E-COSM - Rencontres Scientifiques de l'IFP - 2-4 Octobre 2006, Proceedings, pp. 33-40

Rapid Control Prototyping with Dymola and Matlab for a Model Predictive Control for the air path of a boosted diesel engine

K. Hoffmann[1*], F. Heßeler[1], D. Abel[1]

1 Institute for Automatic Control (IRT) at RWTH Aachen University, Steinbachstraße 54, 52074 Aachen – Germany
e-mail: K.Hoffmann@irt.rwth-aachen.de, F.Hesseler@irt.rwth-aachen.de

** Corresponding author*

Abstract — The contribution at hand gives an insight into a tool chain for Model Predictive Control (MPC) which is capable of solving diverse problems in the field of closed loop control of internal combustion engines. This type of control includes a model which in our case is physically motivated. The advantages of these physical process models as well as the possibility to easily implement extensions into this type of model will be presented. Based on a simulation model of the plant, which also is used as a test environment for the controller, redundant work is drastically reduced. The developed tool chain is presented, which reuses the process models in the most effective way but also provides the possibility to handle the most complex forms of process description.

INTRODUCTION

The legislation-impelled progress in development of combustion engines has lead to modern boosted diesel engines which from a controlling point of view are a highly complex multiple coupled systems. Driven by the contrary wish for more power on the one hand but nevertheless exhaust legislation becoming stricter and fuel consumption more expensive on the other hand the research on modern diesel engines is commonly encouraged. These modern engines not only consist of the motor block, crank and valve train but additionally multiple components which's main requirement are to increase power and/or to reduce emissions but which also interact. Therefore their coexistence and cooperation has to be optimized to achieve the best operation condition and a closed loop control is of urgent need.

1 CONTROL PROBLEM AIR PATH

One typical example for these contrary aims but also interferences is the advanced air path set up of turbocharger with variable geometry turbine (VGT), exhaust gas recirculation (EGR) and – becoming more and more important – exhaust aftertreatment. This could mean a particulate filter as well as any NO_x-reducing sub-assembly. An overview of a possible configuration of a modern heavy-duty diesel and its exhaust path with aftertreatment components is shown in Figure 1.

Obviously a noninteracting control would ideally decouple the mentioned type of problem, yet the coupled control of at least two variables was already successfully validated for the simultaneous control of the contrary tasks boost pressure and EGR-rate [1]-[4].

