrival at the intersection to the crossing of the intersection and with that leaving the system. We define the *total waiting time* by the overal sum $W = \sum W_i$ and formulate with that the first requirement to the intended control algorithm:

$$\Delta W := W_{new} - W_{old} \to \min, \tag{1}$$

where W_{new} denotes the overall waiting time of all vehicles after and W_{old} before the considered cycle of green phases. Although W_{new} and W_{old} are hard to measure, the difference ΔW, however, is not: the difference is depending on the number of vehicles crossing the intersection in the considered green phase cycle. This first requirement has the consequence that control algorithms with so-called vacant green phases are rated worse.

Let us now assume a scenario with high traffic rate. In this case, there exists a simple strategy to avoid vacant green phases by just not switching the priority lane. A controller that serves only one direction has no vacant green time and therefore fulfills the first requirement.

We formulate therefore a second requirement to prevent this undesirable behavior:

Both directions are to be served periodically. (2)

We denote the *green times* $t_{NorthSouth}$ and $t_{EastWest}$ for the two directions, the *minimum green time* by and t_{min} and the circulation time t_{Clock} by the sum of all traffic lights phases. Therefore holds

$$0 < t_{min} \leq t_{NorthSouth}, t_{EastWest} < t_{Clock} \tag{3}$$

and with that $2 \cdot t_{min} \leq t_{Clock}$.

3.2 Outlook: Additional Requirements from Functional Safety

Finally, note that there are additional requirements e.g. from functional safety:

1. *emergency control mode*: Traffic lights from the major roads turn off and the traffic lights from the side streets blink yellow. This indicates that the proper operation of the traffic lights is not guaranteed and supports on the other hand the given traffic signs.
2. *secure on the electrical level*: If a light source is out of order, so none of the directions may be given the green signal to avoid a so-called *hostile green* and it should, if possible, the red signal be given.

These two additionally requirements stemming from the functional safety are not in the scope of this paper and will therefore not be considered in the following.

3.3 System Design

The considerations so far motivate to model the entire traffic system as a controlled system composed by two components for

- the *plant component* consisting of four lanes and
- the *controller component* calculating the duration of the green times $t_{NothSouth}$ and $t_{EastWest}$ by an *explicit computation rule* from given numbers of the traffic members provides by the plant component.

Finally, we describe the interfaces. The interfaces between the plant and the controller component are given by six real values: four *numbers of vehicles in the waiting queues* n_{North}, n_{East}, n_{South}, and n_{West} and the two green phase values $t_{NorthSouth}$ and $t_{EastWest}$.

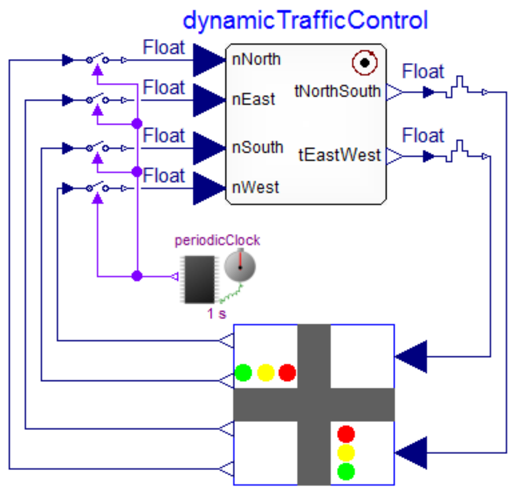

Figure 6: The composition of the system in Modelica. The symbol above right in the controller indicates the atomic exectution behavior of the controller component.

The definition of the components and its interfaces, modeled in Modelica see Figure 6, is the first fundamental design decision, see the library SAFEDISCRETECONTROL in [11].

