The Flap Controller

AVACS H3

[1] Carl von Ossietzky Universität Oldenburg
Ammerländer Heerstraße 114-118, 26111 Oldenburg, Germany
[2] OFFIS e.V., Escherweg 2, 26121 Oldenburg, Germany
[3] Albert-Ludwigs-Universität Freiburg
Georges-Köhler-Allee 51, 79110 Freiburg, Germany
[4] Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany

Abstract. We first give a general descriptions of the Flap Control system and its Statemate model, then we discuss the derived HLang models as benchmarks for our model checking approach developed within the context of AVACS H3 project. Our approach combines SAT, And-Inverts Graphs, and first-order reasoing for automatic verification of hybrid systems with large discrete state-space.

1 General description

We use as a benchmark from the Avionics a simplified version a flap control system, reported in [1]. This presentation additionally builds on notes of a lecture by Georg Mai, Airbus Industries, given as part on a course on "Design of Safety Critical Embedded Systems" at the Carl von Ossietzky Universität Oldenburg, April 2000.

Flaps are movable surfaces situated on the inboard trailing edge of the wing (see Figure 1). Flaps can be extended and lower towards predefined positions (number of positions dependent on aircraft type), under the primary control of the pilot. It is well known that the lift-coefficient of a wing depends on its camber. Deflecting the trailing edge downwards can change the wing camber. Lowering the flaps thus increases lift and drag, allowing a slower approach during landing.

Flaps are extended and retracted through the Flap Power Control Unit (FPCU); the FPCU links to segmented shafts in each wing, and controls their rotation, which then mechanically translates into extension and retraction of the Flaps. See Figure 2 for a drawing of the mechanical subsystem.

For example, on the Airbus 340, the Flaps can be in five configurations (lever positions). The pilot can select the configuration of the Flaps manually by means of the Slat/Flap Lever. The Command Sensor Unit (CSU) converts a selection into a set of discrete electrical signals. These signals are dispatched to their respective channels within a Slat-Flap Control Computer (SFCC) (See Figure 3). The new commands are "validated" within each computer, as elaborated below. Each computer independently signals the associated hydraulic solenoid valves on the Power Control Unit (PCU) to release the power off brake (by re-setting input

Fig. 1. The Flap Controller

P of its valve block) and to initiate either a retraction (input R) or extraction (input E) using the hydraulic motors to drive the Flaps to the new position. When the PCU position and the demanded position are within a specified threshold range ahead of target position, the system will slow down (input H is low), until the PCU achieves the actual demanded position; this basic control loop is realized within the Flap Control Computer, by maintaining the intended position and comparing it with the position monitored by the Feedback Position Unit Sensor. At this point the pressure-off brake is applied (by setting the input P of its valve block), and the PCU is shut down. This concludes the so-called normal drive sequence. The controller has to take into account failures such as low pressure and power interrupts, as well as turnaround commands.

The system-architecture is following the Duo-Duo-Duplex design paradigm, providing two separate computers per aircraft, two separate HW channels for Flap and Slat each, and two dissimilar HW lanes per channel, and triple redundancy for the hydraulic pressure lanes. Two dissimilar (disagreed) commands are necessary to generate an error. Any discrepancy of lane computations stops and freezes the defected system. Continued operation in case of such failures is ensured through the other SFCC's channel, but with half drive speed at the torque shaft.

In the following we will focus on *one channel* and the *Flap subsystem*. The Flap Controller also implements an automatic mode of operation, the so-called *auto-relief function*, which will retract the Flaps to relief Flaploads. In auto-relief mode, the Flap Controller constantly monitors aircraft speed and aircraft attack angle to assess, whether the load for the pilot position is too high. The Flaps are retracted from the extended position to the next lower lever setting, if the aircraft speed exceeds a certain threshold speed for the selected configuration for more than a specified limit. The selected position will be automatically commanded again if the aircraft speed falls below the threshold speed. The system is designed in a way that makes critical events extremely improbable. Critical events that may significantly impact aircraft safety include the powered runaway of the Flaps, inadvertent Flap retraction due to auto-command, overspeed and asymmetric extension. System protection against the events is achieved by sensing and monitoring of PCU and Flap positions and rate of change of posi-

