Hybrid Systems: Modelling Embedded Controllers

This paper indicates several possibilities to model embedded controllers as hybrid systems.

It is shown that hybrid systems, the combinations of time-continuous and discrete-event systems, allow many powerful mathematical descriptions.

Instances of hybrid systems include embedded controllers, supervisory systems and asynchronous processes.

Although simulation programs are currently becoming more powerful for dealing with hybrid systems, analysis and synthesis are still open questions.

Paul P.J. van den Bosch[1]
Maurice Heemels[2]

Eindhoven University of Technology
Faculty of Electrical Engineering
Section Measurement and Control
P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands

[1]Tel : +31(0)40.247 23 00 - Fax : ...243 45 82
e-mail : p.p.j.v.d.bosch@tue.nl

[2]Tel : +31(0)40.247 35 87 - Fax : ... 243 45 82
e-mail : w.p.m.h.heemels@tue.nl

1 Introduction

The class of time-discrete asynchronous systems or discrete-event systems (DES) yields challenging problems for control engineers.
DES are all around us and are part of embedded controllers that emerge in all kinds of consumer electronics ranging from micro-wave units, audio and video devices, PC, cars, tools to toys.
They take care of exception handling, safety, alarm detection, starting and stopping procedures, mode switchings, etc. The code that realises an embedded controller can be divided into two parts. One small part contains the local digital controllers (difference equations) that realise, within one mode of operation, the required dynamic behaviour of the controlled process or the regulation of a small subprocess.
Another, much larger part is utilised for starting, stopping, exception handling, event detection and mode transitions (DES). Also this last part influences the overall dynamic behaviour.
Both theory and software support analysis and synthesis of the local digital controllers. However, for designing the larger part of the controller code (DES) in combination with the physical process and the digital controllers, almost no support is present. Even the combined modelling and simulation of these two parts can yield considerable problems.

As models are the ultimate tools for obtaining and dealing with knowledge, not only in engineering but also in, for example, philosophy, sociology and economics, a search has been undertaken for appropriate mathematical models for the combination of embedded controllers and time-continuous plants.
These hybrid models are capable of describing the dynamic relations between the hardware (physical plant), the digital controllers and the software processes. Hybrid formulations include, for instance, generalised semi-Markov processes, queuing theory, timed and hybrid Petri nets, differential and hybrid automata, switched bond graphs, mixed logical dynamic models, duration calculus, real-time temporal logic and simulation languages.
These hybrid dynamic models differ from discrete-event models, because time is taken explicitly into account. When time is left out of the formulation we have untimed models such as (ordinary) Petri nets and finite state automata. These are used for problems of state reachability, safety and deadlock avoidance in, for example, computer science.
Obviously, each hybrid model description has its own capabilities and limitations. The choice of a suitable framework is a trade-off between two conflicting criteria: the modelling power and the decisive power.
The modelling power indicates the size of the class of systems that fit in the specific model description, while the decisive power is the ability to prove qualitative and quantitative properties of systems in this class. It is obvious, that a too broad class will result in only restricted knowledge of individual elements in the class. As a consequence, it is often not possible to transform one formulation into another one, without adding or loosing information.

A hybrid system description applies in the following situations:

- Systems containing both time-continuous and discrete-event models. Examples are embedded controllers, hierarchical control systems or supervisory systems implemented in a physical environment, command and control systems, air traffic management [46], co-ordinated submarine systems and automated highway systems [31].
  In describing e.g. air-traffic control systems, a combined description contains continuous-time models of the dynamics of the air planes, while a discrete-event model describes the human and/or organisational processes that realise the communication and assignments between the planes and the traffic control centre.
  Such a description is more accurate than an exclusively continuous-time model that neglects or simplifies the human organisation or a discrete-event model that simplifies the dynamic behaviour of the air planes too much [46].

In command and control situations, such as guided vehicles, both the dynamics of the controlled objects (ships, aircraft or cars) and the sensing, decision and action tasks interfere [31]. Also in manufacturing there is a mixture of physical phenomena and information flow, each requiring its own model description.

- Systems with different dynamical regimes. One might think of switching controllers like relays, thermostats, gain scheduled or sliding mode controllers and electrical circuits containing diodes, thyristors or other switches. In certain systems mode transitions are not artificially introduced, but are physically present.
  Examples include the slip and stick phases in friction phenomena, saturation or deadzones in engineering systems, constrained and unconstrained modes in rigid body dynamics (e.g. a robot arm striking the surface of an object) and so on.
- Physical systems containing asynchronous processes. As an example, consider an encoder measuring the angular position of a motor. The measurements depend on the changing velocity of the motor and are consequently asynchronous in time.
  Similar sensors are used for magnetic/optical disc drives and transportation systems where the longitudinal position is only known when a marker has passed. Another example is the actual heart rate of patients. The rate information becomes only available after a complete cycle has been realised, each cycle with a different cycle time.