3.4 Module Design and Implementation

3.4.1 Component Plant

The component plant simulates an intersection of two roads, which runs in a north-south and an east-west direction, see again Figure 4. The four waiting queues are named by the facing directions North, East, South and West. We suppose a simple growth model for the population of the four lanes

$$\dot{n} = \begin{cases} c - c_{Out} & \text{if corresponding lane has green} \\ c & \text{else,} \end{cases} \tag{4}$$

where $n \geq 0$ denotes $n_{North,East,South}$ and n_{West} and $c \geq 0$ represent the *uniform growth constant*, and c_{Out} the additionally *decay constant of the waiting queues* in the green phase of the corresponding lane representing the number of vehicles passing the intersection.

The two opposite lanes are governed by two opposing traffic lights with the same signal sequence. Note that we neglect in the following the modeling of the yellow phase and it holds for the green time phases

$$t_{NorthSouth} + t_{EastWest} = t_{Clock}. \tag{5}$$

3.4.2 Component Controller

The component controller realizes roughly speaking a mapping from $\mathbb{R}^4$ to $\mathbb{R}^2$ fullfilling the requirements (1) and (2) - consequently, there exists an infinite number of implementations!

A very simple first strategy to fulfill the requirements is distribute the available time t_{Clock} equally to both green phases

$$t_{NorthSouth} = t_{EastWest} = t_{Clock}/2, \tag{6}$$

see also Figure 7 for the implementation in Modelica.

We initialize the component controller with red lights for both directions.

3.5 Module Integration and Test

In this phase, we validate the module designs and their implementations by so-called Model-in-the-loop simulations before we move on to the next development phase. Therefore, we analyse given use cases and test the controll algorithm by virtual integration.

We set for the module test phase the following general parameters:

- the minimim green time $t_{Min} = 10[s]$,
- the circulation time $t_{Clock} = 150[s]$, and

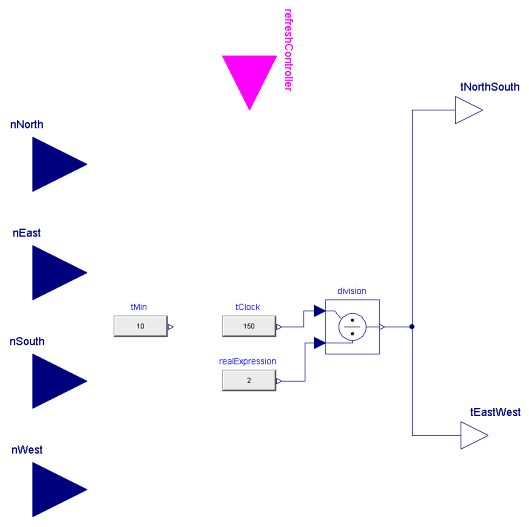

Figure 7: The implementation of the equations (6) for the symmetric strategy in Modelica.

- the initial numbers of vehicles in the waiting queues $n_{North} = n_{South} = 100[1]$ and $n_{East} = n_{West} = 50[1]$.

3.5.1 First Use Case: Equally Busy Lanes

In this use case, we assume that both roads *north-south* and *east-west* are equally frequented and chose the following use case specific parameters

- the growth constants $c_{EastWest} = c_{NorthSouth} = 1\left[\frac{1}{s}\right]$ and
- decay constant of the waiting queues $c_{Out} = 2.2\left[\frac{1}{s}\right]$.

Because it holds for the growth and decay constants

$$c_{EastWest} + c_{NorthSouth} \leq c_{Out}, \tag{7}$$

there may pass more vehicles through the intersection than new ones join in the waiting queues. We therefore expect a good controller to reduce the waiting queues over time.

Figure 8 shows the signal time plan corresponding to Figure 2. The evolution of vehicle values $n_{North} = n_{South}$ and $n_{East} = n_{West}$ in the waiting queues is given in the Figure 9.

This Model-in-the-loop simulations confirms the symmetric control strategy as expected. We therefore study a further asymmetric use case to test our first implementation.

