



KTH Machine Design

Quality of control and real-time scheduling

Allowing for time-variations in
computer control systems

MARTIN SANFRIDSON

Doctoral thesis
Department of Machine Design
Royal Institute of Technology
Stockholm, Sweden

TRITA-MMK 2004:7
ISSN 1400-1179
ISRN/KTH/MMK--04/7--SE


TRITA-MMK 2004:7
ISSN 1400-1179
ISRN/KTH/MMK--04/7--SE

Quality of control and real-time scheduling — allowing for time-variations in computer control systems.

Martin Sanfridson

Doctoral thesis

Academic thesis, which with the approval of Kungliga Tekniska Högskolan, will be presented for public review in fulfilment of the requirements for a Doctorate of Engineering in Machine Design. The public review is held at Kungliga Tekniska Högskolan, Valhallavägen 79, Stockholm in Kollegiesalen at 09.00 am on the 4th of June 2004.

Mechatronics Lab Department of Machine Design Royal Institute of Technology S-100 44 Stockholm, Sweden		TRITA-MMK 2004:7 ISSN 1400-1179 ISRN KTH/MMK--04/7--SE	
<i>Author(s)</i> Martin Sanfridson mis@md.kth.se		<i>Document type</i> Doctoral Thesis	<i>Date</i> 2004-05-07
<i>Title</i> Quality of control and real-time scheduling — allowing for time-variations in computer control systems		<i>Sponsor(s)</i> Vinnova (Nutek) SSF via Artes	
<i>Abstract</i> <p>The majority of computers around us are embedded in products and dedicated to perform certain tasks. A specific task is the control of a dynamic system. The computers are often interconnected by communication networks forming a distributed system. Vehicles and manufacturing equipment are two types of mechatronic machines which often host dedicated computer control systems. A research problem is how the real-time behaviour of the computer system affects the application, especially the control of the dynamic system.</p> <p>If the internal or external conditions varies over time, it becomes difficult to assign a fixed resource reservation that will work well in all situations. In general, the more time an application gets of a resource, the better its gauged or perceived quality will be. A strategy is to alter the resource reservation when the condition changes. This can be constructed as a negotiation between competing applications, a method for which the term <i>quality of control</i>, QoC, has been coined. Scalability is the ability to change the structure and configuration of a system. It promotes evolving systems and a can help manage a complex product family. An architecture for a QoC middleware on top of a scalable computer system, has been proposed.</p> <p>As a <i>quality measure</i> of a control application, the well-known weighted quadratic loss function used in optimal control, has been revised to encompass a subset of the so called <i>timing properties</i>. The timing properties are the periods and the delays in the control loop, including time-varying period and delay. They are the interface between control and computer engineering, from a control engineering viewpoint. The quality measure can be used both off-line and on-line given a model of the sampled-data system and an appropriate description of the timing properties.</p> <p>In order to use a computer system efficiently and to guarantee its responsiveness, real-time scheduling is a must. In fixed priority scheduling each task arrives periodically and has a fixed priority. A task with a high priority can preempt a low priority task and gain access to the resource. The best-case response time characterizes the delays in the system, which is useful from a control viewpoint. A new algorithm to calculate the <i>best-case response time</i> has been derived. It is based on a scheduling scenario which yields a recurrence equation. The model is dual to the well-known worst-case response time analysis.</p> <p>Besides the dynamic fixed priority scheduling algorithm, optimal control using <i>static scheduling</i> has been studied, assuming a limited communication. In the static schedule, which is constructed pre-runtime, each task is assigned a time window within a schedule repeated in eternity. The optimal scheduling sequence is sought by optimizing the overall control performance. An interesting aspect is that the non-specified control period falls out as a result of the <i>optimal schedule</i>. The time-varying delay is accounted for in the control design.</p>			
<i>Keywords</i> Real-time scheduling, sampled-data control, performance measure, quality of control, limited communication, time-varying delay, jitter.		<i>Language</i> English	

Preface

Mechatronics is a multi-disciplinary research field. When I started at DAMEK, it was within the Nutek funded project DICOSMOS — a cooperation between DAMEK, Automatic control in Lund and Computer engineering at Chalmers and during the last years of the project also Volvo Technological Development. The acronym was spelled out: distributed control of safety-critical motion systems. This reveals the span of possible research directions that were open to me. I decided to focus on the so called timing properties, which taken together constitute an interface between control and computer engineering, from a control engineering point of view. My first study plan contained the phrase “the controller is not alone in the loop”. This research line is still pursued as quality of control. The work on quality of control is substantially different from the work on the best-case response time, the performance measure or the optimal scheduling. It is a top down approach which never really makes it down to a model of any kind, but to a description and proposal of an architecture.

During the first years, I read a lot of control theory, real-time scheduling and similar subjects. Like many other PhD students I was puzzled by the thought of finding something new, or “unpublished and original” as it is usually expressed in a call for papers. Today, it feels like I can figure out more interesting problems with reasonable technical innovation potential, than there is time to dig into them. The text book and the research papers teach you how to think and how to model a problem. New ideas will come if you stare at the model and equations long enough to start viewing the problem from a different perspective and long enough to start asking the right questions. The great idea will come a gloomy day in October, a rainy day when you don’t feel like riding the bicycle through the city to the office. It will come as a sudden glimpse of clarity the moment you sink down in your sofa.

To borrow terms from the control theory, research is in my opinion observable but not controllable. Hard work and luck are two virtues, and you might very well stumble over something relevant and fruitful. The general research direction should be broad, but the actual work should be restricted and focused. Looking back, it is painful to see how much effort was spent on ideas that were later surrendered. The PhD study is an education to take responsibility, find a problem, search data bases, read related literature, dig into the theory, read more, find a better problem and dig deeper. The devil is in the details.

Acknowledgement

First of all, I would like to express my gratitude to my supervisors Professor Martin Törngren and Professor Jan Wikander. Thank you Martin for always being so optimistic and for your golden comments. Thank you Jan for your understanding and supportive guidance through all these years.

The Mechatronics lab — or DAMEK — a division of the department of Machine Design at KTH, is a wonderful place to work and it is full of talented people. It has been fun.

And to my co-authors: Thank you Dr. Ola Redell (DAMEK) and Dr. Henrik Rehbinder (KTH, OptSyst) for good cooperation in the writing of successful papers! I've learned a lot. Spontaneous cooperation deserves much attention.

During my many weeks in Gothenburg for the DICOSMOS case study, I particularly enjoyed working with Dr. Magnus Gäfvert (Automatic control, LTH) and Dr. Vilgot Claesson (Computer engineering, Chalmers), and learned to enjoy their mutual devotion to Thai food lunches and movie nights. I would also like to thank Volvo Technological Development and the group around Dr. Mats Andersson, for their part in the DICOSMOS case study.

I would like to express my gratitude to my fellow colleagues of the ReTiS Lab at Scuola Superiore Sant'Anna, especially Paolo, Luigi and Peppe, for sharing their interests and enthusiasm with me during my stay in Pisa.

And last but not least, to my friends, relatives, parents and sister, who have done nothing to this thesis, but who have endured my many working days — thank you.

Martin Sanfridson

KTH, May 2004

Acknowledgement

Appended papers

PAPER A

Martin Sanfridson and Ola Redell, “Exact best-case response-time analysis of fixed priority scheduled tasks with arbitrary response times”, submitted to a journal, September 2003.

A paper submitted to a journal, describing in detail how to model and analyse the best-case response time of a fixed priority scheduled uniprocessor with preemption.

PAPER B

Martin Sanfridson, “Discretization of loss function for a control loop having time-varying period and control delay”, Technical report, ISRN KTH/MMK/R--03/2--SE, Mechatronics Lab, KTH, May 2004.

A technical report describing in detail the discretization of a continuous time process and quadratic loss function, in case of time-varying sampling period and control delay.

PAPER C

Martin Sanfridson, Martin Törngren and Jan Wikander, “A quality of control architecture and codesign method“, work in progress conference paper at the Real-Time and Embedded Technology and Applications Symposium, Toronto, May 2004.

An extended version of a conference work in progress paper, describing a framework for on-line adaptation of resources with respect to periods and delays.

PAPER D

Henrik Rehbinder and Martin Sanfridson, “Scheduling of a limited communication channel for optimal control“, Automatica, Vol. 30, No. 3, pp. 491-500, March 2004.

A brief paper published in the control theoretical journal Automatica, describing a method to optimize the static processing schedule of optimal controllers.

Other publications

- Chen, D.-J. and Sanfridson M., “Introduction to distributed systems for real-time control“, Technical report, ISRN KTH/MMK/R--98/22--SE, Mechatronics Lab, KTH, November 2000.
- Claesson V., Gäfvert M. and Sanfridson M., “Proposal for a distributed computer control system in heavy-duty trucks“, Technical report 00-16, Computer engineering, Chalmers university of technology, 2000.
- Gäfvert M., Sanfridson M. and Claesson V., “Truck model for yaw and roll dynamics control“, Technical report, ISRN LUTFD2/TFRT-7588--SE, Automatic control, Lund institute of technology, 2000.
- Redell O. and Sanfridson M., “Exact best-case response time analysis of fixed priority scheduled tasks“, Proc. of the 14th Euromicro Workshop on Real-Time Systems, Vienna, 2002.
- Rehbinder H. and Sanfridson M., “Integration of off-line scheduling and optimal control“, Proceedings of the 12th European Conference on Real-Time Systems, Euromicro, pp. 137-143, Stockholm, 2000.
- Rehbinder H. and Sanfridson M., “Scheduling a limited communication channel for optimal control“, Proceedings of the 39th IEEE Conference of Decision and Control, volume 1, pp. 1001-1016, Sydney, December 2000.
- Sanfridson M., “Problem formulation for QoS management in automatic control“, Technical Report, ISRN KTH/MMK/R--00/3--SE, Mechatronics Lab, KTH, March 2000.
- Sanfridson M., “Timing problems in distributed real-time computer control systems“, Technical report, ISRN KTH/MMK--00/11--SE, Mechatronics Lab, KTH, May 2000.
- Sanfridson M., “Timing problems in distributed control“, Licentiate thesis, ISRN KTH/MMK--00/14--SE, Mechatronics Lab, KTH, May 2000.
- Sanfridson M., “An overview of the use of LMI in control to assess robustness and performance“, Technical Report ISRN KTH/MMK--03/08--SE, Mechatronics Lab, KTH, 2003.
- Sanfridson M., Claesson V. and Gäfvert, M., “Investigation and requirements of a computer control system in a heavy-duty truck“, Technical report, ISRN KTH/MMK--00/5--SE, Mechatronics Lab, KTH, June 2000.
- Törngren M. and Sanfridson M. (ed.), “Research problem formulations in the DICOSMOS project“, Technical report, ISRN KTH/MMK--98/20--SE, Mechatronics Lab, KTH, 1998.
- Törngren M., El-khoury J., Sanfridson M., and Redell O., “Modelling and simulation of embedded computer control systems: problem formulation“, Technical report, ISRN KTH/MMK--98/20--SE, Mechatronics Lab, KTH, 2001.