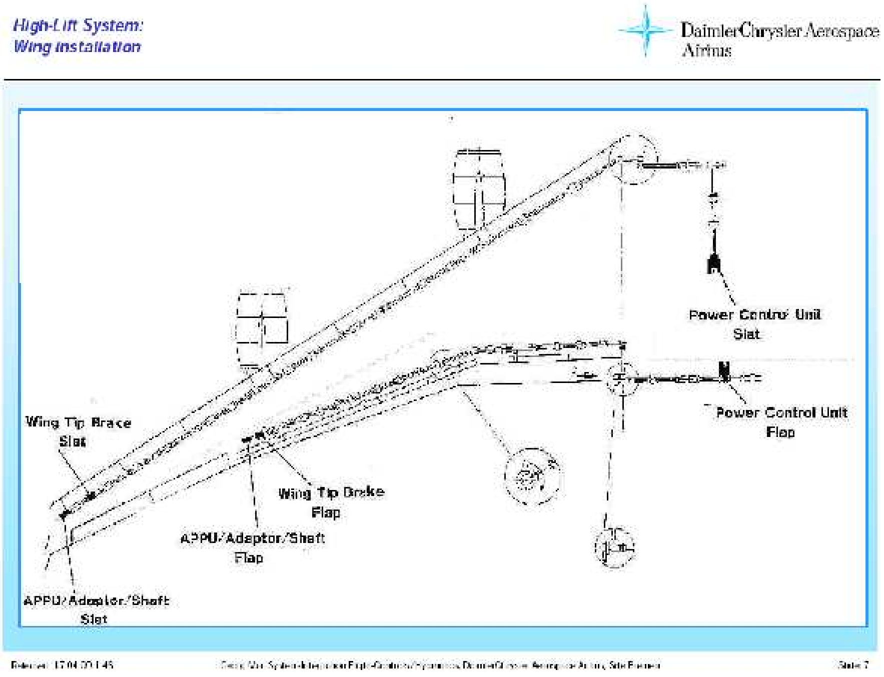

Fig. 2. The Flap Controller: The mechanical subsystem

tions. Whenever a deviation is detected, both the pressure-off-brake as well as the wing-tip brakes are applied. This functionality is implemented within the Flap Controller.

A reference model of the Flap Controller covering the above functionality has been developed using the Statemate product from Telelogic AB. Figure 4 below shows the interfaces and subcomponents of the Flap Controller on an abstract level. Inputs shown on the left side are (from top to bottom):

- The Flap lever position as signaled by the pilot (a discrete input);
- The consolidated aircraft speed and attack-angle (floats);
- Status information on the hydraulic subsystem;
- The position of the flap as measured at the gearshift within the FPCU;
- The position of the flap as measured at the wing-tip (for asymmetry detection).

The CHK-Lever sub-controller performs plausibility checks on the commanded lever positions. The sub-controller `Auto_CMD` and `CHK POS_THRESH` are responsible for realizing the auto-relief function. The Monitoring component combines sub-controllers for each of the critical conditions (powered runaway of the Flaps, inadvertent Flap retraction due to auto-command, over-speed and asymmetric extension). The actual control commands to the FPCU are issued by the sub-controller called `MOVEMENT_CMD`. The controller outputs shown on the right hand

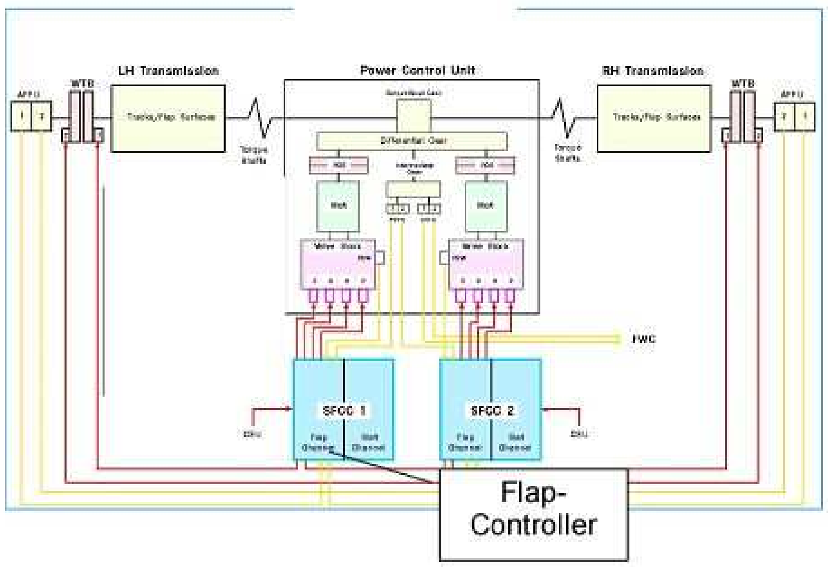

Fig. 3. The Flap Controller: SFCCs

side are controlling the E, R, H, and P inputs of the FPCU explained above, the wing-tip bake, as well as the generation of status information to the pilot.