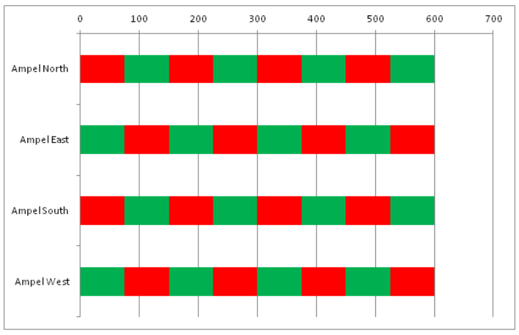

Figure 8: The signal time plan of the first use case.

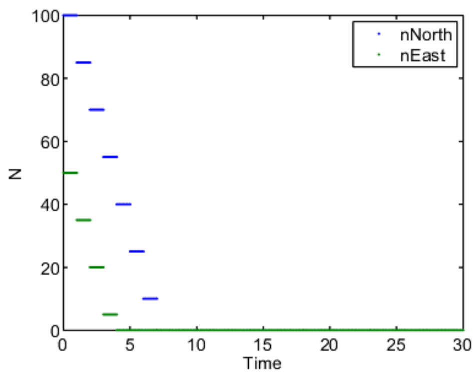

Figure 9: The evolution of vehicle values $n_{North} = n_{South}$ and $n_{East} = n_{West}$ in the waiting queues for the first use case.

3.5.2 Outlook: System Simulation

As an outlook, we mention here, that Modelica provides further tools for simulations like a full system simulation. The toolbox Modelica3D allows to visualize the full intersection. Figure 10 provides a picture of a movie produced by Modelica3D. For further details see [3, 2].

3.5.3 Second Use Case: Main and Secondary Road

We change the first use case only slightly and then simulate a scenario in which the north-south road is less traveled than the east-west road and assume the following parameters

- the growth constants $c_{EastWest} = 2\left[\frac{1}{s}\right]$, $c_{NorthSouth} = 0.2\left[\frac{1}{s}\right]$ and
- decay constant of the waiting queues $c_{Out} = 2.2\left[\frac{1}{s}\right]$.

This time, as many vehicles arrive at the intersections as may pass through the intersection. We expect a good controller not to increase the number of vehicles in the waitings queues.

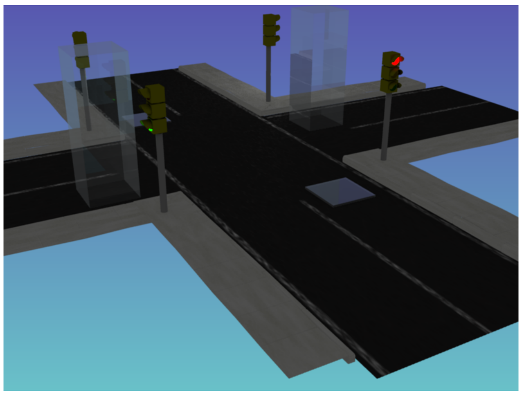

Figure 10: This picture of a movie produced by Modelica3D shows the behavior of the system example intersection. The waiting queues are visualized by boxes with hights depending on the length of the corresponding waiting queue.

The evolution of vehicle values $n_{North} = n_{South}$ and $n_{East} = n_{West}$ in the waiting queues is again given in the Figure 11.

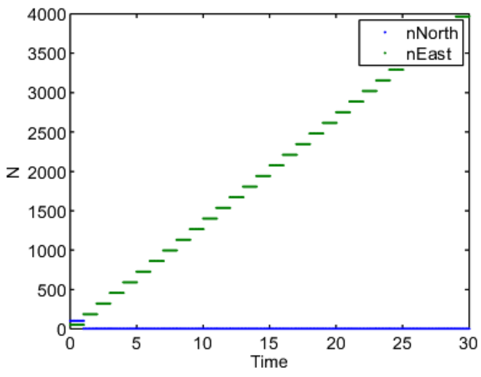

Figure 11: The evolution of vehicle values $n_{North} = n_{South}$ and $n_{East} = n_{West}$ in the waiting queues for the second use case.