Contents

1	Introduction to the thesis	15
1.1	Problem description	15
1.2	Aim of the research	16
1.3	Research approach	16
1.4	Delimitation of the scope	17
1.5	Background, research projects	18
2	Preliminaries: the timing properties	18
3	Off-line: codesign of computer control systems	21
3.1	Partitioning, allocation and scheduling	21
3.2	The purpose of a performance measure	22
3.3	Making trade-off decisions	23
3.3.1	Choice of period vs. utilization	23
3.3.2	Delay jitter vs. delay to cancel the jitter	23
3.3.3	The choice of priority	23
3.3.4	Skipping a task in case of temporary overload	24
3.4	Jitter in a static schedule	24
3.5	The response time of a fixed priority scheduled task	25
4	On-line: negotiating for resources	27
4.1	The early design and availability of information	27
4.2	Flexibility and modularity	28
4.3	Optimizing the control performance	28
4.4	Quality of control negotiation	29
5	Summary of the appended papers	30
5.1	Paper A: The best-case response time	30
5.1.1	Outline of the paper	30
5.1.2	Contributions	31
5.1.3	Related work by the author	31
5.2	Paper B: A control performance measure	31
5.2.1	Outline of the paper	31
5.2.2	Contributions	31
5.2.3	Related work by the author	32
5.3	Paper C: A Quality of Control architecture	32
5.3.1	Outline of the paper	32
5.3.2	Contributions	32
5.3.3	Related work by the author	32
5.4	Paper D: Optimal scheduling with limited communication	33
5.4.1	Outline of the paper	33
5.4.2	Contributions	33
5.4.3	Related work by the author	33
5.5	A comparison to the licentiate thesis	33
6	Future work	34
6.1	The worst-case gain of a time-varying sampled-data system	34
6.2	The worst-case jitter in a control loop	34
6.3	Expected case response time for fixed priority scheduling	34
6.4	On-line schedulability test and adaptive task parameters	35
6.5	Safety-critical and testing issues for QoC	35
6.6	Work bench for scalability and QoC	35

7	References.....	36
	Paper A.....	37
1	Introduction.....	39
2	The task model.....	42
3	Best-case phasing.....	44
4	The response time calculation.....	48
5	The best-case response time.....	52
6	Pseudo code and an example.....	57
7	Discussion.....	62
8	Conclusions.....	64
9	Acknowledgements.....	64
10	References.....	65
	Paper B.....	67
1	Introduction.....	73
	1.1 Purpose and aims.....	73
	1.2 Methods and tools.....	73
	1.3 A short summary of the theory.....	74
	1.4 Related work.....	75
2	Treating time.....	78
	2.1 The timing properties.....	78
	2.2 The sampled and delayed process, constant period and delay.....	79
	2.3 Simplifying assumptions.....	80
	2.4 Describing time in the discrete model.....	81
3	The discrete Markov chain.....	83
	3.1 Fundamentals of the Markov chain.....	83
	3.2 A causal ordering condition for delays.....	85
	3.3 The periodic chain.....	86
4	Closing the loop.....	88
	4.1 Sampling and actuation timelines.....	88
	4.2 A model of the closed loop.....	90
	4.3 The closed loop state vector.....	92
	4.4 The controller.....	92
	4.5 The process and its discretization.....	93
	4.5.1 Constant sampling period, no delay.....	94
	4.5.2 Constant sampling period and constant delay.....	94
	4.5.3 The general case.....	95
	4.5.4 Multiple changes of the control signal.....	96
	4.5.5 Three special cases.....	98

4.5.6	Calculating the integrals.....	99
4.5.7	The state and measurement noise.....	101
4.6	The closed loop.....	102
5	Discretization of the loss function	104
5.1	The continuous time loss function.....	104
5.2	Discretization, no delay.....	105
5.3	Discretization, constant delay.....	107
5.4	Discretization, the general case.....	109
5.5	Inserting the control signal.....	111
5.6	On the choice of weight matrix.....	112
5.7	Discussion.....	113
6	A performance measure	115
6.1	Covariance and loss of the state and measurement noises.....	115
6.1.1	The state noise.....	116
6.1.2	The measurement noise.....	116
6.2	Covariance in case of a constant sampling period.....	117
6.3	Stochastic jump linear system.....	118
6.3.1	The Markov chain.....	118
6.3.2	Calculating the loss.....	119
6.3.3	The Kronecker and vector notation.....	120
6.3.4	Calculating the state-dependent covariance.....	120
6.4	Periodic systems.....	121
7	Stability and robustness	124
7.1	Stability concepts, second moment stability.....	124
7.2	Coupled matrix equations.....	125
7.3	Robustness to the timing properties.....	127
7.4	Discussion.....	128
8	The effect of a single timing property	130
8.1	The constant sampling period.....	130
8.2	A constant delay in the closed loop.....	133
8.3	Delay jitter.....	134
8.4	Sampling period jitter.....	135
8.5	Transient error.....	136
8.6	Discussion.....	137
9	Applications of the performance measure.....	138
9.1	Constant period vs. constant delay.....	138
9.2	Constant delay vs. delay jitter.....	139
9.3	A comparison between sampling and delay jitter.....	140
9.4	A comparison between input and output jitter.....	141
9.5	The worst and the best types of delay jitter.....	143
9.6	Deliberate delay with compensation.....	144
9.7	The value of making skips.....	145
9.8	Random and periodic delay jitter.....	146
9.9	Periodic schemes.....	147
9.10	Discussion.....	148
10	Summary and discussion of the report	149
11	References.....	152
	Appendix A: Description of the example systems	154

A.1	Damped second order process with a PI controller.....	155
A.2	DC-motor with a state feedback controller.....	156
A.3	Third order process with an LQG controller.....	157
A.4	Non-minimum phase process with an LQ controller.....	158
A.5	First order with delay and Smith predictor.....	159
A.6	Double integrator and deadbeat controller.....	160
A.7	Inverted pendulum with a minimum variance controller.....	161
Appendix B: Validation by simulation		163
B.1	Constant period.....	164
B.2	Constant delay.....	164
B.3	Period and delay jitter, restricted.....	165
B.4	Period and delay jitter, relaxed.....	166
B.5	Periodic schedule.....	167
Appendix C: Some matrix sizes		168
Appendix D: Integrals, delay at the controller's side.....		169
Appendix E: Integrals, delay on the process' side		171
Appendix F: Calculating the integrals.....		173
F.1	System and input matrices.....	173
F.2	Sub-weights for delay at the controller's side.....	173
F.3	Sub-weights for delay at the process' side.....	174
F.4	State noise covariance and additional loss.....	175
Paper C.....		177
1	Introduction.....	179
1.1	Contributions.....	180
2	Architecture preliminaries.....	181
2.1	Application characteristics.....	181
2.2	Definitions.....	182
2.3	Requirements.....	182
3	Architecture for QoC.....	183
3.1	Building blocks, architecture overview.....	183
3.2	Quality estimation and optimization.....	184
3.3	Admission control.....	186
3.4	Negotiation and activation.....	186
3.5	Codesign method.....	188
3.6	Implementation.....	188
4	Example	189
4.1	Scenario.....	189
4.2	Applications.....	190
4.3	Use cases.....	191
4.4	Results.....	191
5	Related work	192
6	Summary and future work	193

7	Acknowledgement.....	194
8	References.....	194
	Paper D.....	197
1	Introduction.....	199
2	Problem formulation	201
	2.1 Sampled-data LQ-control.....	201
	2.2 Periodic systems.....	204
	2.3 Optimality function.....	205
3	Solving the optimization problem	206
	3.1 Structural properties.....	206
	3.2 Optimization heuristics	208
4	Examples.....	209
	4.1 Control design.....	210
	4.2 Optimal sequences	211
	4.3 Further study	211
5	Discussion and summary.....	212
6	Acknowledgement.....	213
7	References.....	213

Wir denken die einzelnen Elemente unseres Gegenstandes,
dann die einzelnen Teile oder Glieder desselben
und zuletzt das Ganze in seinem inneren Zusammenhange zu betrachten,
also vom Einfachen zum Zusammengesetzten fortzuschreiten.
Aber es ist hier mehr als irgendwo nötig,
mit einem Blick auf das Wesen des Ganzen anzufangen,
weil hier mehr als irgendwo
mit dem Teile auch zugleich immer das Ganze
gedacht werden muß.

von Clausewitz, Vom Kriege, 1832

1 Introduction to the thesis

The introduction to the appended papers is organized as follows. After a description of the research problem the aims are stated. The technologies and methods used, together with a brief annotation of delimitations are added to make the picture clearer. Thereafter, a bit of the background in terms of research projects follows. Before the four appended papers are introduced, a brief treatment will introduce codesign of computer control systems, which this thesis very much is about. The introduction is split in three sections: the so called timing properties, the off-line approaches and the on-line approach. This is to give more material to the beginner in this area, but also to help place the appended papers in their proper context. A brief treatment of each appended paper stating its contents, contributions, division of work and related work is given. Some ideas for future research conclude this introduction.

