Boderc: Model-based design of high-tech systems

A collaborative research project for multi-disciplinary design analysis of high-tech systems.

Editors:

Maurice Heemels Eindhoven University of Technology
Gerrit Muller Embedded Systems Institute

Publisher:
Embedded Systems Institute, Eindhoven, The Netherlands

Publisher:
Embedded Systems Institute
TU/e Campus, Den Dolech 2
P.O. Box 513, 5600 MB Eindhoven
Eindhoven, The Netherlands

Keywords:
system design; system engineering; high-tech systems; modeling; high-level method; performance; multi-disciplinary

ISBN-13: 978-90-78679-01-1
ISBN-10: 90-78679-01-8

The Boderc project has been executed under the responsibility of the Embedded Systems Institute, and is partially supported by the Netherlands Ministry of Economic Affairs under the Senter TS program.

Contents

1 Introduction 1

2 A design methodology for high-tech systems 11

3 The key driver method 27

4 Threads of reasoning 43

5 Budget-based design 59

6 Effective industrial modeling: The example of Happy Flow 77

7 Heat modeling in copiers 89

8 Modeling of performance 101

9 Virtual printer modeling 115

10 Using stepper motors in printers 129

11 Simulating the environment of embedded software 141

12 Evaluating embedded system architectures 151

13 Model-driven design of real-time systems 161

14 Time-varying delays in control 171

15 Sheet feedback control in a printer paper path 183

16 Event-driven control 195

17 Design trajectory and controller-plant interaction 205

18 Impact, lessons learned and conclusions **215**

A List of Boderc publications **227**

B List of authors **233**

Chapter 17

Design trajectory and controller-plant interaction

Authors: P.M. Visser, J.F. Broenink and J. van Amerongen

17.1 Introduction

In the design process of a system various simulations and experiments should be performed to study the behavior of the system, to analyse the system performance and to make design decisions. As virtually all high-tech systems are multi-disciplinary in nature the modeling and simulation methods must deal with interaction between the various disciplines to support the system design. It is important to bring the disciplines together in an early design stage in order to avoid severe problems at the system integration phase that cause delays and additional design effort (see the Boderc research hypothesis in Chapter 1).

In this chapter a system design trajectory is proposed that facilitates the design steps from initial models to final realization. The system (or part of the system) is considered to be composed of three components: the plant, the input/output (I/O) interface and the controller as depicted in Figure 17.1.

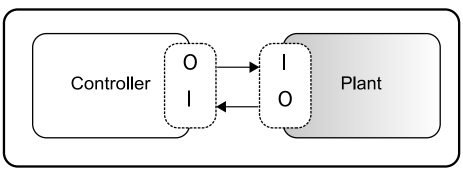

Figure 17.1: System scope

The plant is a physical device that can be controlled via the input/output (I/O) interface. For example, the paper path or a single pinch of the paper path that is driven by a motor. For the controller we will only consider the feedback control part, which in most modern systems is realized in software and embedded in the complete software of the system. The reason to focus is on the feedback control instead of the complete software is that the feedback control dominates the requirements of the hardware and software architecture. The emphasis of the feedback controller in this chapter is on the implementation on the control computer from the point of view of the software discipline; the control algorithm is supposed to be given.

Figure 17.2 shows an example of such a controller-plant system. The plant considered here is a motor that drives a pinch. The controlled variable of the plant is the angular position of the motor which can be measured by an encoder. The value of the encoder is sampled and forms the input of the digital feedback controller. The calculated output of the controller is applied by means of pulse width modulation (PWM) to the plant.

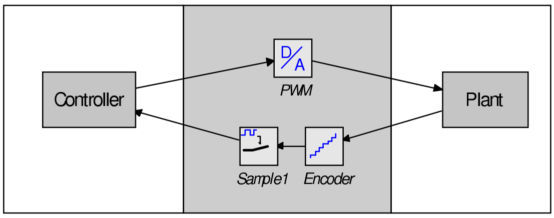

Figure 17.2: Controller-plant overview

The starting point of our system design trajectory consists of a running simulation of a plant model, a digital feedback controller model and the corresponding I/O. The plant does not have to be prototyped yet, the design trajectory can be commenced in the early design phases.

The final realization consists of the physical plant and the digital feedback controller that runs on the target. A target is a computer that is capable of computing the control law of the feedback controller.

The expected benefit from using the proposed design trajectory for the industrial user is a substantial reduction of the design time due to a reduction in integration effort. Moreover, as the interaction between the disciplines is clearer from the start of the design process, better choices can be made to improve the overall system behavior.

17.2 System design trajectory

The design trajectory depicted in Figure 17.3 proposes a systematic stepwise design trajectory with the goal to obtain a less error-prone path from model to realization [119]. It is a model-driven approach in which simulations are used to check whether refinement updates keep the model compliant with the requirements. Via various 'in-the-loop simulations', the design trajectory runs from complete simulation (stage 1) to final realization (stage 6). By dividing the design in multiple stages, possible errors are isolated and can be diagnosed faster. In each stage a verification test is performed. If a verification test fails one should locate and solve the error in the refinements with respect to the previous stage and repeat the verification test.