This time, we observe that the North-south road drops to 0 and remains constant, where as the East-west road linear increases. The constant *green phase ratio*

$$t_{NorthSouth} : t_{EastWest} = 1 : 1 \tag{8}$$

obviously does not reflect the asymmetric *vehicle growth ratio*

$$c_{NorthSouth} : c_{EastWest} = 1 : 10 \tag{9}$$

good enough. This undesirable behavior motivates another requirement for the implementation of the controller algorithm.

3.5.4 Additional Requirement on System Level

We introduce two key indicators:

- the *ratio of the waiting queues* defined by

$$r_{wq} := (n_{North} + n_{South}) \,/\, (n_{East} + n_{West}) \qquad (10)$$

  and

- the *ratio of the green phases* given by

$$r_{gh} := t_{NorthSouth} / t_{EastWest}. \qquad (11)$$

We assume that the longer the green phases, the more vehicles may pass the intersection. In the sense of the customers of the intersection, we therefore additionally require

$$r_{wq} \approx r_{gh}. \qquad (12)$$

3.5.5 Module Design and Implementation for the alternative Controller

In this section, we develop a second, alternative controller, and solve therefore the system of equations (5) and (12) to fullfill mathematical exact the requirements. We define the *load distribution* for the two roads

$$\lambda := \frac{n_{East} + n_{West}}{n_{North} + n_{East} + n_{South} + n_{West}}. \qquad (13)$$

The value $\lambda = 0 = 0\%$ reflects no traffic in east-west direction and consequently minimum green time for east-west and $\lambda = 1 = 100\%$ correspondingly for the other direction. Then, keeping in mind the mimimum green time requirement (3), this yields to

$$\begin{aligned} t_{EastWest} &= \min(t_{Clock} - t_{Min}, \max(t_{Min}, \lambda \cdot t_{Clock})) \\ t_{NorthSouth} &= t_{Clock} - t_{EastWest}, \end{aligned} \qquad (14)$$

see also Figure 12 for the implementation in in Modelica.

The so designed controller has the following desirable properties:

$$\begin{aligned} \lambda = 0 &: t_{EastWest} = t_{Min}, \\ \lambda = 1 &: t_{EastWest} = t_{Clock} - t_{Min} \end{aligned} \qquad (15)$$

with corresponding

$$t_{NorthSouth} = t_{Clock} - t_{EastWest}. \qquad (16)$$

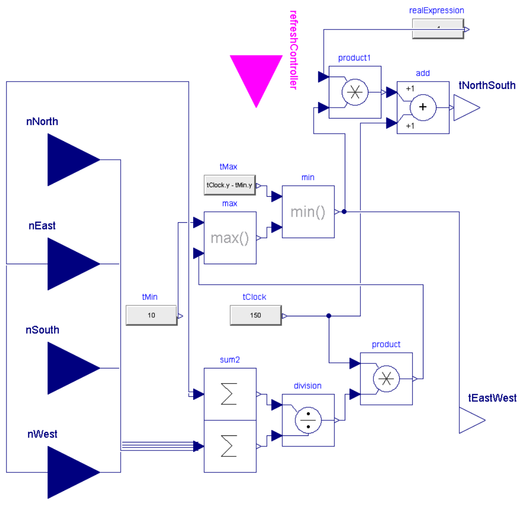

Figure 12: The implementation of the alternative control algorithm given by (14) in Modelica.

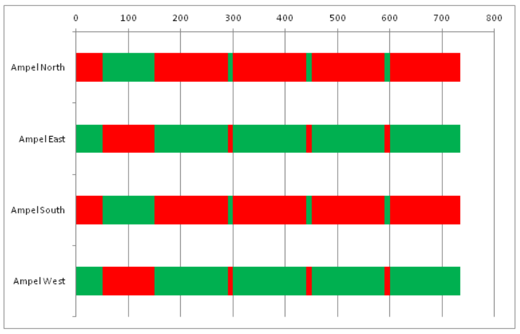

Figure 13: The signal time plan the alternative control algorithm given by (14) in Modelica.