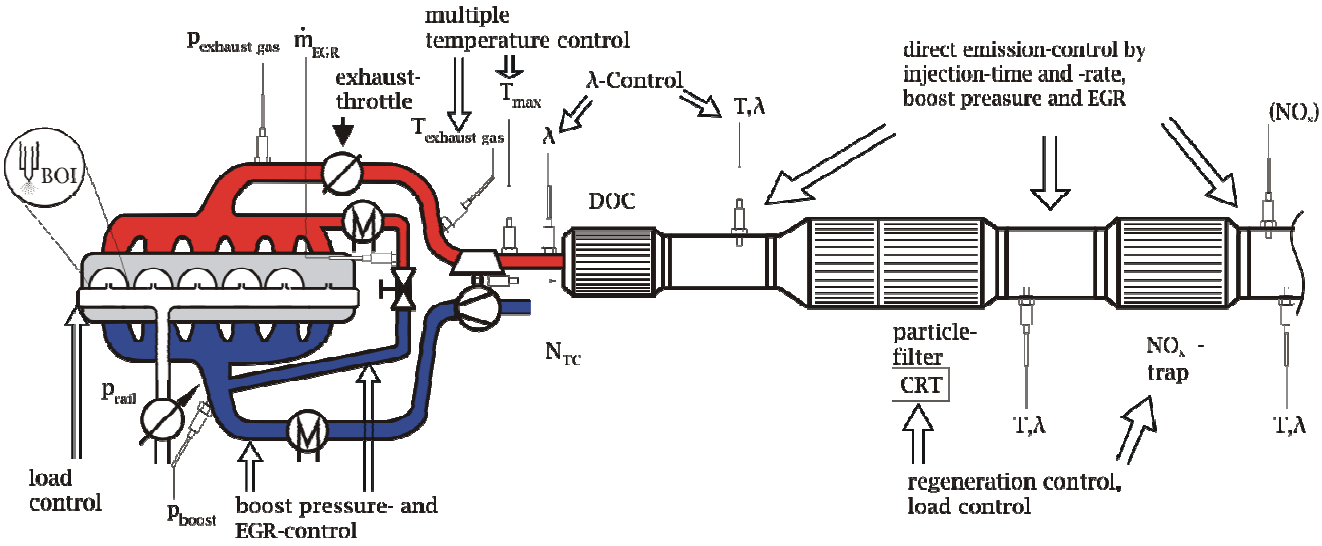

Figure 1:
Control tasks at a diesel engine and its air path with aftertreatment

Model based control patterns therefore offer the chance to improve the closed loop control by using process knowledge.

To set up and test controllers in a simulation-based development environment is broadly accepted since several years now. Often the tool chain suffers from a leakage: For configuring the controller a relatively complex, often physically based modeling is arranged.

In contrast to this modeling effort the implemented models used in the controller itself – if any at all – mostly are very simple approaches, obtained e.g. from a parametric identification. Consequently this means a complete new modeling coexisting to the simulation model without any benefits from or "recycling" of the simulation model besides the in- and output variables. Moreover the total expenses for configuring such models obtained from measuring the system's responses to certain inputs grow extremely with increasing complexity of the complete system. If opposed to this proceeding a model adapted from physical knowledge is also used in the controller all describing variables are completely available. From these only the ones of interest have to be selected as system output by adapting the state space matrix C.

In the following a development environment is introduced, which allows the building of a Model Predictive Controller based on physically motivated models. Especially the realization of the modeling is to be supported by the environment. Starting from the description of simple partial systems the total model is set up avoiding laborious manipulations of formulas.

Moreover the existing system understanding is ideally shared by the described procedure by partly implementing the same approaches in the controller, drained off the plant model. The plant model differs from the model used in the controller mainly by the implemented detail level, that is in the case of the example the number of volumes and pipes.

2 DEVELOPMENT ENVIRONMENT

The kern of the development environment is set up in Matlab/Simulink (Figure 2). Here exists a detailed commented user interface for the test management and a comfortable environment for the interpretation of the results. The diesel engine itself was modeled in Dymola (chapter 2.1) and inserted into the Simulink surrounding, because Simulink has several advantages for building controllers but Dymola has conveniences in modeling.

The controller was implemented as C-code s-function. For demonstrating the possible application the controller was implemented on an automotive-typical rapid control prototyping (RCP) electronic control unit (ECU) and tested in a Hardware-in-the-Loop-Test (HIL), compare chapter 2.2.

The first part of the proposed tool chain is the mathematical description of the modeled process itself. Known formulas are used to define mathematical equations with a number of free parameters. These free parameters are used to fit the describing formulas to measured data. This is done automatically using one of Matlab's optimization-routines. The result is a set of formulas describing the modeled system.

2.1 Realization of the controlled plant in Dymola

For the realization of the controlled process the program Dymola was chosen, which is based on Modelica, an object-oriented language for physical modeling developed by the Modelica Association. As there is no directional use of equations in Dymola, it is very suitable for the modeling of complex physical systems [7]. Therefore the user can leave the necessary steps of manipulating the equations to the program. A physical system often is described by a set of differential equations. Using Dymola one does not have to reformulate this set for one certain variable to be the calculated value and the other variables to be the inputs. The program only needs the set of equations and does the reformulation automatically. As an example in Dymola the exact same model can be used as an electric motor or as a generator. The only difference would be the definition of the system's input, which would be a power supply in the first, a momentum in the second case. Figure 3 shows the implementation of the diesel engine in Dymola. A fragment of the describing Code in Modelica language is shown in Figure 4.

The data exchange between the single elements is done simultaneously via connectors for any amount of variables, which are separated in flow- and potential variables [7].

In combination with the possibilities of the inheritance of modules the best conditions are given for an easy extension of the models. As an example the modeling shall regard the variables pressure, temperature and mass flow. The two-gas-pin in Figure 4 is a base class of all components and supplies two gas interfaces. Only by modifying the connector the structural preconditions for an allowance of the gas composition in all elements is given.