The Statemate model of the Flap Controller consists of 30 charts (Activity Charts and StateCharts); most of them are instances of generic chart definitions. The state space of the model is spanned by

- 143 states (in the sense of StateCharts, including hierarchical states, i.e. AND/OR states);
- 33 variables of type real;
- 14 variables with enumeration types;
- 7 variables of type integer;
- 38 conditions (in the sense of Statemate);
- 12 events.

In addition, the model uses 9 constants of type real, 10 constants of type array of constants of type real, one constant of enumeration type, and 17 integer constants.

2 Derived HLang models

For the purpose of benchmarking the verification methods developed within AVACS, we have developed various versions of this reference model and providing translations into the input language HLang. We have in particular developed simple environment models along the directions explained below.

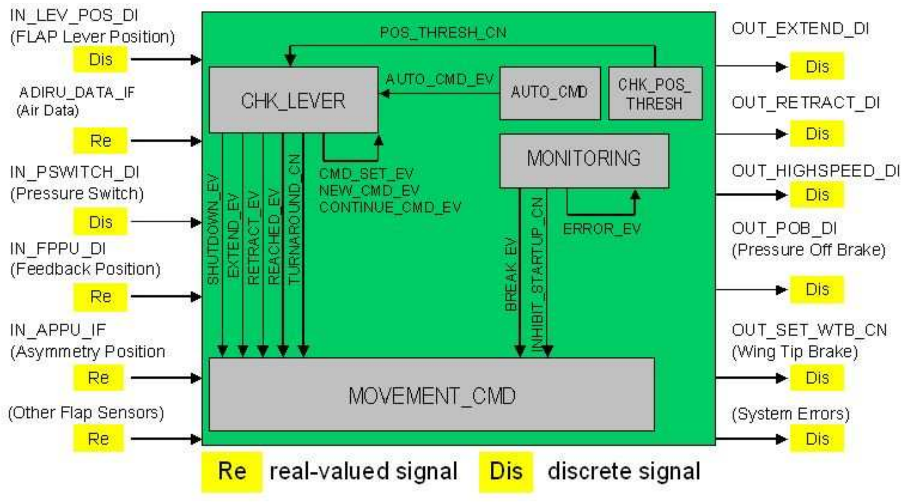

Fig. 4. The Flap Controller: The Statemate model

1. *The Reasonable Pilot.* The pilots can select the configuration of the Flaps manually by means of lever positions. First, by construction, lever positions can only be incremented and decremented by one, an environment assumption made by the Flap Controller, and built into our model of a reasonable pilot. Secondly, pilots by training don't choose flap positions which are strongly incompatible with tolerable loads (unless extreme flight maneuvers are required due to prevent higher hazard levels), and we build this knowledge into our model of a "reasonable" pilot.
2. *Aircraft Model.* The Flap Controller is designed to work within the boundaries dictated by the dynamics of the aircraft. We include a simple model of the aircrafts dynamics, reflecting natural maximal rates of changes in aircraft speed (and angle of attack).
3. *PCU Model.* We provide an abstract model of the FPCU, accepting the FPCU inputs and returning a flap angle (corresponding to the FPPU sensors of the FPCU). Thus the hydraulic subsystem is abstracted away.
4. *The Flap Subsystem.* We focus on the auto-reflief function in the Flap controller. The Flap Controller constantly monitors aircraft speed, whether the load for the pilot commanded lever position is too high. The Flaps are retracted from the extended position to the next lower lever setting, if the aircraft speed exceeds a certain threshold speed for the selected configuration for more than a specified limit. The selected position will be automatically commanded again if the aircraft speed falls below the threshold speed.
5. *Health Monitoring.* We provide a monitoring subsystem for detecting deviations of the aircraft's speed and the actual flap position, and for detecting failures of behavior of the controller and the pilot. Whenever a situation

endangering the flaps is detected, the controller can deviate one more from the input lever position.