1.1 Problem description

An embedded computer system is physically a set of processing and communication units, which are parts of some machine. The machine is not directly referred to, or thought of, as having a computer system — the embedded computer is a machine element that is dedicated to perform a certain job. The embedded computer system is typical for a mechatronic system, where mechanical and electronic machine elements are integrated.

Vehicles and manufacturing systems are two common examples of mechatronic systems. In many cases, the motions of the mechanical machine is controlled by interconnected microprocessors. The control law, which controls the mechanical system or process, is implemented in software. The control theory is based on the feedback principle: the deviation between a setpoint and a measured value of interest, is used to compute a control law, and the result is then fed back to the physical process via an actuator. Almost from the advent of computers, as computers grew more commercially available, controllers have been implemented in software.

A controller has to react both adequately and timely in relation to its physical environment. The environment is a process which has its own dynamics. This kind of requirement put on a computer system leads to the notion of real-time. The hardware has to sustain a sufficiently high rate of actions, without getting overloaded by the requested utilization, in order for it to respond timely to its environment. A traditional definition of real-time is based on the notion of deadline. In a real-time system, should the processing time of tasks and the transmission time of messages not conform to the deadline specifications, the result could be a system failure. The ability of a computer system to meet deadlines for to the intended rate of actions depends, among other

things, on how fast the computer system is and how well it is programmed. If the timing is predictable, real-time guarantees can be stated.

The timing behaviour of a processing or communication unit can be controlled by scheduling periodically recurring tasks and messages. Problems can often be solved by throwing in more resources or using parts which are fast and state-of-the-art, but such alternatives are usually expensive. The quest for cost-efficiency is an industrial perspective on the research. It is a constraint which calls for an optimization of affordable resources. To achieve cost-efficiency, the use of components-off-the-shelf, such as the hardware, the communication protocols and the operating system, is often proposed instead of a closed proprietary system. The need for speed can to some extent be met by a better real-time scheduling. Although every part of a computer system contributes to the timing behaviour, the scheduling is the most interesting part to study. From a practical point of view, it is desirable that a system designer has a good knowledge of both computer and control engineering. It is vital to understand the consequences of a trade-off in order to make good design decisions. To over-dimension a design without achieving any additional qualities must be considered bad engineering practice.

1.2 Aim of the research

The main objective of the research has been to study the timing aspects of computer control systems from a codesign viewpoint. The stability, robustness and performance of a control loop are affected by the choice of period, the actual delay and any time-variations in both the period and the delay. These timing properties are ultimately the result of design activities such as: the choice of processing and communication hardware, the choice of operating system, the synchronization policy, the scheduling strategy, and the actual execution times of tasks.

The timing properties are useful both at design-time and at run-time. At design-time they help convey vital information between the control and computer engineers. At run-time they can be used as a foundation for optimizing a control related performance measure.

1.3 Research approach

The research described in this thesis is interdisciplinary. The two primary research fields, which are relatively independent, are real-time scheduling theory and feedback control theory. Interdisciplinary combinations are typical for the research in mechatronics. The idea of interdisciplinary is not to unify two fields into one, but to recognize the dependence and combine them both when addressing a problem. However, addressing two disciplines can make the work complex. A traditional approach to master complexity is to break down the system into suitable pieces, analyse the pieces and then study the reassembled system, cf. the quotation on the first page. First, a look at the system as a whole, then each entity apart, and finally all entities should be put together and the entire design reviewed.

An ambition has been to find theoretical models, that describe specific problems and offer some kind of prediction or valuable statement. For example, if the utilization of a fixed priority scheduled uniprocessor is below 0.2, the response time jitter is not likely to be a problem, since the congestion is negligible.

Simulation is a valuable tool when a desired analytical solution is hard to obtain or evaluate. Simulation is also useful for acquiring an understanding of the studied problem and for comparing results obtained by other means. Simulation can be used as a complement to analytical solutions to characterize the average behaviour of a system or a specific scenario, but

it is generally not the right tool when searching for worst-case conditions. As in other modelling approaches, a main difficulty in simulation is to set up a model such that it adequately reflects the phenomenon to be studied. Simulation requires an understanding of the dynamics and in this case also of the timing behaviour. In essence, the problem is to choose the level of details for different parts of the simulation model in order to make good approximations.

Schedule simulation is a pure discrete event simulation, and can be used to evaluate the average performance of a task set, e.g. in terms of latencies. The implementation of the scheduler and dispatcher is preferably done in a high level programming language such as Matlab. The simulation tool Simulink has been used throughout the whole research, since it is well suited for simulation of hybrid systems.

Response time analysis in distributed systems is very complex. Task chains stretch from one node over communication channels into other nodes. It is difficult to obtain the upper and lower limits of the end-to-end response time, which is interesting from control point of view. The end-to-end response time of a controller usually corresponds to the time from the actual sampling to the actual output. Without loss of generality, the end-to-end response time can for convenience be thought of as the response time from the first to the last task in a task chain. It makes sense to start an analysis by studying the response time of a uniprocessor rather than that of a distributed system, since the complexity is lower and the expected impact on the controller is to some extent, via the timing properties, independent of the complexity of the computer system.

1.4 Delimitation of the scope

In order to facilitate an analysis, a useful approximation is to consider linear time invariant (LTI) processes and controllers only. This is a very common delimitation and it is also adopted in this thesis. An LTI can be obtained by linearization of a nonlinear system around an equilibrium point. The analysis of system properties, such as stability, is carried out in the region local to the linearization point. Hybrid systems have mixed continuous and discrete event driven state evolution over time, an evolution which cannot be reduced to either continuous or discrete states only. Non-linear and event-driven systems in combination also belong more to the area of hybrid control. A computer control system is indeed an example of a hybrid system, but by limiting the analysis to LTI systems, the analysis becomes more tractable.

When sampling, anti-aliasing filtering should be applied to the continuous time signal, since frequencies higher than the Nyquist frequency fold and falsely add variance to lower frequencies. However, anti-aliasing filtering is seldom seen in theoretical control papers and is not discussed in this work either. The analog filtering can be treated as dynamics belonging to the process, and the dynamics of the process should be more dominating.

Apart from partitioning and allocation, the implementation details of algorithms have received no attention here. The actual coding with respect to numerical issues must be considered in the codesign. For example, the finite resolution (word length) of a discrete value used to represent a continuous time value, can possess a problem if the sensitivity to quantization effects is high.

A distributed control system typically has control loops with different sampling periods. To start with single rate systems and to extend the study with multirate systems later, is the preferred order. A multi-input-multi-output (MIMO) control system with different delays on every signal, should also be considered for a more general distributed system. A MIMO system is more complex to analyse and can behave quite differently and unexpectedly compared to a single-input-single-output (SISO) system.

1.5 Background, research projects

The DICOSMOS 2 (Distributed Control of Safety Critical Motion Systems) research project was a cooperation between Computer Engineering at Chalmers (CTH), the Mechatronics Lab at the Royal Institute of Technology (KTH), the Department of Automatic Control at Lund Institute of Technology (LTH), and Volvo Technological Development (now called VTEC). The project was funded by VINNOVA (Verket för innovationssystem, the Swedish agency for innovation systems), formerly called NUTEK, and VTEC. The project was a continuation of the first DICOSMOS project, which started in 1993. A case study was a central part of DICOSMOS 2. The aim of the case study was to increase the understanding of a specific safety critical distributed real-time system for control applications: a heavy-duty truck and semi-trailer combination. The case study primarily concerned functions for an electronic braking (ABS) and on top of that, vehicle dynamics control (VDC). The idea was to study and develop interdisciplinary analysis and design methods for this type of system (Törngren et al., 2001).

When the DICOSMOS project ended, the FLEXCON project (Flexible Embedded Control Systems) continued from January 2003. The project is funded by SSF (Stiftelsen för strategisk forskning, the Swedish foundation for strategic research) and runs through 2005. Via ARTES (a national Swedish strategic research initiative in real-time systems) this was a logical transfer. FLEXCON is a cooperation between Automatic control and Computer Science at Lund Institute of Technology, the Mechatronics Lab at KTH, MRTC at Mälardalen University and DRTS at the University of Skövde. The main research direction of the FLEXCON project is the problem of providing flexibility and reliability in embedded control systems implemented with components-off-the-shelf. The key issues are design and implementation techniques that support dynamic run-time flexibility. ARTES/ARTES++ is a network within the real-time research community in Sweden, which among other things provides support for international mobility and arranges an annual summer school. During two months in early 2002 the author had the opportunity to visit the real-time research ReTiS Lab at Scuola Superiore Sant'Anna in Pisa.

2 Preliminaries: the timing properties

A control system is indeed a real-time system. The architecture of a computer control system can be modelled in many different ways, and in this context the timing behaviour is of primary interest. Figure 2.1 shows two views of a computer control system. In the *computer engineering* view to the left, the process P is external to the model of the computer system. In the *control engineering* view to the right hand side of the figure, the dynamics of the computer system are similarly rationalised to allow for a focus on the process P and the dynamics of the closed loop. The input-output (I/O) units constitute the physical interface between the computer and process. At every sampling event, the continuous time signal is sampled by the input unit and after a computation of the control law, the new control signal is held constant, by the output unit, until the next control signal has been calculated. This sample and hold scheme is called zero-order-hold.