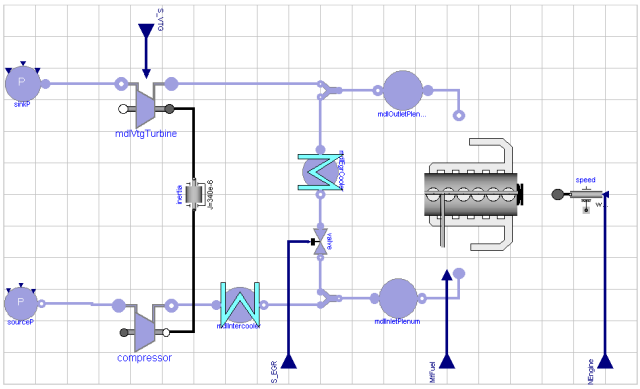

Figure 3:
Diesel engine with VGT-turbine and EGR in Dymola

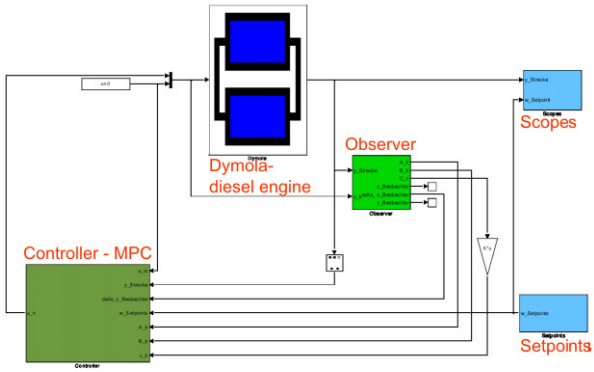

Figure 2:
Top layer of the development environment with Simulink as common basis

Also the change of the systems topology from a single to a bi-turbocharger is possible only by adding an adequately parameterized second turbocharger without the need to change any parts of the rest of the model or introducing new interfaces.

```
model Volumen;
    //connectors
    extends two_gas_pin;
    //parameter snd local variables
    parameter Real V;
    [...]
equation
    //preasure from 1. theorem
    p=pin1.p;
    der(p)=(kappa*R/V)*(pin1.MF*pin1.T+
pin2.MF*pin2.T);
    //massbalance
    der(m) = (pin1.MF + pin2.MF);
    //ideal gasequation
    p*V = m*R*T;
end Volumen;
```

Figure 4:
Modelica code part of a volume

The fundamental idea of Dymola is the combination of graphics and model code, which eases the modeling by full graphical support, but still models received from third parties remain transparent as all equations are visible in the code. In contrast to that Simulink offers the (strictly signal-oriented) s-function for this purpose as user-defined block, but an access to the overall code of the whole Simulink model is not possible. Nevertheless it is possible to protect ones know-how by hiding code and therefore securing knowledge in Dymola by encrypting the Modelica code

using Dymola's model management package as well as creating a Simulink s-function from a Dymola model.

Furthermore the handling of the models differs a lot between Dymola and Simulink which is caused by the different way of modeling. Dymola's object-oriented approach leads to a structure which resembles the real components much more than Simulink models do. This fact facilitates an intuitive way of modeling.

Additionally the possibility of implementing the equations in any formulation supports a generalization, as even known directional correlations do not have to be modeled. This supports the user, as the manipulation of formulas becomes unnecessary for modeling with Dymola.

By the symbolic analysis and the analytic reformulation of the model before the simulation remarkable advantages in calculating time and numerical reliability can be gained. Especially algebraic loops can be handled very easily as they are resolved by the program.

The completed Dymola model is integrated into the Simulink environment by the Dymola-Simulink-interface, which comes along with Dymola. During the operating-time a precompiled model is simulated. For development purposes the direct access to the Dymola model still is available.

2.2 Hardware implementation

Especially in the field of research on control mechanisms for internal combustion engines (ICE) it is extremely relevant to prove the applicability of the controller code on an ECU. Therefore it is inevitable to perform some kind of test using the real ICE or less dangerous a real-time model and set up a HIL-test.