Moreover, in our models we only use one HW channel for flaps. More details will be given in the later discussion.

Model structure The system consists of four components: the “reasonable” pilot behavior, the controller, the flap mechanism, and the rest of the aircraft (Figure 5). It controls two continuous variables v (velocity) and f (flap angle), and two discrete variables ℓ (lever position set by the pilot) and c (corrected position, set by the auto-relief function in the controller). For each lever position, there is a pre-defined flap position $\mathit{flap}(\ell)$ and a pre-defined nominal velocity $\mathit{nominal}(\ell)$. as described in the following table:

ℓ	0	1	2	3	4	5	6
$\mathit{flap}(\ell)$	0	15	20	30	40	47	55
$\mathit{nominal}(\ell)$	335	235	220	200	180	165	150

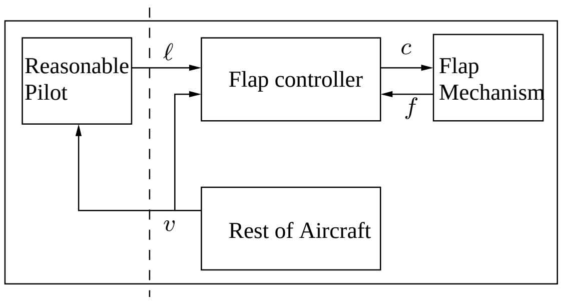

Fig. 5. System components.

For each component in the system, there is one extra health monitoring subcomponent, which monitors the behavior of the pilot and the flap controller, checks whether the measurements of the flap angle and the speed are accurate. The health monitoring subcomponent are triggered by fixed timeouts.

Model details The pilot component in our model ensures reasonable lever positions, by guaranteeing that the lever is at most one notch too high. The “reasonable” pilot will decrease the lever position if the aircraft’s velocity exceeds $\mathit{nominal}(\ell - 1)$. The behavior of the controller depends on both ℓ and v: When the velocity is greater than the nominal max value ($\mathit{nominal}(\ell) + 2.5$ knots), the modification of the pilot behavior is activated ($c = -1$); when the velocity has changed to less than the nominal min value ($\mathit{nominal}(\ell) - 2.5$ knots), the modification is turned off ($c = 0$). (This models the auto-relief function of the controller.) The flap mechanism controls the continuous variable f, and depends on the discrete variable c. It models the mechatronic which adapts the physical flap angle f to the position commanded by c. This is a process which takes time. f has a range from 0 to 55.0. The flap angle may increase or decrease by $\mathit{\Delta}_f = 0.3125$ per 100 ms, or keep as it is. (This is a simplified version of the FPCU, and abstracts the actual physical system.) The rest of the aircraft increases the velocity by 0.5 knots (per 100 ms), within a range from 140.0 to 340.0 knots. (This is a simplified version of the aircraft model.)

The health monitoring subcomponents are triggered by timeouts. Each subcomponent is modeled as a state machine with three states. Initially, the state machine is in the "OK" state, and moves on clock timeout to a "Polling" state, and checks the health of the monitored component. When the component is not providing an answer or answers "Sick", the responsible health monitor goes to a "Failure" state; otherwise, it moves back to the "OK" state. This non-deterministic choice is modeled as discrete inputs in HLang. Currently, the health monitoring system can interact with system in the following way: On diagnosed pilot failures, failures of the controller, or in the situations when the speed or flap angle measurement is found to be inaccurate, one "alarm" is raised and the controller will deviate more from the input lever position, e.g. the modification on the lever position can deviate by one position further, thereby trying to stay on the safe side (retract flaps earlier). (This is a slightly different version of the monitoring system in the previous section.)

Property The invariant property "safe" to establish for the model is the following: "For the current flap setting f, the aircraft's velocity v shall not exceed the nominal velocity $\mathit{nominal}(f)$ plus 7 knots". Whether this requirement holds for our model depends on a "race" between flap retraction and speed increase. The controller is correct, if it initiates flap retraction (by correcting the pilot) early enough.