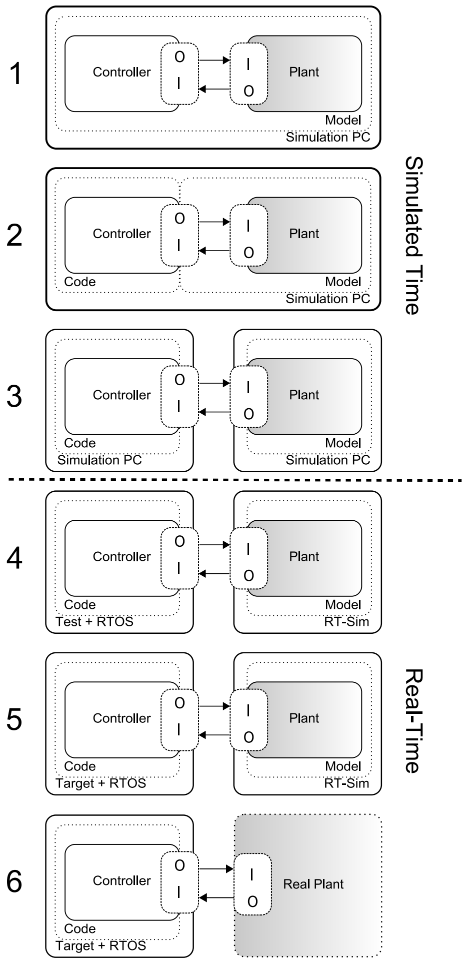

Figure 17.3: System design trajectory

In Figure 17.3 the simulated-time domain (stage 1, 2 and 3) and the real-time domain (stage 4, 5 and 6) are separated by a dashed line. The arrows between the I/O boxes denote the interconnections, which in the final realization stage are the connecting cables. The three components (controller, plant and I/O) are surrounded by a dotted box which is the realization 'form' of the components. The realization form can be the model, the code or the real instantiation (the physical plant). The outer solid box is the platform on which the model or code is simulated/executed. A platform is a computer that is capable of performing the calculations required to perform the simulation or execution. The following platforms are used:

- Simulation PC platform, a PC that is capable to execute the model simulation.
- Real-Time Simulation PC platform, a fast PC with a real-time simulator, that is capable to simulate the plant model in real-time.
- Test platform, any commercial off-the-shelf (COTS) platform that can be used to run the controller.
- Target platform, the platform that is used in the final realization.

The design trajectory will be explained per stage.

Stage 1

In this stage both the controller and plant are simulated in the same modeling environment/tool on the same Simulation PC. The plant model will be simulated with a numerical integration method to approximate the continuous-time behavior. In order to obtain a deterministic computation time a fixed step-size numerical integration method is chosen. This is to ensure that the plant can be simulated in real-time (stages 4 and 5). The step-size of the numerical integration method has to be chosen, among others, to adequately handle the plant dynamics [19] (see also Section 17.3.3).

The simulation will be used to analyze the plant behavior and optimize the controller. This stage can be called Model-In-the-Loop Simulation.

Stage 2

In this stage the controller model has been transformed to executable code by means of code generation, also called synthesis export, and compilation. The controller executable runs simultaneously with the simulation of the plant model. The verification test is obvious: the simulation results here should be identical to the simulation results of stage 1.

In principle, no discrepancies are expected, because the control-laws executed in simulation and executed in the executable should behave identical. An error that could occur, for example, is that a different floating point library is used in the controller model and the controller executable.

The purpose of this stage is to check that generated code yields exactly the same results as the simulation in stage 1, i.e. that the code generation from the modeling tool and compiler works as expected.

Stage 3

In this stage the controller executable and the plant model run on two separate Simulation PC's. The controller executable runs simultaneously with the simulation of the plant model. The interconnection between the Simulation PC's is via digital I/O. This implies that the I/O signals are still numbers and not yet physical signals. The controller runs in a non-real-time environment as a task.

This stage is used to obtain a rudimentary estimation of the CPU usage, which can be used to facilitate the choice for the target hardware. The simulation results should be identical to stage 2.

Stage 4

In this stage both the plant and controller run in *real-time* on two separate computers: the Test platform for the controller and the RT Simulation PC for the plant. The real-time simulation of the plant model must resemble the real plant behavior closely. Hence, the simulated plant model must have the same interface as the real plant, requiring that the I/O signals are the real physical signals. The simulated plant must be replaceable by the real plant without any modifications of the controller.

By choosing a Test platform similar to the Simulation PC's in the previous stages, only the migration of the nature of time is verified here. Since in stage 1, a fixed step-size numerical integration method was chosen, both pieces of code generated from the model (controller and plant) are functionally identical to stage 1 and should yield the same simulation results. However, due to the real-time setting, no synchronized communication between the plant and controller (which is explained in detail in Section 17.3.3) and limited computation time, simulation results may differ compared to those in the previous stages.

In this stage the real-time behavior of the controller can be studied and accurate processor and memory usage can be determined in order to determine the target platform. For example, a design trade-off can be made to accept a worse control performance (e.g. by changing the sample frequency) or chose for a faster (more expensive) target. Depending on the trade-off one should return to a previous stage to analyse the effect.

Although the target platform is not necessarily used for the controller, this stage can be considered as Hardware-in-the-loop-Simulation since various COTS Test platforms can be used to support the selection of the final target platform. The constraints posed by the target system are dealt with in the next stage.

Stage 5

In this stage the target platform replaces the test platform at the controller side. The transformation to this stage may be complicated by specific compilers and/or hardware resource limitations. Hence, the transformation to this stage will consume more time and should be taken after the hard real-time behavior is analyzed in the previous stage.

The verification test should show similar behavior compared to the previous stage. Similar but not identical since the timing of the target system and the compilers used may differ from the test platform. If the verification test is successful the real plant can be connected.

Stage 6, the realization

In this final stage, the plant model is replaced by the real plant. Because in stage 4 the interface of the real-time simulation of the plant model was similar to the real plant only the cable-ends of the target system need to be connected to the real plant. The controller and the target system are the same as in the previous stage.

Differences in the simulation results compared to the Hardware-in-the-loop-Simulation (stage 5) are caused by the difference between the plant simulation and the real plant.

In this stage the behavior of the final system should satisfy the requirements. If the requirements are met the design process has been successfully performed. If the requirements are not met, one should return to a previous stages to solve the issue.

A case study in [119] illustrates the results of the design trajectory.

17.3 Controller-plant interaction

This section will illustrate that the choice of the controller implementation approach is not strictly an implementation issue but a design issue which can have a large impact on the overall performance of the system and influences many design disciplines.