Before testing code on the hardware often a Model-in-the-Loop-Test (MIL) is performed, in which the controller algorithm is run controlling a software model of the plant as realized here in the development environment. In the case presented the HIL-test was chosen to prove the possibility to realize this numerically demanding MPC in a real RCP application. Here the hardware from dSpace was chosen to perform the test, because there exists an easy-to-handle connection from Matlab to hardware-specific code by using its Real-Time Workshop to load the code to the hardware, compare Figure 5. With this last step the tool chain from the model formulation to the implemented C-code is completed.

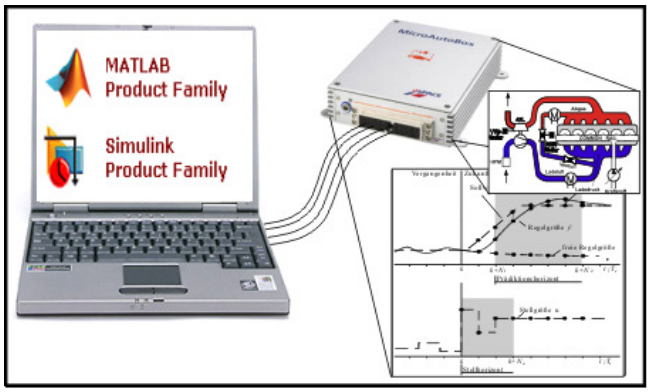

Figure 5:
Code upload to the specific hardware

As simulation platform the dSpace DS1005 system was chosen, which is capable to calculate the developed Dymola model in real time at a sample time of one millisecond. The controller algorithm was calculated by a dSpace DS1006 system at a sample time of twenty milliseconds. Another DS1005 system was used to log the data of both systems, controller and HIL diesel engine model. Figure 6 shows the structural setup of the performed HIL test. It has to be admitted that the chosen hardware offers much more calculating power than a series hardware does. Nevertheless the designed controller does have the potential to be reduced and mathematically optimized unless it also works on a series ECU, which become more and more powerful, too.

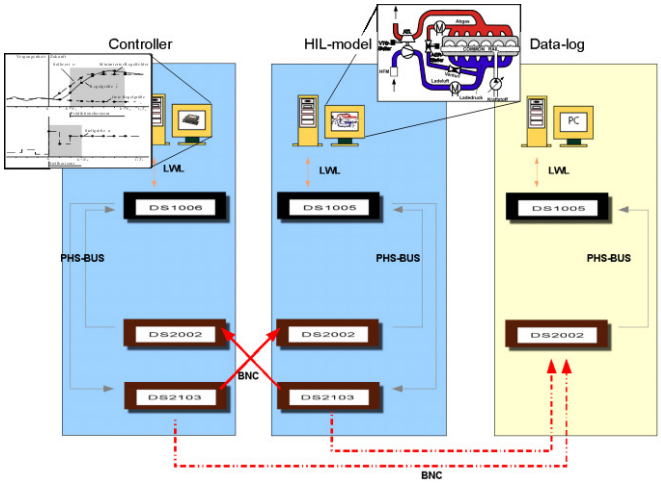

Figure 6:
Structure of the performed HIL-test

3 CLOSED LOOP CONTROL

In the following chapter the control theory of the realized type of controller will be explained.

3.1 Model Predictive Control

Model Predictive Control (MPC) uses an internal model of the controlled plant for calculating the optimal value for the actuating variable with regard to the plant's future behavior. For this purpose the so called 'cost-function' J is minimized, which is used to evaluate the discrepancy between predicted system behavior and the trajectory of setpoints in a certain time-frame. This onward-moving time slot is defined by the lower and upper cost horizon N_1 and N_2 (Figure 7).