3.5.6 Module Test and Integration of the alternative Controller

Figure 13 shows the results of the Model-in-the-loop simulation of the second implementation of the controller for the second use case.

Figures 14 and 15 present the evolution of the number of vehicles and the green phases.

The alternative controller responds to the asymmetric load much better. After a transient phase the ratio of the green phase 1 : 10 reflects the ratio of the loads 1 : 10 almost perfectly.

3.5.7 Regression of the First Use Case

Also the first use case can be controlled by the alternate controller. Although it produces, in contrast to the first controller, a small oscillation, but remain limited to vehicle values.

A simple validation shows that the second controller also produces the expected behaviour for the first use

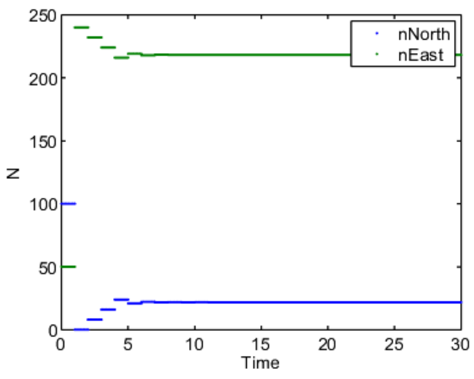

Figure 14: The evolution of the numbers of vehicles for the second use case *main and secondary road* with the second implementation of the controller.

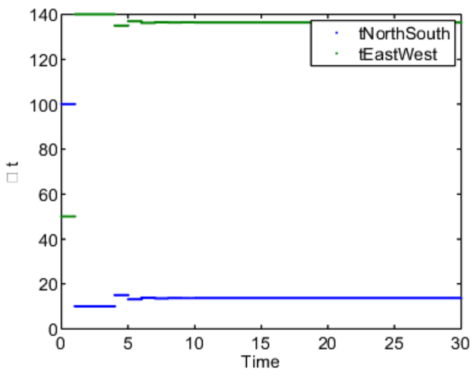

Figure 15: The evolution of the green phases for the second use case *main and secondary road* with the second implementation of the controller.

case, see Figure 16.

3.6 System Integration and Test

In the last phase of the system development, we integrate the validated control algorithm in an evaluation board (MicroController with 80 MHz, 512KB Flash, 32KB RAM, USB) and development environment MPLAB Version 8.84 from Microchip Technology Inc., see Figure 17.

The presented approach differs from the method described in [8], where the control system is executed part on a PC, and part on a microcontroller board.

The relevant code fragment of the from Dymola produced file `dsmodel.c` can easily be identified for this specific controll development and integrated in the environment of the microcontroller code. This transformation can be performed automatically by a phyton script.

The Processor-in-the-loop simulations reflect in detail the observed Model-in-the-loop results.

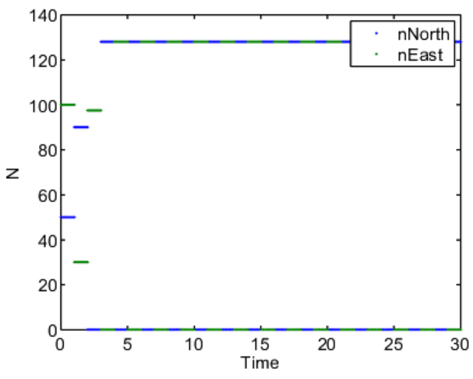

Figure 16: The evolution of the vehicle numbers for the first use case with the second controller, compare with Figure 9.

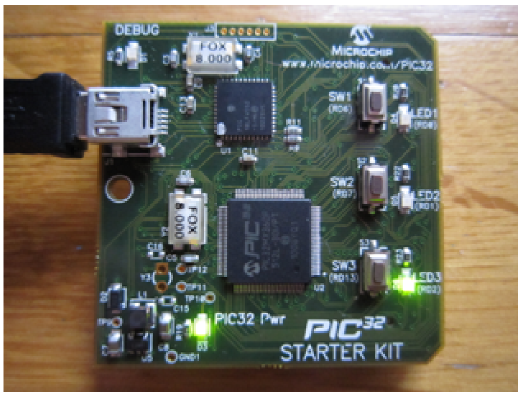

Figure 17: The PIC32 Starter KIT for the Processor-in-the-loop simulations showing the green LED representing the green traffic light.