The controller-plant interaction is explained by using diagrams. In order to keep the diagrams readable and uncluttered the following simplifications and notational conventions are used:

- The controller does not receive an event directly at the time at which it occurs but receives the event at the next sampling moment. In implementation terms, the I/O buffers the event until the controller reads the buffer.
- The plant is simulated with a fixed step-size of T_c. $T_c(k_c)$ is used to denote the time of the simulated plant model at time $t = k_c T_c$.
- The sampling interval of the controller is T_d. $T_d(k_d)$ is used to denote the time of the digital controller at time $t = k_d T_d$.
- The step size of the plant simulation is chosen ten times smaller than the sampling interval of the digital feedback controller (T_d=$10T_c$). This choice is explained in Section 17.3.3.

17.3.1 Controller implementation approaches

The approach that is most common in practice is time-driven control. In time-driven control there are two different control approaches [7, pages 328-330], which are depicted in Figure 17.4. The values T_{AD} and T_{DA} are the analog to digital and digital to analog conversion times, which are assumed to be constant. The value T_{Comp} is the computational time required to compute the control signal. The computational time will vary and is denoted with $T_{Comp}(k_d)$.

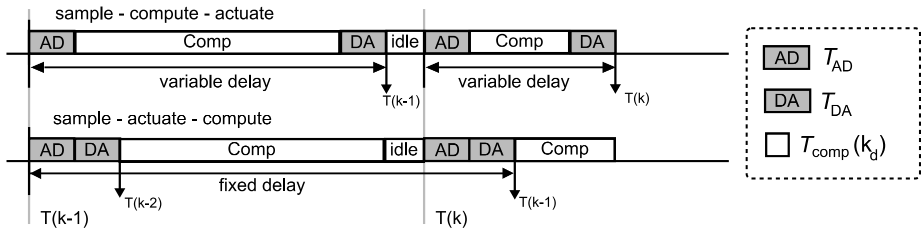

Figure 17.4: Control implementation approaches

The behavior of both approaches is as follows:

- 'sample-compute-actuate': a sample is taken on time $t = T_d(k_d)$ and used to compute the control signal which is applied on $t = T_d(k_d)+T_{AD}+T_{Comp}(k_d)+T_{DA}$. In this approach the control signal is applied with a varying time delay since the computational time could vary. The time between two control updates is periodic with jitter $(T_d + T_{Comp}(k_d + 1) - T_{Comp}(k_d))$.
- 'sample-actuate-compute': the control actuate signal for $t = T_d(k_d)$ is computed with the sample taken at $t = T_d(k_d - 1)$ and applied at $T_d(k_d) + T_{AD} + T_{DA}$. In this approach the control signal is applied with a fixed time delay equal to the sampling interval (T_d) plus the conversion times ($T_{AD} + T_{DA}$). The time between two control updates is periodic (T_d).

In both approaches there is a time delay between the moment a sample is taken and the moment the control signal is applied. To avoid issues in the final realization (stage 6) the time delay should explicitly be taken into account at the start of the design (stage 1). Therefore, the example controller-plant system of Figure 17.2 is extended with the addition of a time delay depicted in Figure 17.5 to deal with the control implementation approach. In the sample-compute-actuate approach, the time delay is $T_{Comp}(k_d)$. In the sample-actuate-compute approach, the time delay is T_d.

17.3.2 Interaction in a simulated-time simulation

The interaction in case of simulated-time simulation (stage 1, 2 and 3) is depicted in Figure 17.6. On the left side the sample-compute-actuate interaction is depicted and on the right side the sample-actuate-compute interaction is depicted.

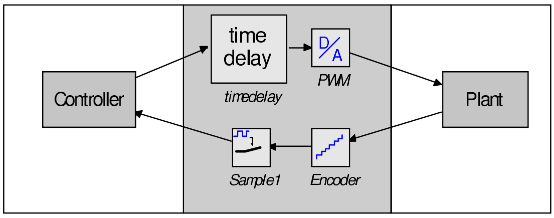

Figure 17.5: Controller-plant system with time delay

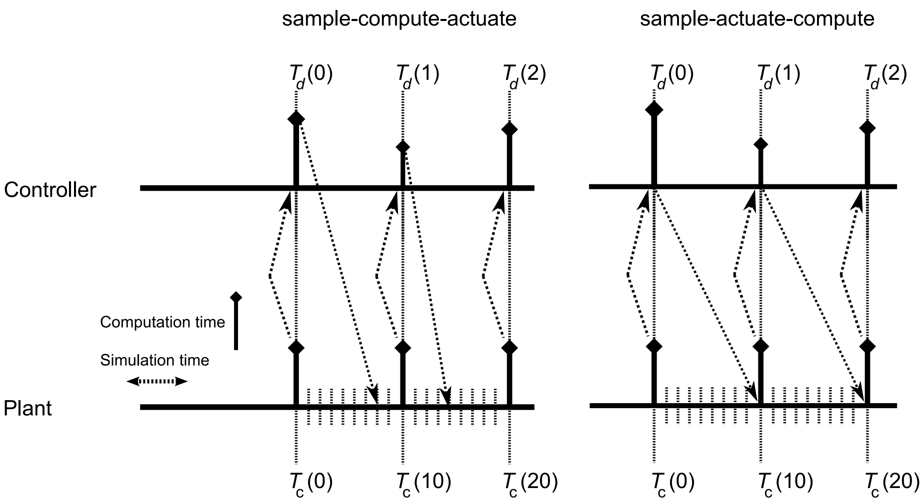

Figure 17.6: Simulated-time interaction

The main observation is that the control signal for a signal sampled at time $T_c(0)$ will be applied at $T_c(1) < t < T_c(10)$ in case of sample-compute-actuate (varying delay) and exactly at $T_c(10)$ in case of sample-actuate-compute (fixed delay).