Additionally changes of the manipulated variable or its absolute deflection can be considered by adding a respective term to the cost-function. In this manner any additional mathematically describable criterions can be integrated in the cost-function. Changes of the manipulated variable often are tolerated only in a quite narrow timeframe – the control horizon N_u – to limit the complexity of the optimization problem and therefore the calculating effort. In the basic version for a one-dimensional problem this type of cost-function is represented by (1).

$$J = \lambda \cdot \sum_{N1}^{N2} (w - \hat{y})^2 + \mu \cdot \sum_{1}^{Nu} \Delta u^2 \qquad (1)$$

J	cost-criterion
w	trajectory of setpoints
$\hat{y}$	predicted system output
Δu	change of the manipulated variable
λ, μ	weighting factors
N_1, N_2, N_u	time horizons

By adjusting the weighting factors λ and μ more emphasis can be put on the deviation between setpoint and the plant's predicted output or the change of the manipulated variable. Here the trajectory of setpoints w is assumed to be known. The predicted controlled variable $\hat{y}$ has to be formulated in dependency on the change of the manipulated variable Δu by using the representing model of the plant. The circumflex in (1) hints to the fact, that this future output of the system is estimated. The time horizons N_1, N_2, N_u and the weighting factors λ and μ here are selectable but fix tuning parameters for the controller's behavior. What still is left is the formulation of the cost-function in dependency on Δu, which then is to be minimized by appropriate algorithms.

In the case of more than only one input and one output of the system (single-input-single-output, SISO) the equation is reformulated using matrices, which in our case leads to the weighting-matrices $\boldsymbol{\Gamma}$ and $\boldsymbol{\Lambda}$ instead of the scalar factors μ and λ. In this multi-input-multi-output (MIMO) case these matrices can also be used for weighting the corresponding values among each other. Details on this topic can be found in [4].

Proceeding for Model Predictive Control

1. detect actual state of the controlled system
2. calculate free system response (use model for the prediction)
3. receive actual trajectory of setpoints
4. lay down cost-function for next prediction step (model for prediction with regard to horizons)
5. minimize cost-function conditioned by control horizon (optimization)
6. output first element of the calculated necessary series of changes of the manipulated variable
7. new sample step: receding of the horizon
8. proceed with step 1 with the new calculated state of the system x_k

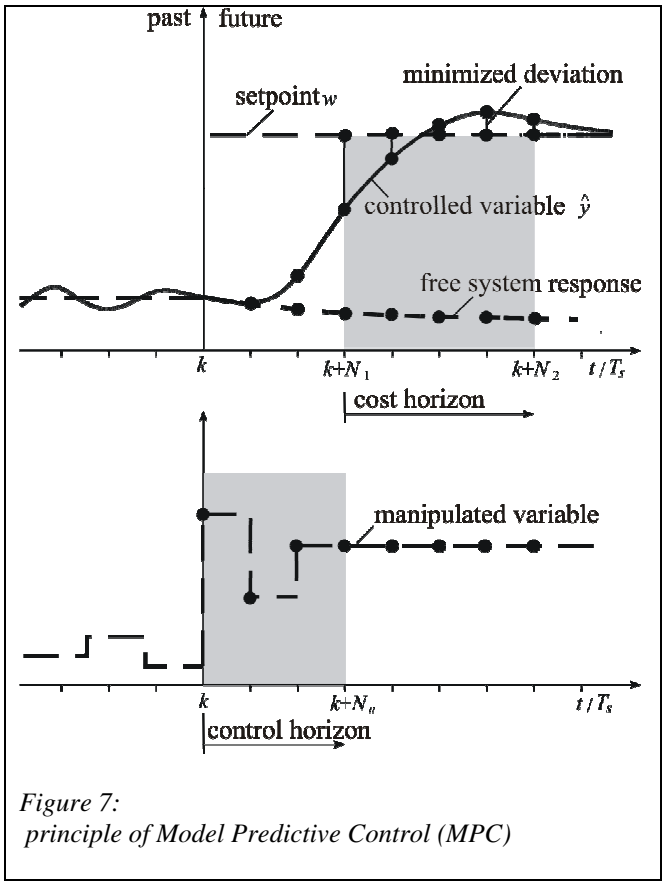

Figure 7:
principle of Model Predictive Control (MPC)

For minimizing the cost-function one has to choose from a huge variety of possible ways. If a linear model description of the regarded process is used, the cost-function (1) becomes a strictly convex expression, whose global minimum can be guaranteed. In the unconstrained case the solution of the problem can be found analytically. If constraints are to be considered a suitable optimizing procedure has to be found. If the constraints can be formulated as linear inequations as in the considered example, the problem can be solved by a quadratic program for which standard algorithms are available. For the problem at hand the minimum and maximum values of the manipulated variable, of its change and of the system's output were regarded.