4 Conclusion

The application of the virtual integration has many advantages because it allows the observation of the behavior of a fully integrated system in an early development phase. Realistic tests in the early phase of development by virtual integration enables comprehensive evaluation of the interaction of a) functions, b) components, c) tools, and d) decision makers and allows a seamless, continuous development process. The method virtual integration allows therefore integration of new technologies and domains.

The following questions arises: How can we systematically identify other development-related interactions? This remains for future work.

References

[1] Adolf-Peter Bröhl and Wolfgang Dröschel. *Das V- Modell. Der Standard in der Softwareentwicklung mit Praxisleitfaden*. Oldenbourg R. Verlag, September 1995. ISBN 978-3-348622-207-4.

[2] C. Höger et al. Homepage Modelica3D, 2012. [Online; Status 24 May 2012].

[3] C. Höger et al. Modelica3D - Platform Independent Simulation Visualization (submitted). In *Modelica Conferene 2012 & Conference Proceedings*.

[4] R. Haberfellner, Olivier L. de Weck, E. Fricke, and S. Vössner. *Systems Engineering – Grundlagen und Anwendungen*. Orell Füssli Verlag, Zurich, 12th edition edition, January 2012. ISBN 978-3-85743-998-8.

[5] Thomas Huckle and Stefan-Alexander Schneider. *Numerische Methoden: Eine Einführung für Informatiker, Naturwissenschaftler, Ingenieure und Mathematiker*. Springer, 2006.

[6] IABG. V-Modell, 2004. [Online; Stand 24. Mai 2012].

[7] Andreas Maier and Stefan-Alexander Schneider. Analyse des Einflusses der Co-Simulation bei der Modellintegration. *Tagungsband ASIM 2011*, 2011. ISBN 978-3-89967-733-1.

[8] Marco Bonvinia, Filippo Donidab, Alberto Leva. Modelica as a design tool for hardware-in-the-loop simulation. Technical report, Dipartimento di Elettronica e Informazione, Politecnico di Milano, 2009.

[9] Richard Harwell. Systems Engineering, A Way of Thinking, A Way of Doing Business, Enabling Organized Transition from Need to Product, 1997. [Online; August 1997].

[10] Stefan-Alexander Schneider, B. Schick, and H. Palm. Virtualization, Integration and Simulation in the Context of Vehicle Systems Engineering. In *Embedded World 2012 Exhibition & Conference Proceedings*. Weka Fachmedien, 2012.

[11] Stefan-Alexander Schneider, B. Thiele, and P. Mai. A Modelica Sub- and Superset for Safety-Relevant Control Applications (accepted). In *Modelica Conferene 2012 & Conference Proceedings*.

[12] Tim Weilkiens. Die rolle des systems-engineerings.

[13] Wikipedia. LISA+ — Wikipedia, Die freie Enzyklopädie, 2011. [Online; Stand 28. März 2012].

[14] Wikipedia. Signalzeitenplan — Wikipedia, Die freie Enzyklopädie, 2011. [Online; Stand 28. März 2012].

[15] Wikipedia. Systems Engineering — Wikipedia, the free encyclopedia, 2012. [Online; Status 13 May 2012].

[16] Wikipedia. V-Modell — Wikipedia, Die freie Enzyklopädie, 2012. [Online; Stand 29. März 2012].

[17] Wikipedia. V-Modell — Wikipedia, the free encyclopedia, 2012. [Online; Status 18. Juni 2012].

Acknowledgments

The authers hereby thank C. Höger from TU Berlin for the animation of the intersection in Figure 10 with Modelica3D and last but not least B. Thiele from DLR for the modeling of the controll algorithms, e.g., in Figure 6 using the library SafeDiscreteControl, see [11].