In the simulated-time both the controller and the plant are synchronized by the modeling environment which prevents the occurrence of drift ($T_d(1) = T_c(10)$).

To accurately model the varying time delay (caused by using a computer to implement the control law) in the sample-compute-actuate approach, one needs detailed knowledge about the final target on which the controller will run. The delay can be estimated by analyzing the time that the instructions of the control algorithm (e.g. PID) will take on the target. A maximum time may be chosen if it can be determined that the varying delay does not hamper the control performance. Simulation can be used to study the impact of the varying delay.

In case of the sample-actuate-compute approach the delay is fixed to a unit delay (T_d). This requires no knowledge of the target and a proper controller design is able to deal with such a fixed time delay.

From a system point of view the sample-actuate-compute approach is preferred. The approach allows a predictable design since no knowledge is required of the target, which may not be chosen at the start of the design process. In systems were the 'best' obtainable control performance is required and costs are of secondary importance, the sample-compute-actuate approach may be preferable. In such a system the varying delay will be small with respect to the sampling interval (T_d), since a high performance computer is used for computing the control algorithm.

For both approaches the target has to be chosen fast enough to compute the control algorithm in time.

17.3.3 Interaction in a real-time simulation

The interaction in case of real-time simulation (stage 4 and 5) is shown in Figure 17.7. On the left side the sample-compute-actuate interaction is depicted and on the right side the sample-actuate-compute interaction is depicted. The computation time is strictly coupled with real-time, denoted with squares.

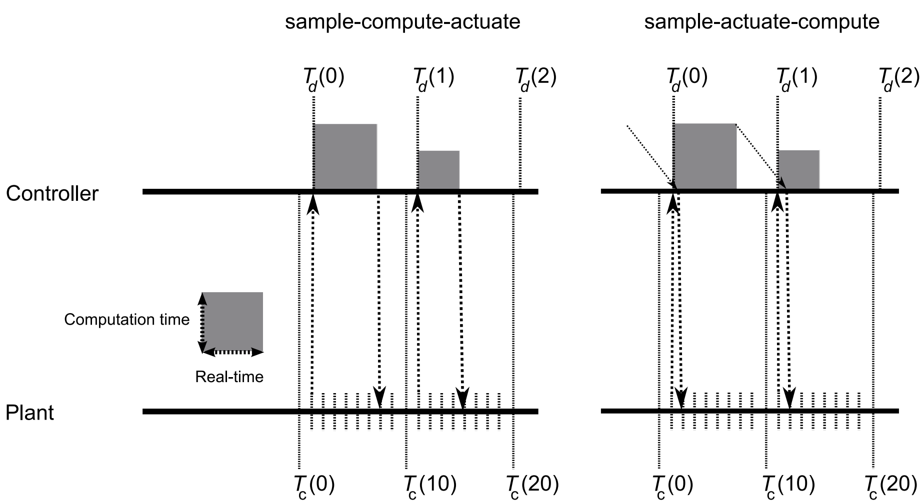

Figure 17.7: Real-time interaction

In real-time simulation there is no synchronization of time between the simulated plant model and the controller. Opposed to the *simulated-time* simulation, the controller will not wait for the plant to calculate its output and vice versa. In the simulated-time domain computing a second in simulation may take 10 seconds of computation time. In the real-time domain computing a second in real-time simulation must take less (or equal) than a second. As a consequence of the asynchronous behavior the time may drift ($T_d(1) \neq T_c(10)$). The asynchronous behavior is caused by fact that the controller and the simulation of the plant model run on separate computers with separated clocks. The impact on the simulation results caused by this asynchronous interaction depends on the clock drift and the step-size of the plant simulation. Clock synchronization e.g. by hard-wiring is not 'allowed' because the idea of Hardware-in-the-Loop simulation is that the simulated plant can be replaced with the real plant without any modifications. The error caused by this asynchronous interaction is at least one numerical integration step (T_c). Hence, decreasing the step-size (T_c) of the plant simulation will decrease the error caused by the asynchronous interaction. A ratio of at least $10T_c \geq T_d$ is advised.

The real-time simulation results for the sample-actuate-compute approach will be similar to the simulated-time simulation results if the target is fast enough to compute the controller output in time. The results will not be identical because of the asynchronous interaction.

The simulation results for the sample-compute-actuate approach may differ from the simulated-time simulations. This is the case when the estimated time for the computational delay (used in stages 1,2 and 3) is not the same as the actual time that is required for the computation in the real-time simulations. If the control performance is not satisfactory, one has to adjust the estimated computational time in one of the previous simulated-time stages.

17.4 Conclusions and discussion

In this chapter we presented a systematic stepwise design trajectory to obtain a less error-prone path from model to final realization. For two common control implementation approaches the controller-plant interaction was discussed and indicated how they should be handled within the design trajectory.

In the design trajectory simplifying assumptions were made on the controller-plant interaction. Future work will focus on removing these assumptions. In particular, in the controller-plant interaction the events were only received by the controller at the sample moments. This is a limitation as many reactive systems need to deal with events at the moment they occur. Hence, the next step is to deal with events, both in the simulated-time simulations and the real-time simulations, in a realistic manner. When events can be taken into account the stepwise refinement trajectory can be extended from time-driven (synchronous) control to event-driven (asynchronous) control as discussed in Chapter 16.

Bibliography

[1] P.P. Acarnley and P. Gibbons. Closed-loop control of stepping motors: prediction and realisation of optimum switching angle. *IEEE Proceedings*, Part B, 129(4), 1982.

[2] I. Alexander. Towards automatic traceability in industrial practice. *Proceedings of the First International Workshop on Traceability, Edinburgh*, pages 26–31, 2002.

[3] G. Altshuller. The innovation algorithm. triz, systematic innovation and technical creativity. 2000.

[4] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[5] D. Amyot and G. Mussbacher. Urn: Towards a new standard for the visual description of requirements. *LNCS 2599*, pages 21–37, June 2002.