3.2 Realization and application of the controller

If non-linear models of systems are used the cost-function J (1) becomes arbitrary complex. For this case so far no proven and applicable methods exist for securely finding the global optimum or which are able to guarantee global convergence of the solution. For several special cases exist solutions, like for example for the use of a non-linear state space description of the process. The calculating effort extremely rises for these optimization methods. In respect of the available calculating power, no approaches with non-linear system descriptions were pursued.

By gradually linearizing the non-linear state space model for the controller at every sample step and the use of an extended Kalman-filter in the observer a structure of the whole system results as sketched in Figure 8.

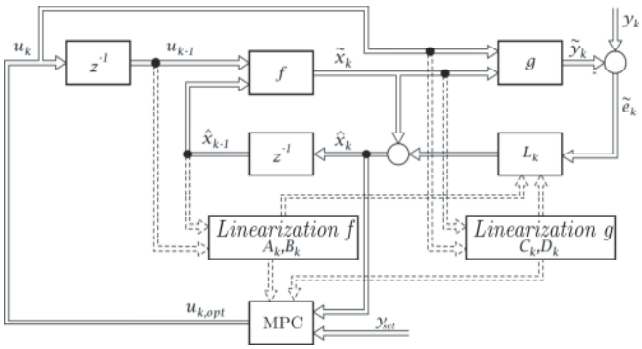

Figure 8:
Structure of the implemented closed loop control

The subsystems to be developed are:

- Non-linear state space model of the plant (*f* and *g* in Figure 8)
- Linearized state space model of the plant (matrices A_k to D_k in Figure 8)
- Kalman-filter algorithm L_k
- Optimization - *MPC*

The first two steps mentioned in any case are system-related and have to be performed for every new modeled plant. For the last to steps has to be kept in mind that these functions are generic and do not have to be reproduced. Details on the used algorithms can be found in [8].

For automating the necessary steps of development for the model-specific components as far as possible the formula manipulating program Maple is used. The formulas used to set up the Maple-environment are the

same as the ones used for the Dymola-model. The Dymola-model only becomes more detailed than the one used in the MPC by including more volumes and the use of more detailed thermodynamical knowledge. At this point the reuse of the found formulas avoids redundant work.

The implementation of the interrelationships is done by Maple's automatic formula manipulation engine and C-code generation. The setting up of the formulas is done in any (choosable) formulation in the correct mathematical sequence in the Maple worksheet. The manipulation of the implemented formulas to the desired form necessary to describe the state space requires the user only to define the state and the input variables. The necessary steps of manipulation are done automatically. The whole linearization as well as the generation of the C-code are performed automatically. The result is a C s-function which is ready for compilation in Matlab, Figure 9.

Maple – modelequations

Mapping of Vectors

```
> x :=<T_IC|M_IC|T_EC|M_EC|T_OP|M_OP|
    T_IP|M_IP|N_TC|s_EGR|s_VTG>:
u :=<d_s_VTG|d_s_EGR|MF_FUEL |N_ENG>:
```

Matrix A: Linearization of f(x,u)

```
> A := Matrix(L_x,L_x):
  for m from 1 by 1 to L_x do
    for n from 1 by 1 to L_x do
      A[m,n] := diff(f(x,u)[m],x[n]):
    end do:
  end do:
> A := convert(A,array):
```

C - CODE MATRIX A

```
> code_A :=
  C(A,optimize=true,deducetypes=false,
  defaulttype=numeric,resultname="A",
  output=string):;
```

C-Code – s-function

```
static void mdlOutputs([…]){
[…]
// deklaration of variables-,
// created by Maple
#include "Variablen.h"


//matrices, created by Maple
#include "ZSR_A.txt"
#include "ZSR_B.txt"
#include "ZSR_C.txt"
#include "ZSR_D.txt"
// output matrix A
     for (ii=0; ii<L_x; ii++) {

          for(jj=0; jj<L_x; jj++){
                  y3[(ii+jj*L_x)]      =
A[ii][jj];
  }
     }
//Output of other matrices
[…]
}
```