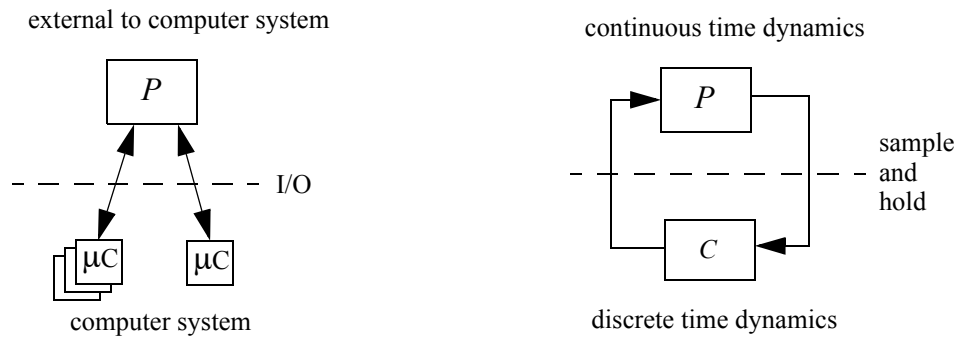


Figure 2.1 Computer engineering view (left) and control engineering view (right). P is the process and C is the controller. μC denotes a processing unit.

The design process is often a team work, but for simplicity of the exposition here the system designer will take on four different roles instead. Each role corresponds to apparent activities in the design process: the control engineer (control analysis and synthesis), the software engineer (architecture design and scheduling), the programmer (implementation) and the service technician (upgrade and repair). The actual design process is always iterative in nature, but this is off the topic.

On the left hand side of Figure 2.2 the architecture of a computer system is sketched. A temporal characterization includes for example the synchronization of tasks and messages, and the scheduling of nodes and communication channels. The temporal characterization is expressed in terms of period, execution time, deadline, response time, transmission delay, target token rotation time, synchronization offset, etc. To the right in Figure 2.2 the control system is represented by a process P and a controller C . The closed loop performance, robustness and stability depend among other things on the timing behaviour of the computer system.

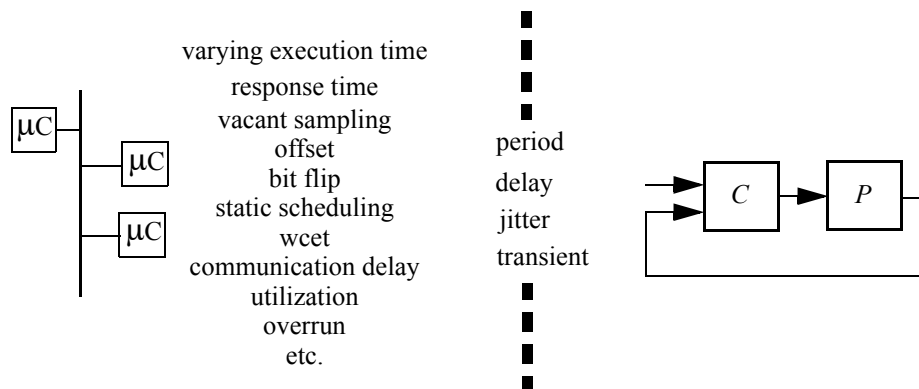


Figure 2.2 The *timing properties* constitute an interface between a model of the timing behaviour of computer control systems (left) and a control theoretical model of the sampled-data system (right).

The timing behaviour of the computer system is described by the so called *timing properties*. This is a temporal characterization of a computer system from a control point of view, see Sanfridson (2000a). These properties must be communicated between the control and software

engineers. The timing properties are:

- *The choice of period.* The periodicity of events in the computer system has a clear correspondence to the theory of discrete control systems. The choice of nominal sampling period is an activity common both for the control design and the implementation with respect to scheduling. The choice of period that the control engineer makes, is often based on some rule-of-thumb regarding the dynamics of the closed loop system. In the traditional view of sampled-data control design, the sampling is uniform, i.e. the sampling period is constant.
- *Constant delay.* A general time-varying delay can be modelled by a constant part plus a time-varying part. A constant delay in an LTI system can conveniently be analysed by familiar control engineering tools. The general rule is that a delay in the closed loop should be as small as possible, because of the decrease in phase margin with increasing delay. The time delay can be inherent in the process, but it can also emerge from the computer system, and thus be called *control delay*. For some control algorithms it is quite easy to compensate for a nominal delay, but allowing for an additional delay brings a cost in terms of control performance.
- *Jitter in delay and period.* Jitter is here defined as the time-variation compared to a nominal constant value. There is a difference between delay jitter and period jitter, since a (continuous) signal (or process) is sampled. When sampling jitter is present, the acquired value depends on the sampling instant. Control delay jitter, on the other hand, causes a delay of the control output to the continuous process. Delay jitter and sampling jitter can of course be present simultaneously. The jitter does not become unintentional because the designer is unable to reduce it. A control designer should of course neglect jitter that is small enough to be of no importance.
- *Transient error.* A transient error is an infrequent change in the normal processing or communication, which causes a long interrupt or delay. The transient error can be the result of a previous fault in the computer system, an example is vacant sampling (or vacant actuation) because of jitter. Transient errors are typically devastating to a system without forgiving dynamics. The transient error can also be deliberate, introduced by skipping tasks or messages to clear the processing unit or communication channel from a temporary overload. A computer system is prone to malfunctions which permanently or temporarily can suspend the normal periodic execution of a control algorithm. A mishap can for example be caused by a memory bit flip, an omitted message or an overrun due to overload. These situations typically occur very seldom, compared to the dynamics of the closed loop, hence the term transient (timing) error. There are many ways to counteract transient errors by redundancy in space and time. An implication of the infrequent occurrence is that a steady state behaviour is less interesting to analyse, compared to the short term effects of a transient error.

The timing properties should not be regarded as problems which need to be solved, but rather as attributes which are important to understand. The period is often chosen deliberately, but the delay and jitter are traditionally something that results from several design choices. Jitter is practically always present in a computer system, but it is often small enough to be neglected. Minor jitter could for example stem from hardware devices such as A/D samplers, pipelines, caches and clock circuits. Larger time-variations, arising from the synchronization and scheduling of tasks and messages are likely to dominate, and they are of course more interesting. Severe jitter might arise when the utilization of a processing unit or communication channel is

high, but this also depends on the scheduling strategy.

Knowing the cause and effect of the timing properties is useful at *design time* to make improvements and trade-offs, and can help the system designer and control engineer to choose a computer architecture or a scheduling strategy from a control point of view. It is obviously relevant to estimate the impact of the timing properties in order to evaluate alternative choices. For example, the most prominent trade-off is to choose sampling period, since it affects both the behaviour of the control loop and the actual utilization of the computer. In design, there is always a multitude of aspects to consider and the aim to optimize a solution is a central theme in any design work.

There is also a *run-time* use of the timing properties, in terms of dynamic scheduling and on-line optimization. The idea is to design an architecture that incorporates some knowledge of the control performance with respect to the timing properties. A dynamic (on-line) approach is more flexible, since decisions are partly made at run-time instead of at design-time. One fundamental difference between run-time and design-time is the availability of information regarding details of the timing behaviour. Execution times, end-to-end latencies and other computer related timing measures can be difficult to estimate pre-runtime. At the best, this information becomes available at a late stage and possibly also becomes more accurate when more design decisions have been settled. This on-line approach is here called quality of control, QoC.

3 Off-line: codesign of computer control systems

This section about codesign of computer control systems at *design time* and the next section devoted to *run-time* issues, serve as an introduction putting the appended papers in their context.

3.1 Partitioning, allocation and scheduling

The goal is to find an architecture for the intended logical and temporal functionalities of the dedicated computer system. Prior to a scheduling of tasks and messages, the software engineer has to both determine the task set, and to do this for each processing, communication or other unit of the distributed system to be scheduled. The first of these activities is called partitioning the latter is called allocation. The partitioning and allocation depend very much on the application, the hardware at hand and on the logical functionality.

In the *partitioning* part of the design process, the software is split into a number of tasks. Their logical and temporal relationship (precedence and exclusion constraints) can be described by constructing a so called task graph (also called directed graph). The partitioning of a control task can be made following a suggestion in Åström and Wittenmark (1997), in order to minimize the latency from sampling to actuation. The idea is to partition updates of the controller's states in a separate task, which can be executed after the actuation instant but before the next sampling. Thus, the partitioning decisions will affect the timing properties.

The *allocation* is of course intimately associated with the topology of the computer system and the task graph. For example, in a distributed control system, a sensor reading and ensuing signal processing can be allocated to a dedicated sensor node, the controller and the actuator can be allocated to other nodes, and their messages allocated to a common bus. Clearly, the allocation decision will affect the timing properties.

The next step — the *scheduling* — can be studied separately from the partitioning and

allocation, if given a set of tasks, a task graph and the allocation structure. An exception is utilization balancing strategies. The scheduling is less application specific than the partitioning and allocation, which makes real-time scheduling stand out as a discipline of its own. Because of different physical prerequisites, communication channels, uniprocessors, multiprocessors and distributed systems require different real-time scheduling strategies. For example, CSMA and Ethernet are fundamentally different and the scheduling strategy should be chosen accordingly. Clearly, the scheduling activities will also affect the timing properties.

3.2 The purpose of a performance measure

Assume that all information concerning partitioning, allocation and scheduling is available to the software engineer. The software engineer is then, to some degree, able to figure out the timing properties, e.g. the end-to-end response time of a control loop's task chain. In order to evaluate and compare solutions, a measure of the relative goodness is needed. This ultimately links the role of the software engineer together with the role of the control engineer. A traditional delimitation imposed to avoid the potential complexity of this relation, is to state that (a) there is no delay jitter, (b) there is no sampling jitter and (c) transients errors do not occur. What remains is the choice of sampling period. With a performance measure, these constraints can be relaxed e.g. by specifying limits of allowable delays and periods, (Törngren, 1995 and Redell et al., 2004).