[6] A. Antón, J. Dempster, and D. Siege. Managing use cases during goal driven requirements engineering: Challenges encountered and lessons learned. *TR-99-16, North Carolina State Univ*, December 1999.

[7] Karl J. Åström and Björn Wittenmark. *Computer Controller Systems: Theory and Design*. Prentice Hall, Upper Saddle River, New Jersey, 3rd edition, 1997.

[8] S. Baruah, D. Chen, and A. Mok. Jitter concerns in periodic task systems. *Proceedings of the Eighteenth Real-Time Systems Symposium, San Francisco, CA*, pages 68–77, 1997.

[9] I. Bate and N. Audsley. Flexible design of complex high-integrity systems using trade offs. *Proc. 8th Int. Symp. High Assurance Systems Engineering*, 2004.

[10] J.O. Bayer, P. Flege, R. Knauber, D. Laqua, K. Muthig, T. Schmid, Widen, and J.M. DeBaud. Pulse: A methodology to develop software product lines. *Proceedings of the symposium on software reusability*, pages 122–131, 1999.

[11] G. Behrmann, A. David, and K.G. Larsen. A Tutorial on UPPAAL. In *Formal Methods for the Design of Real-time Systems*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–236. Springer, 2004.

[12] Matthijs H. ten Berge, Bojan Orlic, and Jan F. Broenink. Co-simulation of networked embedded control systems, a CSP-like process-oriented approach. In *Proc. of joint CCA, CACSD and ISIC*, pages 434–439, Munich, Germany, October 2006.

[13] S.S. Blackman. *Multiple target tracking with radar applications*. Norwood, MA: Artech House, 1986.

[14] H. Bode. *Network Analysis and Feedback Amplifier Design*. Van Nostrand Reinhold, New York, 1945.

[15] Leo J. van Bokhoven, Jeroen P.M. Voeten, and Marc C.W. Geilen. Software synthesis for system level design using process execution trees. In *Proc. of 25th Euromicro Conference*, pages 463–467, 1999.

[16] L. Boltzmann. Ableitung des stefanschen gesetzes betreffend die abhängigkeit der wärmestrahlung. *Annalen der Physik und Chemie*, 22, 1884.

[17] Egor Bondarev, Michel Chaudron, and Peter de With. Quality-oriented design space exploration for component-based architectures. Technical report, Technical University of Eindhoven, Department of Mathematics and Computer Science, February 2006.

[18] P.F.A. van den Bosch and E.H. van de Waal. A case study of multi-disciplinary modelling using matlab/simulink and truetime. *Proc. INCOSE Symposium 2005*, 2005.

[19] P.P.J. van den Bosch and A.C. van der Klauw. *Modeling, Identification and Simulation of Dynamical Systems*. CRC Press Inc., 1994.

[20] P.C. Breedveld. *Dynamische systemen : modelvorming en simulatie met bondgrafen*. Open universiteit, The Netherlands, 1994.

[21] J.F. Broenink and J.P.M. Voeten. Viewcorrect: Predictable co-design for distributed embedded control systems.

[22] D. de Bruin and P.P.J. van den Bosch. Measurement of the lateral vehicle position with permanent magnets. In *Proceedings of IFAC workshop on Intelligent Components for Vehicles*, pages 9–14, Seville, Spain, 1998.

[23] R.J.A. Buhr. Use case maps as architectural entities for complex systems. *IEEE Transactions on Software Engineering*, 24, Issue 12:1131 – 1155, December 1998.

[24] B.H.M. Bukkems, M.J.G. van de Molengraft, W.P.M.H. Heemels, N. van de Wouw, and M. Steinbuch. A piecewise linear approach towards sheet control in a printer paper path. In *Proc. of the American Control Conference*, pages 1315–1320, Minneapolis, USA, 2006.

[25] B.H.M. Bukkems, J.H. Sandee, J.B.C. Beckers, Z. Yuan, B. van der Wijst, and M.J.G. van de Molengraft. A case-study in multidisciplinary modeling of dynamic embedded systems. *IEEE Conference on Mechatronics and Robotics 2004, Aachen, Germany.*, 2004.

[26] Giorgio C. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Kluwer Academic Publishers, 1997.

[27] Anton Cervin, Dan Henriksson, Bo Lincoln, Johan Eker, and Karl-Erik Årzen. How does control timing affect performance? *IEEE Control Systems Magazine*, pages 16–30, June 2003.

[28] S. Chakraborty, S. Künzli, and L. Thiele. A general framework for analysing system properties in platform-based embedded system designs. In *Proc. 6th Design, Automation and Test in Europe*, pages 190–195, 2003.

[29] M. Chen, C.R. Zhu, and G. Feng. Linear-matrix-inequality-based approach to H_∞ controller synthesis of uncertain continuous-time piecewise linear systems. *IEE Proc.-Control Theory Appl.*, 151(3):295–300, May 2004.

[30] Carlo Cloet. *A Mechatronics Approach to Copier Paperpath Design*. PhD thesis, University of California, Berkeley, CA, USA, 2001.

[31] Marieke Cloosterman, Nathan van de Wouw, Maurice Heemels, and Henk Nijmeijer. Robust control of networked control systems with uncertain time-varying delays. *DCT internal report 2006-121*, 2006.

[32] Marieke Cloosterman, Nathan van de Wouw, Maurice Heemels, and Henk Nijmeijer. Robust stability of networked control systems with time-varying network-induced delays. In *Proc. of the 45th Conference on Decision and Control*, San Diego, California, USA, December 2006.

[33] A. Cockburn. *Writing effective use cases.* Addison-Wesley, 2000.

[34] P. Crnosija. Microcomputer implementation of optimal algorithms for closed-loop control of hybrid stepper motor drives. *IEEE Transactions on Industrial Electronics*, 47(6), 2000.