HLang models From this system, we can make models with different number of lever positions. The initial states can then be defined accordingly: for a given initial lever position ℓ, we set v as $\mathit{nominal}(\ell)$, f as $\mathit{flap}(\ell)$, and $c = \mathbf{ff}$. In the HLang models, there are totally three different modes: $\mathbf{m}_0$, $\mathbf{m}_1$ and $\mathbf{m}_2$. The evolutions of the continuous variables are defined as follows: $\dot{v} = 0.5$ and $\dot{f} = 0.3125$ for mode $\mathbf{m}_0$; $\dot{v} = 0.5$ and $\dot{f} = 0.0$ for mode $\mathbf{m}_1$; and $\dot{v} = 0.5$ and $\dot{f} = -0.3125$ for mode $\mathbf{m}_2$ (of course $\dot{t} = 0.1$ for timer). The range information for v and f is encoded as global constraints. The constraints on the flap position and the aircraft's velocity, such as $\mathit{flap}(\ell)$, $\mathit{nominal}(\ell)$, and $\mathit{nominal}(\ell) \pm 2.5$, together with the periodic timeout to trigger the health monitoring subcomponents, are treated as the boundary conditions for each mode. Health monitoring subcomponents are modeled as automata with three states, The non-deterministic choice whether the monitored component is in a health status is modeled as discrete inputs. Extra discrete variables are needed to store these inputs and to distinguish whether a health monitoring component has finished its transitions. The behavior of the pilot and the controller, the interaction of the health monitoring components and the system constitutes discrete transitions in the model. After the decision of the pilot and the controller, an appropriate mode will be selected.

In summary, in our models, there are three continuous variables: the velocity v, the flap angle f and the timer value t, three modes: $\mathbf{m}_0$, $\mathbf{m}_1$ and $\mathbf{m}_2$. Discrete states of the controller and of the health monitoring system contribute to the discrete state space. We have boolean variables for lever positions ℓ, the modification c set by the auto-relief function, and "alarm" raised by the health monitoring system. Moreover, there are four boolean variable to record the non-deterministic choice for each health monitoring subcomponent, and 12 boolean variables to capture the status of the health monitoring system. The discrete state space contains 2^{20} discrete states. This size is clearly out of reach for hybrid verification tools known from the literature, which do not scale in the discrete dimension, since modes – the only discrete states considered – are represented *explicitly* when performing reachability analysis.

3 Verification results of the HLang continuous models

With AVACS H3 subproject, we have proposed an approach for verification of hybrid systems, which contain large discrete state spaces and simple continuous dynamics given as constants [4] We use AND-Inverter Graphs (AIGs) extended with linear constraints (Lin-AIGs) as symbolic representation of the hybrid state space. Our model checker was able to prove the given safety invariant for the case study in 888.6 CPU seconds. The Lin-AIG representation had a maximum number of 30887 nodes and a maximum number of 80 linear constraints. This result clearly demonstrates that our approach is able to successfully verify hybrid systems including discrete parts with state spaces of considerable sizes. Our experiments were run on an AMD Opteron with 2.6 GHz and 16 GB RAM.

4 Verification results of the Statemate discrete model

For the verification results presented in this section the Statemate model of the Flap Controller System was used. This is possible by using a pre-existing translation of Statemate to an internal representation called SMI [3, 6], to which the ω-Cegar approach [5] can be applied directly. As usual, a preceding cone-of-influence reduction of the model might remove parts of the model if by syntactical analysis it can be determined that they have no impact on the property.

The proofs being checked on the Flap Controller System are reachability checks for invariant violations and certain system states. The verification technique to achieve these results was the ω-Cegar approach in combination with a bounded model-checker (*BMC*). Since the model is too big for a BDD-based model-checker (e.g. *VIS* [2]), the unreachability of certain expressions could not be proved. Instead, the BMC technique showed the unreachability within a certain number of steps being higher than the anticipated diameter of the model. Table 1 shows statistical information of some properties being checked on this model. For a detailed description of the provided values see [5].