There are many ways to construct a performance measure. When tuning a controller, the control engineer decides how it will react to disturbances and reference changes. Usually, there is not only one tuning method, or one set of parameters, but many, that will render a good result. The reason is that control performance is to some extent subjective. The situation is similar for the synthesis when selecting the structure of the controller. One of the simplest tuning methods is to look at overshoot and rise time. However simple and effective this might be, it is less applicable to a more automated procedure. Instead, it is natural to apply statistical measures to assess the control performance. A synthesis method frequently encountered in the control research community is optimal control. This method gives the control parameters that will minimize the variance according to a specified objective function, also called loss function. A traditional choice is the integrated square of some signal of the control loop. This measure of energy contents (or variance) is the foundation of a major design theme besides pole placement. The common loss function has the following structure:

$$\bar{J} = E \lim_{T \rightarrow \infty} \frac{1}{T} \int_0^T \begin{bmatrix} x_p(t) \\ u(t) \end{bmatrix}^T \begin{bmatrix} \bar{Q}_{11} & \bar{Q}_{12} \\ \bar{Q}_{12}^T & \bar{Q}_{22} \end{bmatrix} \begin{bmatrix} x_p(t) \\ u(t) \end{bmatrix} dt \quad (1)$$

The scalar loss \bar{J} depends on the weight matrices \bar{Q}_{11} , \bar{Q}_{12} and \bar{Q}_{22} , which are the parameters of the loss function defining what is optimal. Note that the symbol T is used both to denote a time period and the matrix transpose, but there should be no risk of confusion. The signals punished are the state vector of the process $x_p(t)$ and control signal vector $u(t)$, coming from the controller to the process. The objective function (1) is often seen, but a loss function can also have a completely different structure and include the weighted variance of other signals. As a simple example, consider the variance of the process output $y(t) = C_p x_p(t)$.

A measure of type (1) is applicable even when the system varies with time because of jitter. However, it is appropriate only to measure the average behaviour. When it comes to infrequent transient errors, it can be an ambiguous measure since a transient effect also can cause a

violation of the allowable state space. This error condition is not necessarily detectable by a measure based on the variance. In robust control, the worst-case gain is typically used instead, or as a complement.

A discrete loss function corresponding to (1) can be written

$$J = E \frac{1}{N} \sum_{k=0}^{N-1} \begin{bmatrix} x_p(t_k) \\ u(t_k) \end{bmatrix}^T \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12}^T & Q_{22} \end{bmatrix} \begin{bmatrix} x_p(t_k) \\ u(t_k) \end{bmatrix}$$

In case of uniform sampling, it is enough to consider only one sampling period and the average over N periods is superfluous.

3.3 Making trade-off decisions

Here follows a few examples of situations where it can be valuable to make a trade-off in the codesign.

3.3.1 Choice of period vs. utilization

In case the capacity of a processing or communication resource is limited, the choice of period becomes vital. Typically, the performance increases with a decreasing period, but the marginal cost in terms of utilization increases fast for short periods. This trade-off can be cast as an optimization problem if two or more fundamentally similar applications use the same resource.

3.3.2 Delay jitter vs. delay to cancel the jitter

From a control point of view, the performance, robustness and stability are adversely affected by an increased delay in the control loop. At the same time, the control delay jitter is usually also harmful. When implementing the control law, the software engineer (or programmer) should not introduce unnecessary delay. However, an offset between the start of a periodic producer and the start of a periodic consumer can be used for synchronization. The offset should be longer than the worst possible finish time of the producer to avoid a vacant sampling. The drawback is that the offset represents a delay in the task chain and in the loop. The consumer becomes time triggered (TT) by this offset, which can simplify the scheduling algorithm. The jitter is effectively cancelled by this type of buffering. It is clearly of interest to investigate the impact of delay jitter to compare with the cost of an additional offset. A question is how this offset should be selected in an optimal way.

3.3.3 The choice of priority

The actuation should as a rule of thumb follow as soon as possible after a sampling instant. This calls for an event triggered (ET) synchronization, rather than inserting an offset (skew) between the producer and consumer to substitute the precedence and exclusion constraints by a temporal constraint. Consider the previous Section 3.3.2 with an ET consumer. In fixed priority scheduling, a low priority task is prone to jitter because of interference with high priority tasks. Selecting the highest priority will eliminate the problem (but it does not eliminate any blocking). This greedy solution does not work if other tasks need high priority too. Hence, there is a trade-off to be made.

3.3.4 Skipping a task in case of temporary overload

Again, consider the previous Section 3.3.2. The actual delay jitter can be modelled as a stochastic process. Typically, the average delay is considerably smaller than the worst delay — the probability distribution has a long tail — which makes the offset (or deadline) conservative. The violation of the deadline is a sign of temporary overload and a traditional solution is to skip the task. If skips are allowed, the offset can be decreased. Applying a strategy with skips typically decreases control performance and shorter delay increases it, hence there is a trade-off situation.

3.4 Jitter in a static schedule

The trade-off between the timing properties is also relevant for static scheduling. Static scheduling is above all an off-line scheduling method. It is applicable to the scheduling of uniprocessors and communication channels, i.e. to distributed systems. For a broad range of embedded real-time systems, it makes sense to assume that the running software will remain unmodified for long periods of time. The scheduling strategy is quite simple: every task has pre-allocated time slots in a schedule, which is repeated cyclically in eternity. The pattern is constructed pre-runtime and off-line where there is plenty of time to find a feasible or optimal execution pattern. The static schedule can for example be elaborated to encompass mode changes by switching schedules on-line or it can be combined with a dynamic scheduling strategy, to fill non-allocated slots in the, so to speak, background of the primary tasks.

A limited communication channel can be scheduled statically. Many communication protocols are based on a non-preemptive transmission of a predetermined amount of data. In the absence of contention, this gives a possibility to split the transmission into time slots. If a set of periodic messages is to be scheduled, an optimization objective based on for example the latency and the period, can be constructed. Notice that the number of time slots between any two transmissions of the same message do not have to be uniform; jitter can be deliberately built into the schedule. From a control perspective, it is possible to take the known jitter into account in the control synthesis.

In Figure 3.1 an example task set with two messages τ_1 and τ_2 is scheduled. τ_1 is given every second slot to transmit a message, and for τ_2 two alternatives can be compared. τ_2^1 shows an allocation according to a so called harmonic rate pattern. This limits the choice of period, but there will be no jitter. On the other hand, if jitter is tolerable the period can be chosen freely, e.g. as shown by the second alternative τ_2^2 . The down pointing arrows denote arrivals, assuming input at the beginning and output at the end of a slot.

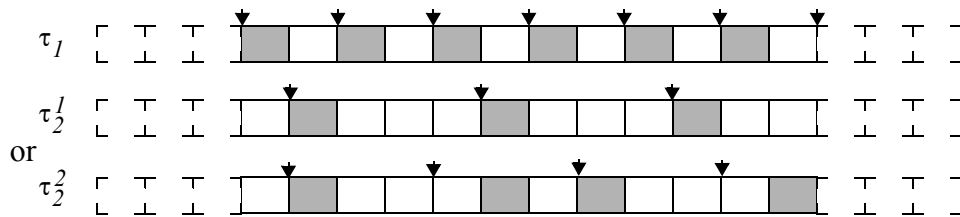


Figure 3.1 A static schedule of two tasks. Top: Every second (2nd) slot is allocated for task τ_1 . Middle: Every fourth slot is allocated for τ_2 . Bottom: As an alternative to τ_2^1 , every third slot is intended to be allocated for τ_2 .

The deterministic static schedule brings predictability and with that, a number of advantages. The absence of unknown jitter makes testing easier, since the number of combinations of possible states of the system is kept low compared to a less predictable dynamic scheduling strategy. By the same reasoning, the error detection coverage is likely to be higher and easier to construct. Further, independent tasks are also temporally isolated from each other, such that a misbehaving task cannot affect the timing of the other tasks.

An idea is to search for an optimal or at least a feasible schedule. In the context of this thesis, an objective function is naturally based on control performance, but it could also be a first-hand measure such as the end-to-end delay. There is nearly an infinite amount of time available pre-runtime, to tune the synchronization of tasks in order to meet the objective. Roughly speaking, this means that — given a fixed task set — a dynamic scheduling strategy cannot make a better job than a static scheduling. The optimization problem is in general intractable.

The greatest advantage of the static scheduling strategy is also its major weakness. For a pure static schedule the drawbacks are primarily the inflexibility, but also the potential under-utilization due to periodic scheduling of non-periodic tasks. In this respect, a dynamic scheduling strategy can do better.

3.5 The response time of a fixed priority scheduled task

Fixed priority scheduling (FPS) is above all an on-line strategy, which is dynamic in the sense that tasks are scheduled at runtime. It means that a set of running tasks can be changed on-line. FPS is applicable to the scheduling of uniprocessors and communication channels, i.e. to distributed systems. FPS has been studied for a long time in the real-time scheduling research community. Just to mention, another well-known dynamic scheduling strategy is the earliest deadline first, EDF. There are many more scheduling strategies other than FPS and EDF, see e.g. Krishna and Shin (1997).

During the execution of a task, higher priority tasks can arrive and gain access to (i.e. preempt) the shared resource, making a context switch. This suspends the execution of the preempted task and thus postpones the time instant it will finish. The *response time*, i.e. the time interval between the start and finish of an instance of a task, will in general vary from one instance to the next. Under some general assumptions, it is possible to calculate both a lower and an upper bound of the response time. These bounds, the *best-case response time* and the *worst-case response time*, are both useful when characterizing varying delays when FPS is used to close a control loop. In addition to this, the *expected response time* is valuable. This is in contrast to traditional analysis of FPS systems, where only the worst-case response time is of interest, where a (hard) deadline typically is the sole requirement specification of real-time behaviour.

The scheduling starts by defining the periodic task set. Each task in the set has in its simplest case the following parameters: period, worst-case execution time (WCET) and priority. The period is chosen according to the needs of the application. The execution time depends on both the hardware and the software, and the actual execution time typically varies from one instance of the task to the next. An estimate of the WCET is needed for the worst-case response time analysis. The problem of finding a WCET is a research subject on its own. Suppose that the execution time is the same for each instance of a task. If we take one task instance and delay its current arrival (and the arrival of all its future instances) a short time, the phasing relative to the other tasks changes. This will change the resulting scheduling pattern. As a matter of fact, the act of scheduling a FPS task set is to a large extent deciding upon the relative phasing (or offsets) between tasks. When the phasing is undecided, the task set can be called “unscheduled”. To

achieve hard real-time guarantees, a schedulability test must be performed pre-runtime, i.e. before the task set starts to run. This is often done off-line, since the task set might need to be adjusted manually.