[35] J. Dahmann, R. Fujimoto, and R. Weatherly. The department of defense high level architecture. In *The 1997 Winter Simulation Conference*, pages 142–149, 1997.

[36] R.L. Dillon, M.E. Paté-Cornell, and S.D. Guikema. Programmatic risk analysis for critical engineering systems under tight resource constraints. *Operations Research*, 51, No. 3:354–370, 2003.

[37] R.C. Doff, M.C. Fatten, and C.A. Phillips. Adaptive sampling frequency for sampled-data control systems. *IRE Transactions on Automatic Control*, AC-7:38–47, 1962.

[38] P.G. Engeldrum. *Psychometric scaling: A Toolkit for Imaging Systems*. Imcotek Press, 2000.

[39] Gang Feng. Controller design and analysis of uncertain piecewise-linear systems. *IEEE Trans. Circuits Syst. I*, 49(2):224–232, February 2002.

[40] Oana Florescu, Menno de Hoon, Jeroen Voeten, and Henk Corporaal. Performance modelling and analysis using poosl for an in-car navigation system. In *Proceedings of the 12th Annual Conference of the Advanced School for Computing and Imaging (ASCI)*, pages 37–45, June 2006.

[41] Oana Florescu, Menno de Hoon, Jeroen Voeten, and Henk Corporaal. Probabilistic modelling and evaluation of soft real-time embedded systems. In *Proceedings of the Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS VI)*, LNCS 4017, pages 206–215, July 2006.

[42] Oana Florescu, Jinfeng Huang, Jeroen Voeten, and Henk Corporaal. Strengthening property preservation in concurrent real-time systems. In *Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 106–109, August 2006.

[43] Oana Florescu, Jeroen Voeten, and Henk Corporaal. Modelling patterns for analysis and design of real-time systems. Technical Report ESR-2006-05, Eindhoven University of Technology, 2006.

[44] Oana Florescu, Jeroen Voeten, Jinfeng Huang, and Henk Corporaal. Error estimation in model-driven development for real-time software. In *Proceedings of the Forum on Specification & Design Languages 2004 (FDL'04)*, September 2004.

[45] Oana Florescu, Jeroen Voeten, Marcel Verhoef, and Henk Corporaal. Reusing real-time systems design experience through modelling patterns. In *Proceedings of the Forum on Specification & Design Languages 2006 (FDL'06)*, September 2006.

[46] International Organisation for Standardization. Space systems - mass properties control. Proposal ISO International Standard, ISO/CD 22010.

[47] Gene F. Franklin, J. David Powell, and Abbas Emami-Naeini. *Feedback control of dynamic systems*. Prentice Hall, Upper Saddle River, New Jersey, USA, 2002.

[48] H.J.M. Freriks. White paper on designing with stepper motors. Technical report, 2005.

[49] H.J.M. Freriks, W.P.M.H. Heemels, and G.J. Muller. On the systematic use of budget-based design. *Proceedings of 16th annual international symposium of the INCOSE*, 2005.

[50] B. Friedland. Optimum steady-state position and velocity estimation using sampled position data. *IEEE transactions on Aerospace and Electronic Systems*, AES-9(6):906–911, 1973.

[51] P. Gahinet, A. Nemirovski, A. J. Laub, and M. Chilali. *LMI Control Toolbox for Use with Matlab*. The Mathworks Inc., Natick, MA, USA, May 1995.

[52] Marc G.W. Geilen. *Formal Techniques for Verification of Complex Real-Time Systems*. PhD thesis, Eindhoven University of Technology, Eindhoven NL, 2002.

[53] Geilen, M.C.W. and Voeten, J.P.M. and Putten, P.H.A. van der and Bokhoven, L.J. van and Stevens, M.P.J. Poosl. *Computer Languages 27*, pages 19–38, 2001. http://www.es.ele.tue.nl/poosl.

[54] T. Glad and L. Ljung. Velocity estimation from irregular, noisy position measurements. In *Proceedings of the IFAC 9th Triennial World Congress*, pages 1069–1073, Budapest, 1984.

[55] Yoram Halevi and Asok Ray. Integrated communication and control systems: Part i and ii. *Journal of Dynamic Systems, Measurement, and Control*, 110(4):367–381, 1988.

[56] W.P.M.H. Heemels, R.J.A. Gorter, A. van Zijl, P.P.J. van den Bosch, S. Weiland, W.H.A. Hendrix, and M.R. Vonder. Asynchronous measurement and control: a case study on motor synchronization. *Control Engineering Practice*, 7:1467–1482, 1999.

[57] W.P.M.H. Heemels, E. v.d. Waal, and G.J. Muller. A multi-disciplinary and model-based design methodology for high-tech systems. *Proceedings of CSER*, 2006.

[58] Martijn Hendriks and Marcel Verhoef. Proc. timed automata based analysis of embedded system architectures. In *Workshop on Parallel and Distributed Real-Time Systems*, 2006.

[59] João P. Hespanha, Payam Naghshtabrizi, and Yonggang Xu. Networked control systems: Analysis and design. To appear in the *Proc. of IEEE*, Special Issue on Networked Control Systems, 2006.

[60] S. van der Hoest. The development of a software-in-the-loop simulation framework for testing real-time control software. Technical report, Stan Ackermans Institute, Eindhoven, 2006.

[61] J. Hooman, N. Mulyar, and L. Posta. Coupling Simulink and UML models. In B. Schnieder and G. Tarnai, editors, *Proceedings of Symposium FORMS/FORMATS 2004*, pages 304–311, 2004.

[62] Jozef Hooman, Hillel Kugler, Iulian Ober, Anjelika Votintseva, and Yuri Yushtein. Supporting uml-based development of embedded systems by formal techniques. *Software and Systems Modeling*, to appear.

[63] Jinfeng Huang, Jeroen P.M. Voeten, and Marc C.W. Geilen. Real-time property preservation in approximations of timed systems. In *Proc. of 1st Conference on Formal Methods and Models for Codesign (MEMOCODE)*, pages 163–171, June 2003.