Whether a continuous or a discrete-event modelling paradigm should be chosen depends on the level of detail at which the world is studied. To illustrate this, consider a colony of bacteria.
If we are interested in the total number of bacteria only, we can try to describe the dynamics of their average number with differential equations, taking the continuous approach. However, if details of the life of a separate bacterium are significant, the behaviour of the colony can be modelled as a set of discrete-event processes interacting with each other. The filling of a tank at a petrol station allows also many viewpoints.
If the duration of this process can be predicted at the start (for example taking an average duration for this pump), the discrete-event modelling approach can be used. Of course, the filling process depends on the dynamics of the pump used, which, if not solved explicitly for the filling duration, requires a continuous modelling approach.

Nowadays, there are mainly two approaches to the design of hybrid systems. The first method performs the synthesis by techniques either exclusively tailored for the continuous-time or discrete-event domain. As a result, the used models neglect either the continuous or discrete characteristics of the system.
The other methodology separates the design of the continuous and discrete parts and merges the embedded controller and the physical plant in the final stage in an ad hoc and heuristic manner. One can imagine that both approaches are not very efficient and require tuning, prototyping and trouble-shooting, which is extremely expensive and time consuming.
Fortunately, it is widely recognised by control engineers [4,35], computer scientists [39], mathematicians and simulation experts that combined synthesis methods are needed by the industry. This interest resulted in a series of workshops on hybrid systems, see e.g. [2,5].

One tries to compensate the lack of analysis and synthesis tools by efficient and accurate simulation packages.
Also the MathWorks, the producer of Matlab, has recognised the importance of hybrid systems and created a new simulation environment called StateFlow to cope with hybrid systems. Many other numerical environments like Chi, Omola/Omsim, Prosim, Psi and Shift are under development to allow simulation of embedded systems.

In this paper, we will discuss the lack of appropriate mathematical models for common-day applications of embedded controllers.

Next, the differences are elucidated between the world of physics on one side and the world of software and computer science on the other. Several mathematical models are studied and it is discussed how they cope with the requirements of describing embedded controllers.
At this point, we would like to state that this work is by no means encyclopaedic and that we do not intend to describe the techniques in detail due to space limitations. However, references are provided that may serve as entries to the broad literature on hybrid systems.

2 Differences between time-continuous physics and discrete-event processes

In this section a distinction is made between time-continuous physics (described by differential equations), time-discrete, synchronous digital controllers (described by difference equations) and asynchronous timing processes in software (described as DES).
As we can easily deal with the combination of synchronous digital controllers and continuous processes (the z-transformation allows detailed analysis and synthesis), we focus on the differences between time-continuous (CT) and discrete-event systems (DES).
For ease of reference, we include also descriptions used in computer science like finite state automata or temporal logic [1,38] in the terminology DES.

In CT models variables represent physical quantities such as energy, mass, velocity, temperature, which vary continuously in time and have a continuous value. Discrete-event models deal with processes that represent information. A process (and corresponding variables) can be created or deleted, while in CT a certain variable always exists.
Most important, the variables in (untimed) DES take discrete (integer) values representing e.g. the number of an operating mode.
A logical switch can be represented by a Boolean variable δ having two discrete values $\delta \epsilon \{0, 1\}$. A striking effect of discrete values of variables is that model reduction is almost impossible (there are some partial results using bisimulations, see e.g. [26,29]).
At first hand it seems reasonable that solving problems with real variables (which can have any real value) is more difficult than dealing with variables taking only limited (sometimes two) discrete values. This assumption is not true in general. It turns out to be much simpler to deal with a problem with real variables than with discrete variables.
The main reason is that CT-models have the notion of local continuity. Linear low order models can therefore approximate the behaviour of complex non-linear systems close to an operation point. With discrete variables there is no local environment. With 10 Boolean variables, for example representing the status of switches, each operating point has 2^{10}-1 neighbours. The behaviour of each neighbour can be completely different.
Searching for the best solution often ends up with evaluating all possible operating points, which leads to a tremendous increase in calculation time (exponential growth) when the complexity increases. For CT models a number of possible model reduction techniques are available.