Figure 9:
Maple and wrapping C s-function

4 COMPLETE TOOL CHAIN AND RESUME

By the presented work a complete tool chain was demonstrated, which is capable to automate all necessary steps as far as possible. These required steps are done only once and their results are reused in a most efficient way by avoiding redundant modeling work. The presented closed loop control scheme based on a generally linearized state space model of the plant offers a most comfortable way to benefit from the high potential of Model Predictive Control.

This leads to a shorter development time for the controller although the models used in the controller are definitive more complex than in usual controller if any. This fact is caused by the reuse of existing model approaches for the components. Starting from the description of the single component it is therefore possible to automate the application up to the hardware-in-the-loop-test by the assistance of the presented tool chain.

Changes in the controlled or the actuating variable, of single components or of the topology of the whole system as well as extensions can be integrated very systematically and automated. The single parts of the Rapid Control Prototyping (RCP) environment interact with each other as shown in the V-model in Figure 10.

The main challenge concerning this tool chain is for the moment the successful implementation and use of the MPC in the real air path of an application. This challenge will be faced in a major project, which will deal with the Modelbased Predictive Control of a series diesel-engine operated in HCCI mode. As a part of the whole HCCI-controller the air path will be considered. This project is planned as a part of the SFB686 with the title "Modelbased Predictive Control of the homogenous low temperature combustion" and is financed by the "Deutsche Forschungs-gemeinschaft". First results will be presented as soon as they are available.

As this tool chain has not been used yet in an OEM's development process of an engine control unit neither the actual costs nor the benefits of its use for industry can be enumerated.

The control of the coupled problems boost pressure and exhaust gas recirculation often simply neglect the linkage between these two tasks. In wide areas of the engine's operation map only one of the two controlled variables really is controlled in a closed-loop while the other is controlled open loop using a look-up table. At higher loads the closed loop control only regards the boost-pressure, at lower loads only the EGR mass. A combined control only takes place at medium rpms and medium loads and often is not decoupled. Normally two PID-controllers are used parallel. Therefore the control-task often is solved only inadequately and the two controllers are hard to apply. Using an MPC to ensure a decoupled control leads to lower emissions and better fuel consumption as the boost pressure can be optimized in wider range.

For future highly integrated control tasks in the field of diesel engines with coupled systems as turbocharger, exhaust gas recirculation and exhaust gas aftertreatment the presented attempt represents a splendid base for further research as well on the controller itself as on the controlled diesel engine. For the last point the fact is very giving, that a change in the system does not necessarily mean a re-measurement of the whole system's behavior and therefore a reduction in research costs.