[64] A. Hughes and P.J. Lawrenson. Simple theoretical stability criteria for 1.8° hybrid motors. *Int. Symp. on Stepping Motors and Systems*, 1979.

[65] IBM/Rational. Rose realtime. http://www.ibm.com/.

[66] M. Jazayeri, A. Ran, and F. v.d. Linden. Software architecture for product families. 2000.

[67] C. Jongeneel. Klokloze chips. *De Ingenieur*, 117(4):52–53, 2005.

[68] D.C. Karnopp, D.L. Margolis, and R.C. Rosenberg. *System Dynamics, modeling and simulation of Mechatronic Systems*. John Wiley & sons, New York, third edition edition, 2000.

[69] G.A. Katopis and et al. MCM technology and design for the S/390 G5 System. *IBM Journal of Research and Development*, 43, No 5/6, September/November 1999.

[70] Bart Kienhuis, Ed Deprettere, Kees Vissers, and Pieter van der Wolf. An approach for quantitative analysis of application-specific dataflow architectures. In *Proceedings of the IEEE ASAP*, pages 338–349, 1997. International Conference on Application-Specific Systems, Architectures and Processors.

[71] T. Kostelijk. Misleading architecting tradeoffs. *IEEE Computer*, May 2005.

[72] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[73] M. Krucinski, C. Cloet, M. Tomizuka, and R. Horowitz. Asynchronous observer for a copier paper path. In *Proceedings of the 37th IEEE Conference on Decision and Control*, volume 3, pages 2611–2612, Tampa, Florida, 1998.

[74] Martin Krucínski. *Feedback Control of Photocopying Machinery*. PhD thesis, University of California, Berkeley, CA, USA, 2000.

[75] C. van Lamsweerde. Goal oriented requirements engineering: A guided tour. *Proc. 5th IEEE Int. Symp. on Requirements Eng.*, pages 249–263, August 2001.

[76] Feng-Li Lian. *Analysis, Design, Modelingadn Control of Networked Control Systems*. PhD thesis, University of Michigan, Ann Arbor, USA, 2001.

[77] J.A. López-Orozco, J.M. de la Cruz, E. Besada, and P. Ruipérez. An asynchronous, robust, and distributed multisensor fusion system for mobile robots. *The international Journal of Robotics Research*, 19(10):914–932, 2000.

[78] M.W. Maier and E. Rechtin. *The Art of Systems Architecting*. CRC Press, Boca Raton, second edition, 2002.

[79] R. Malan and D. Bredemeyer. Software architecture action guide. 2005.

[80] E. Mohammed and et al. Optical interconnect system integration for ultra-short-reach applications. *Intel Technical J. on Optical Technologies and Applications*, 8, No 2, May 2004.

[81] René van de Molengraft, Bram de Kraker, and Maarten Steinbuch. Integrating experimentation into control courses. *IEEE Control Syst. Mag.*, 25(1):40–44, February 2005.

[82] G.J. Muller. *CAFCR: A multi-view method for embedded systems architecting; balancing genericity and specificity.* PhD thesis, Delft University of Technology, Delft, The Netherlands, 2004.

[83] G.J. Muller. Do useful multi-domain methods exist? *Conference System Engineering Research (CSER), Hobroken, USA*, March 2005.

[84] G.J. Muller. Industry and academia: Why practitioners and researchers are disconnected. *Proc. INCOSE Symposium, Rochester, NY, USA.*, 2005.

[85] Payam Naghshtabrizi and Joao P. Hespanha. Designing an observer-based controller for a network control system. In *Proc. of the 44th Conference on Decision and Control, and the European Control Conference 2005*, pages 848–853, Seville, Spain, December 2005.

[86] Xavier Nicollin and Joseph Sifakis. An overview and synthesis on timed process algebras. In *Proc. of the Real-Time: Theory in Practice, REX Workshop*, pages 526–548, London, UK, 1992. Springer-Verlag.

[87] Johan Nilsson. *Real-Time Control Systems with Delays*. PhD thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1998.

[88] OMG. *Unified Modeling Language (UML) - Version 1.5*. OMG document formal/2003-03-01, Needham MA, 2003.

[89] OMG. *UML Profile for Schedulability, Performance, and Time Specification - Version 1.1*. OMG document formal/2005-01-02, 2005.

[90] International Council on Systems Engineering (INCOSE) Technical board. *System Engineering Handbook*. INCOSE, 2004. A "what to" guide for all se practitioners.

[91] G.J.P.M. van Oosterhout. Spectral color prediction by advanced physical modeling of toner, ink and paper, with application to halftoned prints. *Proc. ISandT's NIP19*, pages 797–, 2003.

[92] David L. Parnas. Software engineering: An unconsummated marriage. *Communications of the ACM*, page 128, September 1997. This article can also be found in Software Fundamentals,Papers by David Parnas, Addison-Wesley.

[93] R. Phaal, C.J.P. Farrukh, and D.R. Probert. Technology roadmapping - a planning framework for evolution and revolution. *Technological Forecasting & Social Change*, 71, No 1-2:5–26, 2004.

[94] A.M. Phillips and M. Tomizuka. Multirate estimation and control under time-varying data sampling with applications to information storage devices. In *Proceedings of the 1995 American control conference*, volume 6, pages 4151–4155, 1995.

[95] Colin Potts. Software-engingeering research revisited. *IEEE Software*, Vol. 10, No. 5:19–28, September/October 1993.

[96] P.H.A. van der Putten and J.P.M. Voeten. Specification of reactive hardware/software systems - the method software/hardware engineering. 1997.

[97] QFD Institute. QFD institute. `http://www.qfdi.org/`, 2000.