Proof	$\|Z\|$	dims	$\|\Sigma\|$	$—C_i\|/\|C_v\|$	$\#it$	$\|\pi\|$	$\|A_C\|$	time
φ_1	2^{240}	4+14	587	54/286	58	$\gtrsim$39	100	44 min $+MC_f$[a]
φ_2	2^{240}	4+14	420	130/109	143	6	138	88 min
φ_3	2^{240}	4+14	393	119/116	133	6	163	160 min
φ_4	2^9	2+4	6	2/2	3	6	4	1 min
φ_5	2^{240}	4+14	0	0/0	0	$\gtrsim$50	2	10 min
φ_6	2^{240}	4+14	6323	497/1345	958	8	1414	55 h

[a] MC_f is the time required for the final model-checker run after all refinement steps are completed. Time depends on bound for proving unreachability of property violation therein (BMC takes 130 min for a bound of 39 steps) or, alternatively, time required for BDD-based modelchecker to prove the expected unreachability (more than days required for this proof).

Table 1. Overview on experimental results. The table contains columns for the number of discrete states inside the cone-of-influence, the number of continuous dimensions (input and state-based) within the cone-of-influence, the number of different regulation laws $|\Sigma|$, the number of generalized initial and invariant conflicts $|C_i|/|C_v|$, $\#it$ is the number of iterations of the abstraction refinement process and $|\pi|$ is the length of the path preceded by the symbol $\gtrsim$ if the process was interrupted at the given length of spurious counterexamples. Finally $|A_C|$ is the number of state-bits of the ω-automaton being used for refinement of the abstracted model and the last column contains the total required time for the overall verification process.

Proofs

Property $\varphi_1 := (\mathbf{AG}(\neg(retract \wedge extend)))$ This Property specifies an invariant of the system, saying that the outputs *retract* and *extend* must never be true at the same time. By using a bounded model-checker, it was proven that within the first 39 steps, the invariant cannot be violated.

Property $\varphi_2 := (\mathbf{EF}(retract))$ This property specifies that it must be possible to set the *retract* signal. The result is a witness path of length 6, showing how this is possible.

Property $\varphi_3 := (\mathbf{EF}(extend))$ This property specifies that it must be possible to set the *extend* signal. The result is a witness path of length 6, showing how this is possible.

Property $\varphi_4 := (\mathbf{EF}(overspeed\_cn))$ Here, the reachability of a certain state named *overspeed_cn* is specified and proven by a witness path of length 6.

Property $\varphi_5 := (\mathbf{AG}(\neg(shutdown\_error \wedge (retract \vee extend)))$ This is another invariant of the system where neither the *retract,* nor the *extend* signal must be set, if a *shutdown_error*-state is active. Within a time of 10 minutes, it is proven that the invariant holds at least for the first 50 steps.

Property $\varphi_6 := (\mathbf{EF}(\mathit{continue\_cmd\_ev}))$ This property specifies that it must be possible to set the *continue_cmd_ev* signal. The result is a witness path of length 8, showing how this is possible.

References

1. M. Bretschneider, H.-J. Holberg, E. Böde, I. Brückner, T. Peikenkamp, and H. Spenke. Model-based safety analysis of a flap control system. In *Proc. 14th Annual INCOSE Symposium*, 2004.
2. R. K. Brayton, G. D. Hachtel, A. Sangiovanni-Vincentelli, F. Somenzi, A. Aziz, S.-T. Cheng, S. Edwards, S. Khatri, Y. Kukimoto, A. Pardo, S. Qadeer, R. K. Ranjan, S. Sarwary, T. R. Shiple, G. Swamy and T. Villa. VIS: a system for verification and synthesis. In *Proc. 8th Conference on Computer Aided Verification*, Lecture Notes in Computer Science 1102, pp. 428-432. Springer, 1996.
3. U. Brockmeyer. Verifikation von STATEMATE Designs. PhD thesis, Universität Oldenburg, 1999.
4. W. Damm, S. Disch, H. Hungar, S. Jacobs, J. Pang, F. Pigorsch, C. Scholl, U. Waldmann, and B. Wirtz. Exact state set representations in the verification of linear hybrid systems with large discrete state-space. In *Proc. 5th Symposium on Automated Technology for Verification and Analysis*, Lecture Notes in Computer Science. Springer, 2007.
5. M. Segelken. Abstraction and counterexample-guided construction of ω-automata for model checking of step-discrete linear hybrid models. In *Proc. 19th Conference on Computer Aided Verification*, Lecture Notes in Computer Science 4590, pp. 433-448. Springer, 2007.
6. G. Wittich. Ein problemorientierter Ansatz zum Nachweis von Realzeiteigenschaften eingebetteter Systeme. PhD thesis, Universität Oldenburg, 1998.