In control, we are not only interested in the extreme values that the response time can take. Transient timing errors should of course be avoided, but the performance of a control loop also depends on the actual behaviour of the delay. The average behaviour over a long time period decides the average control performance. There is a need to analyse the scheduling both from worst-case design and performance optimization viewpoints. Embedded control systems are often distributed. Further, tasks depend logically on each other. These factors complicate the analysis and provide challenges for future work.

It is possible to take on a stochastic view to estimate the number of preemptions made by higher priority tasks. The stochastic approach presented here is not the right way to derive the response time analysis formally, but it can provide some insight. Tasks having a priority lower than the studied will not interfere since they are immediately preempted. Let the time-varying R_i be the response time of the studied low priority task with priority i , and view R_i as a stochastic process. A higher priority task, with priority j , period T_j and constant execution time C_j , will interfere and preempt the low priority task R_i/T_j times on the average. The high priority task is thus expected to execute $(R_i/T_j)C_j$ time units during the interval R_i . The studied task executes C_i time units during R_i . When repeated for all higher priority tasks $hp(i)$ this yields,

$$R_i = C_i + \sum_{\forall hp(i)} \frac{R_i}{T_j} C_j \quad (2)$$

which can be interpreted as the expected-case response time (ECRT). It is tempting to view also the execution time as a stochastic process.

The calculation of the *worst-case response time* (WCRT) of a fixed priority scheduled task is a well-known problem and the simplest case with independent tasks on a uniprocessor has been extended in many directions. The worst-case response time analysis is based on a model called the worst-case phasing scenario, where all higher priority tasks arrive and are released simultaneously, at the so called critical instant of the studied low priority task. Since the phasing of every single task relative to another task is determined in this way, the worst-case response time can be calculated in a straight forward manner by a recurrence equation. A *worst-case scenario* is formed when all higher priority tasks $hp(i)$ arrive and are released simultaneously with the studied task i , at the so called *critical instant*. The WCRT is given by

$$R_i^w = C_i + \sum_{\forall hp(i)} \left\lceil \frac{R_i^w}{T_j} \right\rceil C_j \quad (3)$$

where $\lceil \cdot \rceil$ denotes the ceiling operator. The equation has to be solved iteratively.

The *best-case response time* (BCRT) analysis is dual to the worst-case analysis. A model of the phasing of tasks called the *best-case phasing scenario*, renders the best-case response time. This fact can be used in the control design. Similar to the previous recurrence equation, the best-case response time can be written,

$$R_i^b = C_i + \sum_{\forall hp(i)} \left\lfloor \frac{R_i^b}{T_j} \right\rfloor C_j \quad (4)$$

where $\lfloor \cdot \rfloor$ denotes the floor operator. (4) is solved in pretty much the same way as (3). In order to estimate the BCRT, the best-case execution time (BCET) for every task is needed.

The worst-case response time is important in real-time scheduling, because it can be used to check that deadlines are met. The best-case response time is less useful in this respect, but it does give some further information about the delays in the system.

4 On-line: negotiating for resources

This section is a general introduction to quality of control. The motivation of this design architecture starts off-line. An increased number of functionalities in an embedded distributed system threatens to lead to an increased number of nodes. For example, today's mid-range car has several tens of nodes and the number is seemingly increasing for every new model. Such a trend is not sustainable in the long run although many functionalities need dedicated hardware. Today, flexibility in the automotive industry, is apparently upheld by using communication protocols, such as CAN. The protocols hide the implementation of a node, which facilitates modular development of subsystems typically undertaken by subsuppliers. The network acts as an isolator, an indisputable hardware interface, to diverse working groups each with its own node (Electrical Control Unit), programming language and real-time operating system. A node cannot always be equal to a functional module. At some point, cooperating development teams have to abandon the well defined interface typical of a communication protocol. Another set of firm rules, providing isolation between functionalities, is needed.

4.1 The early design and availability of information

The scheduling and response time analysis are typically done at a late design stage. In this context, late means that other activities have to be undertaken first. The timing behaviour depends on many things, such as the choice of micro controller, the communication hardware, the protocols and the operating system. It also depends on the actual task set to be scheduled, the scheduling strategy, the synchronization, the estimation of the execution times, etc. The lack of temporal isolation is annoying, since a even a minor change of one parameter could propagate and affect the timing behaviour of the whole system. It stands clear that an accurate analysis of the response time must necessarily be made at a late stage in the design, when the input to the analysis is more accurate and the design is no longer subject to changes.

The fact that real-time behaviour is hard to predict is a drawback during the whole design, since early design decisions generally are more important than late ones. Decisions taken early are important since they shape the future work, diminishing the solution space. Design decisions are often taken despite a lack of information. This is a well-known problem for every design methodology, and can be illustrated as in Figure 4.1.

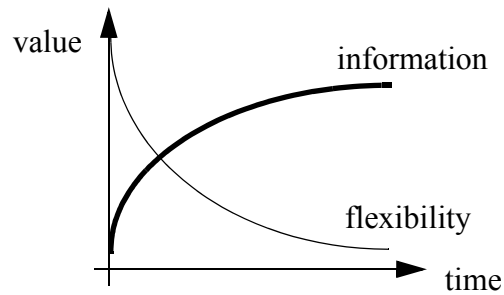


Figure 4.1 Availability of information versus the flexibility to cancel and change decisions made early in a project.

The flexibility to change a decision decreases as time progresses, due to e.g. cost, available man hours and the business aims of a short time-to-market process. To overcome this problem, the available information must be used extensively to predict the outcome of a decision at an early stage. An idea is to compress the development time and collect more information, e.g. by rapid prototyping. From a control engineer's viewpoint, an early estimation of the timing properties in terms of average performance, but also the extreme situations, would be useful.

4.2 Flexibility and modularity

Flexibility is a highly desirable property of an embedded computer control system. Flexibility is often taken for granted in local area networks where bandwidth is more important than guaranteed real-time behaviour. One aspect of flexibility is scalability, e.g. the ability to add and remove hardware and software units without having to redesign the system from scratch.

The flexibility is in many respects more to the customer's or the service technician's advantage rather than to the control or computer engineer's. The modularity concept can be applied to cater for a customer oriented assembly from a family of modules, or it can be useful for upgrades and replacements of parts during the whole life time of a delivered product. A benefit of a modular design is the simplified assembly of customized products, which can reduce the lead time.

There is an initial cost in design effort associated with constructing a modular system. The man hours devoted to redesign and service can be compared against the production cost in terms of processing elements and e.g. memory. As a comparison, consider the use of a high level language or operating system. They both seemingly add to the production cost since more memory and faster processing units are needed compared to assembler on a bare bone system. However, they are highly accepted since they help free the programmer from the complexity of mixing low level implementation details with application specific problems.

4.3 Optimizing the control performance

For many scheduling policies and general task sets, the *expected latency* is considerably shorter than the *worst-case latency*. The expected performance of a control application, i.e. the average performance over an infinitely long time horizon, compared to the dominant dynamics of the closed loop, depends on the expected period, latency and their jitter — and not on the worst-case period or latency (unless these cause instability). However, control applications are often safety-critical and designed to work within a certain state space, and violating specified operating limits is equivalent to system failure. This is the familiar engineering problem of cost efficiency

versus worst-case design. Despite the ability of sustaining a certain amount of timing errors due to the internal memory of a dynamic system, a timing error such as a transient or intermittent vacant sampling/actuation, can be fatal if the allowable state space cannot be upheld. This constraint on the objective of optimizing control performance must be guaranteed. It is typically manifested by the use of hard deadlines. A rational strategy is to minimize the average end-to-end latency or the task period to improve control performance, while still meeting task deadlines. For the same reason, it is necessary to know every task and message in the system, including their worst-case execution times, such that the system is analysable. The worst-case response time needs to be predicted to avoid a deadline miss and the average latency needs to be predicted for the optimization. Unpredictable overload situations cannot be accepted since this could, via excessive response times, lead to an uncontrollable behaviour in the short term.

4.4 Quality of control negotiation

A research hypothesis is that it is advantageous if some work traditionally done at design time could be transferred to take place when the system is running. The abstraction of a quality, as perceived by an application, plays a central role here. Since control applications are the primary design objective, the approach is hereby called Quality of Control, QoC. As the name suggests, ideas have been adopted from the notion of Quality of Service, primarily found in the research areas of multimedia and tele communication. QoC is defined as a method of balancing the demands, primarily of control applications, based on an estimated quality, in order to find a feasible or an optimal use of the available (scarce) hardware resources. The supervisory QoC negotiation is cascaded on top of the applications to negotiate above all period and latency with respect to a measure of performance. The negotiation can be regarded as a feedback loop closed around the controller and its process. The term negotiation is used, rather than optimization, to stress the relative independence of the involved applications.

The benefit of giving one application “more hardware resources” compared to other applications, should be balanced to the needs of all applications. A control application is intrinsically a real-time system, but it is not the only class of application in an embedded system. It is necessary to handle other classes of applications typical for the studied systems, e.g. alarm events, logging and operator interface. An embedded system is rarely altered, i.e. new applications seldom arrive and running ones are seldom removed, compared to the rate of the periodic applications. Frequent changes to a system, with an execution overhead of supervisor logic and optimization, is a minor issue here.

Via the quality measure, an application is to some degree decoupled from the issues of availability and capacity by the ability to accommodate to changes in the processing resources, and in the environment external to the computer system. The decoupling can help facilitate redesign and upgrade during the life time of the product or improve modularity of a product family. The adaptive scheduling procedure is functionally and temporally separated from the applications, and is designed as a negotiation to optimize overall satisfaction.