[98] Sudhendu Rai and Warren B. Jackson. A hybrid hierarchical control architecture for paper transport systems. In *Proc. of the* 37^{th} *IEEE Conference on Decision and Control*, pages 4249–4250, Tampa, Florida, USA, December 1998.

[99] J.B. ReVelle. *The QFD handbook*. Wiley, 1998.

[100] H. Saiedian, P. Kumarakulasingam, and M. Anan. Scenario-based requirements analysis techniques for real-time software systems: a comparative evaluation. *Requirements Eng. J.*, 10:22–33, 2005.

[101] J.H. Sandee. *Event-driven control in theory and practice - trade-offs in software and control performance*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2006.

[102] J.H. Sandee, W.P.M.H. Heemels, G.J. Muller, P.F.A. van den Bosch, and M.H.G. Verhoef. Threads of reasoning: A case study. *Proceedings of the 16th annual international symposium of INCOSE*, 2006.

[103] R. Sanz and K.-E. Årzén. Trends in software and control. *IEEE Control Systems Magazine*, 23(3):12–15, 2003.

[104] A.J. van der Schaft and J.M. Schumacher. An introduction to hybrid dynamical systems. *Lecture Notes in Control and Information Sciences*, page Vol. 251, 1999.

[105] S.A. Schweid and et al. Hybrid step motor position estimation from back emf. *IEEE Conf. on Control Applications*, 1995.

[106] B. Selic, G. Gullekson, and P.T. Ward. *Real-Time Object-Oriented Modeling*. John Wiley & Sons, 1994.

[107] Simulink. The Mathworks. http://www.mathworks.com/products/simulink/.

[108] S.T. Stanton and et al. Overlay error budgets for a high-throughput scalpel system. *Proceedings of SPIE*, 3676:543–555, June 1999.

[109] George Stephanopoulos. *Chemical Process Control*. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1984.

[110] J. Stolte. Understanding instability in hybrid stepper motors. Technical report, Eindhoven University of Technology, 2006.

[111] Telelogic. Combining Rhapsody and Simulink. http://www.telelogic.com/.

[112] Lothar Thiele, Samarjit Chakraborty, and Martin Naedele. Real-time calculus for scheduling hard real-time systems. In *Proc. IEEE International Symposium on Circuits and Systems*, volume 4, pages 101–104, 2000.

[113] Yodyium Tipsuwan and Mo-Yuen Chow. Control methodologies in networked control systems. *Control Engineering Practice*, 11:1099–1111, 2003.

[114] TrueTime. http://www.control.lth.se/truetime/.

[115] UCM-URN. Use case maps. http://www.usecasemaps.org/index.shtml.

[116] Toronto University. Goal oriented requirements language.

[117] A. Veltman and P.P.J. van den Bosch. A universal method for modelling electrical machines. *Conf. on Electrical Machines and Drives*, 1992.

[118] Marcel Verhoef, Peter Gorm Larsen, and Jozef Hooman. Modeling and validating distributed embedded real-time systems with VDM++. In J. Misra, T. Nipkow, and E. Sekerinski, editors, *Proc. Formal Methods 2006*, volume 4085 of *LNCS*, pages 147–162. Formal Methods Europe, Springer, 2006.

[119] Peter M. Visser and Jan F. Broenink. Controller and plant system design trajectory. In *Proc. IEEE Int'l Symposium on Computer Aided Control Systems Conference, CACSD 2006*, pages 1910–1915, Munich, 2006. IEEE.

[120] C.V. Vottis. *Extracting more accurate position and velocity information from optical incremental encoders*. SAI/2yr Thesis, Technische Universiteit Eindhoven, Netherlands, 2003.

[121] G.C. Walsh, H. Ye, and L. Bushnell. Stability analysis of networked control systems. In *Proc. of the American Control Conference*, pages 2876–2880, San Diego, California, USA, June 1999.

[122] E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse. System architecture evaluation using modular performance analysis - a case study. *Intern. journal on software tools technology transfer*, 2006.

[123] P.M. Wecksten, J. Vasell, and M. Jonsson. Towards a tool for derivation of implementation constraints. *Proc. 9th IEEE Int. Conf. Engineering Complex Computer Systems Navigating Complexity in the e-Engineering Age*, 2004.

[124] R. Wieringa. Requirements engineering: problem analysis and solution specification. *ICWE*, pages 13–16, 2004.

[125] Björn Wittenmark, Johan Nilsson, and Martin Törngren. Timing problems in real-time control systems. In *Proc. of the American Control Conference*, pages 2000–2004, Seattle, Washington, USA, 1995.

[126] Y. Yamada. Exposure tool strategy for 90nm 65nm production. *Semiconductor Fabtech, 18th edition*.

[127] S-M Yang and E-L. Kuo. Damping a hybrid stepper motor with estimated position and velocity. *IEEE Transactions on Power Electronics*, 18(3), 2003.

[128] T. C. Yang. Networked control system: a brief survey. *IEE Proc.-Control Theory Appl.*, 153 (4):403–412, July 2006.

[129] Yuequen Yang, De Xu, Min Tan, and Xianzhong Dai. Stochastic stability analysis and control of networked control systems with randomly varying long time-delays. In *Proc. of the 5th World Congress on Intelligent Control and Automation*, pages 1391–1395, Hangzhou, China, June 2004.

[130] E. Yu and J. Mylopoulos. Why goal oriented requirements engineering. *Proc. 4th Int. Workshop Req. Eng: Foundations of Software Quality, Pisa*, pages 15–22, June 1998.

[131] Mei Yu, Long Wang, Tianguang Chu, and Guangming Xie. An LMI approach to networked control systems with data packet dropout and transmission delays. In *Proc. of the 43rd Conference on Decision and Control*, pages 3545–3550, Atlantis, Paradise Island, Bahamas, December 2004.

[132] By Wei Zhang, Michael S. Branicky, and Stephen M. Phillips. Stability of networked control systems. *IEEE Control Systems Magazine*, 21:84–99, February 2001.