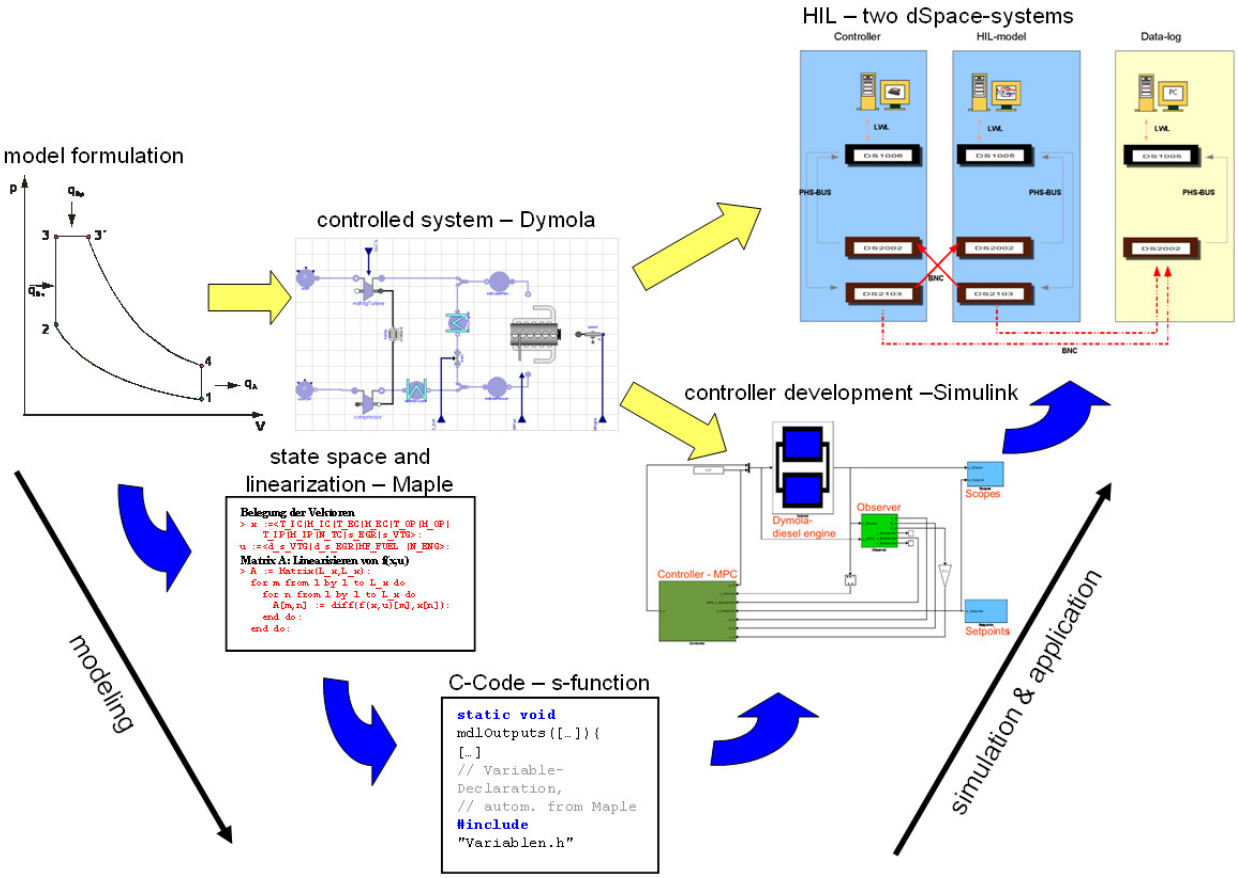

Figure 10:
V-model for the building process of the controller

REFERENCES

1. Rückert, J.; Kinoo, B.; Krüger, M.; Schloßer, A.; Rake, H.; Pischinger, S.: Simultane Regelung von Ladedruck und AGR-Rate beim Pkw-Dieselmotor, Motortechnische Zeitschrift 62, 2001, 11, 956-965
2. Pfeifer; Smeets, M.; Herrmann, H.-O.; Tomazic, D.; Richert, F.; Schloßer, A.: A New Approach to Boost Pressure and EGR Rate Control Development for HD Truck Engines with VGT, SAE World Congress 2002, SAE 2002-01-0964
3. Rückert, J.; Richert, F.; Schlosser, A.; Abel, D.; Herrmann, O.; Pfeifer, A.; Pischinger, S.; Konzepte zur Regelung von Ladedruck und AGR-Rate beim Nutzfahrzeug-Dieselmotor, atp - Automatisierungstechnische Praxis, 45 (2003) Heft 7
4. Rückert, J: Modellgestützte Regelung von Ladedruck und Abgasrückführrate beim Dieselmotor, Dissertation, RWTH Aachen, 2005
5. Maciejowski; J.: Predictive Control with Constraints, Addison Wesley Longman Dezember 2001 ISBN: 0201398230
6. Elmqvist, H.; Mattson, S.E.: An Introduction to the Physical Modeling Language Modelica, Proceedings of the 9th European Simulation Symposium (ESS97), 1997
7. Richert, F.; Rückert, J.; Schloßer A.: Vergleich von Modelica und Matlab anhand der Modellbildung eines Dieselmotors, at-Automatisierungstechnik 51 (2003) Nr. 6
8. Richert, F: Objektorientierte Modellbildung und Nichtlineare Prädiktive Regelung von Dieselmotoren, Dissertation, RWTH Aachen, 2006