The negotiation strategy will depend on the scheduling and synchronization policy and on the operating system, but also on the ideas of how the perceived quality of two different applications should be mixed. Two vital parts of the negotiation are admission control and overload handling. The negotiation is a *long term* quality optimization. However, the timing properties being of a *short time* scale, i.e. at the scheduling level, must also be satisfied. In a multi-layer system, layers can have different objectives but they contribute to the same goal. The negotiation is associated with flexibility, and the scheduling with predictability.

5 Summary of the appended papers

The thesis is divided into four papers, Paper A to Paper D, following this introduction. The organization of the dissertation and the main directions of the research are illustrated by the road map in Figure 5.1.

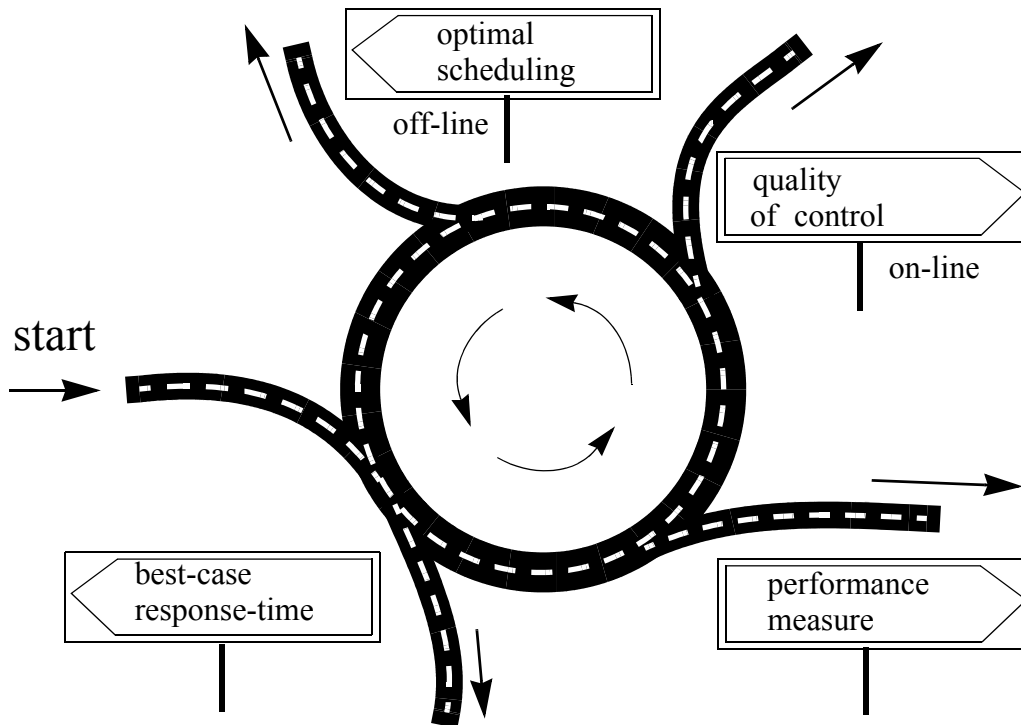


Figure 5.1 Road map to the thesis.

In this section each of the appended papers will be introduced: its contents are outlined, the contributions described and related work done by the author is presented. The division of work between the author and the other contributors is described as well.

5.1 Paper A: The best-case response time

5.1.1 Outline of the paper

This paper has the title “Exact best-case response time analysis of fixed priority scheduled tasks with arbitrary response times” and has been submitted to a journal. The proposed best-case response time (BCRT) calculation is used to obtain a lowest bound on the possible response times of a task within a periodic fixed priority scheduled (FPS) task set. This is useful for synchronization issues, admission control and control design. The response time jitter can be defined as the difference between the worst- and the best-case. It is of interest when it comes to a control theoretical analysis of the end-to-end latencies in a computer control system. The best-case response time has been very little studied, since the application of it is minor compared to the worst-case response time. The analysis and derivation of a recurrence equation to calculate the best-case response time, is based on the here coined *best-case phasing scenario*. The approach is dual to that of finding the worst-case response time using the worst-case scenario,

a well-known piece of theory in the real-time community, extended and refined by many researchers. This BCRT theory now covers the case when released instances of the studied task can interfere with each other within a *busy period*, which is possible if the worst-case response time (WCRT) of a task is greater than its period.

5.1.2 Contributions

The analysis method is by the authors' best knowledge an original piece of theory. The dualism to the worst-case analysis extends the theory of FPS in a beautiful way. The exact best-case response time analysis was previously based on crude and costly searching, and not on a best-case phasing scenario. The idea that the best-case response time could be calculated in pretty much the same way as the familiar worst-case response time, came to the author when working with the problem of stochastic analysis of response times in distributed systems for the DICOSMOS case study. A fresh stochastic view of the traditionally deterministic response time analysis combined with thinking in terms of schedules led to the unexpected finding. Ola Redell immediately grasped the idea and together we quickly formulated a proof and wrote a conference paper. During a two months stay in Pisa, the author had time to review the case when instances of the studied task can interfere with each other — a problem previously set aside. Again, together with Ola the findings were formalised into firm theory.

5.1.3 Related work by the author

The paper is an extension of the earlier conference paper by Redell and Sanfridson (2002). The theory was first presented at the DICOSMOS project's final meeting in December 2001. Theory from the unpublished manuscript of the current paper is also found in Ola Redell's dissertation (Redell, 2003).

5.2 Paper B: A control performance measure

5.2.1 Outline of the paper

The paper has the title “Discretization of loss function for a control loop having time-varying period and control delay“. The timing properties is a proposed model of the temporal interface between control and computer engineering. These are the timing matters that control and system engineers need to consider at design time, and the information that is needed to supervise performance at runtime in a QoC architecture. The paper is an attempt to quantify the impact of the timing properties. A continuous time loss function is discretized together with a continuous time process and combined with a discrete time controller. The report contains a very comprehensive theoretical derivation of the measure, including for example the necessary background theory of jump linear systems. It also contains examples of how severe the influence of different timing properties could be, as well as typical and interesting situations where the measure can be applied.

5.2.2 Contributions

A contribution made in the paper is the discretization of a continuous time process for the case of multiple arrivals of a control signal between two sampling instants. A continuous time loss function is also discretized for this case. The discretization method proposed is a natural extension of the traditional text book version, in which a fractional constant delay splits the constant sampling period into two. The jump linear system model is a vital part of the

discretization. The proposed theory enables an analysis of different timing properties simultaneously, including long delays, in a sampled-data system. It also enables the considered jump linear system to be cast as a linear matrix inequality (LMI). The extended discretization of the process with multiple arrivals, combined with a stochastic modelling of delays and periods, is an original approach and the author is not aware of any identical work. The discretization of two loss functions, when there is (time-varying) delay, is also an original work.

5.2.3 Related work by the author

The current paper is the result of a continuing work towards a performance measure, a work started by a paper on QoC that can be found in the licentiate thesis (Sanfridson, 2000a). In that work, a loss function was used as a performance measure in the negotiation of periods and priorities. The current report was also preceded by a self-study course (Sanfridson, 2003b). A brief version of the rather lengthy paper has been submitted to a control conference.

5.3 Paper C: A Quality of Control architecture

5.3.1 Outline of the paper

The paper has the title “A quality of control architecture and codesign method“. Quality of control can be described as a supervisory on-line negotiation. In an embedded system, there are many applications other than controllers. Also, the kind of information available, its freshness and accuracy are typically better at run-time compared to design time, and this fact should be exploited. A dynamic scheduling strategy where it is possible to alter e.g. periods, offsets and priorities, is necessary. The flexibility resulting from a late but dynamic coupling of modules, is primarily intended for the customer, to provide e.g. management of modules within a product family, easier maintenance and upgrade, but also graceful degradation in case of subsystem failure. The proposed architecture aims to organize the work of the control engineer, system engineer and programmer. It rests on a simple hierarchical structure for the applications based on elementary functions and specifications of the applications. The time scale of the negotiation, admission control and optimization routine, is considerably longer than that of the applications, scheduling and overload handling.

5.3.2 Contributions

The analysis work on QoC is more based on conceptions and descriptions on “higher level” than on mathematical models, if compared to the other papers in this dissertation. The contribution is primarily the view of modularity and flexibility in embedded computer control systems, with the need to design an architecture that supports applications evolving during their whole life time. Further, it is necessary to separate the long time-scale optimization for quality from the short time-scale scheduling for deadlines. It is still necessary to make allowance for traditional hard deadline real-time guarantees. In the present work, the co-authors helped refining and expressing the ideas of scalability and quality of control.

5.3.3 Related work by the author

An architecture for QoS in control applications, QoC, has been developed gradually. The first papers are the problem formulation (Sanfridson, 2000b) and a paper found in the author’s licentiate thesis (Sanfridson, 2000a). In the latter paper a CAN bus was shared with three similar control tasks. The bus was simulated on bit level/frame level to get the right timing behaviour.

The resulting delays and periods were used to compute a quadratic performance measure. Two master theses have been initiated and supervised by the author, (Österman, 2001 and Sjunghamn, 2003). These have been carried out in cooperation with the company Enea Real-Time Systems in Stockholm. The target system was a three link robot arm, each link with a micro controller running Enea's operating system OSE Epsilon and communicating by means of a CAN bus.

5.4 Paper D: Optimal scheduling with limited communication

5.4.1 Outline of the paper

The paper has the title "Scheduling of a limited communication channel for optimal control". The paper is based on the so called periodic Riccati equation. The periodic Riccati equation is found by first lifting the system and then solving an ordinary algebraic Riccati equation to get a stationary solution. This also gives a time-varying optimal controller that compensates for the time-varying delay. The periodic Riccati equation is applied in the static scheduling of a scarce hardware unit, e.g. a communication channel. The following problem can be formulated: given a limited hardware unit and a set of control applications, in what sequence should the controllers use the resource in order to optimize the overall benefit? The resulting schedule is optimal in the sense that it minimizes a weighted LQ loss of participating control loops. The loops are independent apart from the sharing of the uniprocessor or communication channel. The control period for each controller is not presumed beforehand. The control signal fed into the actuator may experience delay jitter. The uniform sampling period defines the slot length of the static schedule. Naturally, the computational complexity of the optimization problem is very high.

5.4.2 Contributions

A contribution of this work is the choice of control period. Unlike traditional scheduling and control codesign, the period is not an input parameter but falls out from the optimization. Further, since the actions of one control loop, as governed by the resulting static schedule, not necessarily are equidistant in time, the notion of period is truly an abstraction. It was Henrik Rehbinder's idea to apply this mathematical tool to the scheduling of controllers. Together we settled upon static scheduling, because it is intrinsically periodic and conceptually clear, allowing a focus on the control theoretical parts.

5.4.3 Related work by the author

The present paper is the last of three of papers on the same theme developed in Rehbinder and Sanfridson (2000a) and Rehbinder and Sanfridson (2000b). Different ways to work around the complexity by reducing combinations of sequences that will render the same or similar result have been tried. All papers have been written together with Henrik Rehbinder and a longer version of the present paper is found in his PhD thesis (Rehbinder, 2001).

5.5 A comparison to the licentiate thesis

The licentiate thesis (Sanfridson, 2000a) was finalized in May 2000 during the DICOSMOS project. The contents of the present thesis compared to the licentiate thesis is as follows. The best-case response time analysis, Paper A, is completely new. Instead of a state-of-the-art report on the timing properties in the licentiate thesis, there is Paper B which describes a performance

measure for the timing properties. The QoC architecture in Paper C is a continuation and generalization of a paper in the licentiate thesis, which focused on a specific hardware setup. The optimal scheduling of Paper D is a refinement of the optimization strategy compared to its predecessor found in the licentiate thesis.

6 Future work

In this section a few ideas of possible future work, to be a continuation of the presented thesis, are outlined.

6.1 The worst-case gain of a time-varying sampled-data system

The weighted performance measure in paper B is related to the H_2 -norm. The measure can be said to give the average energy contents of the system. This is less applicable when it comes to robustness issues such as coping with transient timing behaviour. In that case, the gain or the H_∞ -norm is essential to investigate. This is well-known theory for continuous time multivariable systems (Skogestad and Postlethwaite, 1996) and some work has also been done on discrete time systems (Scherer and Weiland, 2000) and possibly also for multi-rate systems. An open question is if LMIs can be used to study the H_∞ -problem, for the case of time-varying sampling and delay. The discretization presented in paper B was developed with this in mind.

6.2 The worst-case jitter in a control loop

Delay jitter is naturally modelled as a stochastic process having some distribution. For example, the delay can vary between a low and high value, it can be modelled as varying according to a Markov chain, or it can be modelled as a uniform random variable. In the latter case, one parameter is enough to describe the jitter and it can easily be plotted in a diagram, which is convenient for the designer. If the distribution can be associated with a single variable, e.g. the worst-case delay, or the delay that gives the worst performance, a simple rule-of-thumb could possibly be constructed. Assume for example that the best-case, average-case and worst-case delays are given from a response time analysis. The actual timing behaviour of the delay jitter is typically time-varying and hard to characterize. Now, if the worst-case kind of jitter is known, the closed loop can be analysed from a worst-case perspective. This motivates a further study of the worst-case probability distribution of jitter.

6.3 Expected case response time for fixed priority scheduling

The control performance, robustness and stability depend on the actual timing behaviour, which can be characterized better by the average behaviour rather than the worst- or best-case. The response time analysis should thus include a notion of the expected-case response time. This analysis is closer to queuing theory, than classical response-time estimation in the real-time community. In queuing theory, jobs arrive at a queue according to some probability distribution. The service time of an element in the queue is also governed by a probability distribution. The total time from arrival to finish includes waiting time in the queue and processing time in the server. Fixed priority scheduling is one example of a scheduling strategy to analyse. There exists theory for priority driven queues including preemption, but the stochastic arrival and service processes do not match the model of a scheduling system. In basic queuing theory, the poisson process is used, but for fixed priority scheduling, the arrival is periodic, and the service rate is hardly poisson either. The evaluation of an expected-case response time, and possibly any second

moment figure for its accuracy, can be helpful in the design as an approximate indication of a likely timing behaviour.

6.4 On-line schedulability test and adaptive task parameters

A dynamic scheduling policy allows new tasks to be inserted into the task set currently running. In order to guarantee hard real-time, i.e. that every deadline is met, a schedulability test must be undertaken. By changing task parameters such as periods and offsets, a previously unschedulable task set can be made schedulable. This is typically done manually off-line, but in order to achieve flexibility a further automation of the scheduling work could prove useful. The scheduling strategy called earliest deadline first (EDF) has a really easy schedulability test. The response time analysis for fixed priority scheduling (FPS) is also straight forward, at least for a uniprocessor. One problem to study is how to select task parameters rationally in order to satisfy timing constraints (deadlines) during worst-case conditions and avoid transients due to the actual switching of parameters. The change of parameters will also affect the average timing behaviour. An on-line schedulability test made before activation (“pre-runtime”) for a new set of tasks, while the old task set is still running, would increase the chances of admitting a new task.

6.5 Safety-critical and testing issues for QoC

Control applications are typically safety-critical. They need to be verified and tested with respect to the temporal behaviour, to assure dependability. Admission control plays a crucial role here. It is often stated that flexibility and dependability are difficult to combine. For example, a statically scheduled system without random jitter is beneficial in order to avoid state explosion during testing, but it is also beneficial at runtime, e.g. to achieve a high error detection coverage. But the static schedule needs to be substituted by a more dynamic scheduling algorithm in order to increase the flexibility. An important step is to assure that the logically verified and tested functions, keep their logical behaviour and output contained within specified boundaries. Another research idea is to make the scalability and negotiation distributed, since this could help increase the dependability.

6.6 Work bench for scalability and QoC

The proposed QoC architecture in paper C needs to be developed and implemented in at least one demonstrator, since this will increase the credibility of the approach. It is likely that new problems and important details will be revealed during this process, which will help impel and refine the architecture. One of the most important parts here, is to achieve scalability. Scalability is the primary feature to get working, since it is important for all classes of applications, not only for controllers. Different negotiation and admission control concepts should be tried out. These are typically specific for the dedicated systems, and it could be advantageous to investigate existing and running prototypes and products. The aim for scalability must be accompanied by a development method, which among other things will give a well-defined interface between modules. A codesign method has already been outlined. A task is to demonstrate that the effort of developing a flexible and modular platform is motivated.

7 References

- Krishna C. M. and Shin K. G., “Real-time systems”, ISBN 0-07-114243-6, 1997.
- Nilsson J., “Real-time control systems with delays”, Doctoral Thesis, Automatic Control, LTH, 1998.
- Redell O., “Response time analysis for implementation of distributed control systems“, Doctoral Thesis, Mechatronics Lab, KTH, ISRN KTH/MMK/R--03/17--SE, 2003.
- Redell O., El-khoury, J. and Törngren M., “The AIDA tool-set for design and implementation analysis of distributed real-time control systems”, J. of Microprocessors and Microsystems, Elsevier, Vol 28/4 pp 163-182, 2004.
- Redell O. and Sanfridson M., “Exact best-case response time analysis of fixed priority scheduled tasks“, Proc. of the 14th Euromicro Workshop on Real-Time Systems, Vienna, 2002.
- Rehbinder H., “State estimation and limited communication control for nonlinear robotic systems“, Doctoral Thesis, TRITA-MAT-01-OS-09, KTH, 2001.
- Rehbinder H. and Sanfridson M., “Integration of off-line scheduling and optimal control“, Proceedings of the 12th European Conference on Real-Time Systems, Euromicro, pp. 137-143, Stockholm, 2000.
- Rehbinder H. and Sanfridson M., “Scheduling a limited communication channel for optimal control“, Proceedings of the 39th IEEE Conference of Decision and Control, volume 1, pp. 1001-1016, Sydney, December 2000.
- Sanfridson M., “Timing problems in distributed control”, Licentiate thesis, Mechatronics Lab, KTH, ISRN KTH/MMK--00/14--SE, May 2000.
- Sanfridson M., “Problem formulation for QoS management in automatic control”, Technical Report TRITA-MMK 2000:3, Mechatronics Lab KTH, March 2000.
- Sanfridson M., “Discretization of process and loss function for a control loop having time-varying period and control delay”, Technical Report ISRN KTH/MMK--03/02--SE, Mechatronics Lab, KTH, 2003.
- Sanfridson M., “An overview of the use of LMI in control to assess robustness and performance”, Technical Report ISRN KTH/MMK--03/08--SE, Mechatronics Lab, KTH, 2003.
- Scherer C and Weiland S., “Linear matrix inequalities in control”, version 3.0, October 2000.
- Sjunghamn P., “QoS for embedded distributed control systems“, Master thesis, Mitthögskolan, 2003.
- Skogestad and Postlethwaite, “Multivariable feedback control - Analysis and design”, ISBN 0-471-94277-4, Wiley, 1996.
- Törngren M., “Modelling and design of distributed real-time control applications“, Doctoral thesis, Machine Design KTH, ISRN KTH/MMK--95/7--SE, 1995.
- Törngren M., Andersson M., Wittenmark B., Torin J. and Wikander J., “Integrated real-time computer control system architectures — DICOSMOS2 — final report”, Projektnr P11762-1/P11762-2 and Dossie-Diariennr. 1K1P-98-06321/1K1P-99-06187, 2001.
- Åström K. J., Wittenmark B., “Computer controlled systems - Theory and design“, 3rd edition, ISBN 0-13-314899-8, Prentice-Hall, 1997.
- Österman J., “Scalability and QoS for embedded distributed control systems“, Master thesis, Mechatronics Lab, KTH, MMK 2001:81 MDA181, 2001.