Time-continuous models (CT)

- Continuous models are used when studying the dynamics of, for example, mechanical, biological and control systems. All model variables change continuously in time and in parallel.
  The constitutive laws are based on e.g. the conservation of energy and mass and the physics of (ideal) components. Identification techniques can be used to obtain black-box models.
  The assumption of local continuity is crucial for identification, since the restricted number of measurements is inter- and extrapolated to nearby trajectories.
- Model reduction is in principle always possible. The reduced model will exhibit almost the same behaviour in the area of interest. Even for complex systems the model order will be limited.

- ODE (ordinary differential equations) or DAE (differential and algebraic equations) describe the relations between (continuous) variables.
- The dimension of a model, the number of its variables, remains constant during the simulation run. The variables always exist. It is impossible to create or delete variables during the evolution.
- A declarative approach, so defining the variables and their mutual relations, is the most natural way to define continuous models.

Discrete-event models (DES)

- Discrete-event models are mainly used for studying the behaviour of man-made systems like human organisations, queuing problems, computer systems, and manufacturing processes.
- Changes occur at isolated time instants (event times) and not all model entities need to be involved in them. These instants are not necessarily equidistant in time. So, the model exhibits asynchronous behaviour. In between two event times, no actions occur.
- All elements of the model operate in parallel, but each of them executes its actions sequentially.
- As a consequence of the discrete character of the variables, model reduction is almost impossible. Any small part of a model can have a large influence. The lack of local continuity obscures the use of identification techniques.
- The model elements take decisions and execute discrete actions. The scheduling and sequencing of events represent instant relations between causes and effects.
- The size of a model, the number of entities, can vary during a simulation run. So, variables and/or equations can be dynamically created, suspended and/or deleted.
- A language in which a discrete-event model is defined should support a flexible definition of actions and decisions. It has to be a strong procedural language.

In table 1 an overview of the differences is given.

	Time continuous (CT)	Discrete event (DES)
Process	Physical	man-made
Variables	Physical quantities	Information
	Conservation of energy, mass, etc.	operating modes
	Always present	create, activate, suspend, delete
	time continuous	time discrete (asynchronous)
	Continuous in value	discrete in value
State	Physical quantity	time to next event
Models	Parallel, acausal	sequential, causal
	Differential and algebraic equations state space models transfer functions	max algebra Petri nets finite state automata
Model reduction	Possible	almost impossible
Problems	Controllability, observability, stability	Reachability, safety, liveness
Delay	e^{-sT}, storage for T seconds	DELAY(p, T), no storage

Table 1: Differences between time-continuous and discrete-event models

Also graphical representations for DES and CT differ. To give a quick round up of examples, consider the following list (see e.g. [11] for the first three representations).

- *Physical representations*, such as electronic circuits, mechanical drawings, thermal schemes. They represent an acausal parallel description of all variables, which are physical quantities.
  A splitting of a line in an electronic circuit states that $i_1 = i_2 + i_3$.
- *Bond graphs* are composed of components that exchange energy or power through connections (called "bonds"). Both causal and acausal relations are possible. The variables are related to effort and flow thereby keeping a strong relation to physics.
  Bond graphs make use of the similarities present in mechanics, electrical circuits and hydraulics.
- *Block diagrams* represent a causal, parallel description of variables that represent signals. All blocks (think of integrators, gains and additions) have a clear input-output relation: a block cannot influence its inputs. Now a splitting of a line states that $i_1 = i_2 = i_3$.
- *Petri nets* or *finite state automata* depict logical relations between variables and processes. The representation consists of a directed graph between a finite set of automaton states or places and transitions. The arcs contain information (e.g. a required input, weighting or guard) indicating when a transition is enabled and whether a reset must be performed. There is no or a weak relation to physics.
  There are causal relations among the variables.

3 Mathematical models for hybrid systems

Both CT and DES models have proven their value. A merging between both models allowing strong design and analysis methodologies seems attractive for industry, because it will decrease the investments and time-to-market for the development of new products.
The combined model will introduce many difficulties and new problems, also interesting from an academic point of view. Hybrid systems will inherit every bad property of a separate domain, while a good property might be inherited only when both domains own it. No general-accepted language is yet available although several interesting attempts are currently being made.
One might start from either CT or DES models and then to try to include some or all elements of the other domain. In this section several approaches will be discussed and compared. We would like to stress once more, that this list is not exhaustive.
Other interesting methods that are not commented upon in this paper include differential automata [45], duration calculus [16], real-time temporal logic [1,38], time communicating sequential processes [19,27] and switched bond graphs [42].

3.1 Approaches starting from DES/Computer Science

Many different models exist for describing DES, such as the max algebra, finite state automata, Petri nets, queuing networks and Markov chains. Some of these descriptions are available as untimed models for verification of safety and liveness (no deadlock situations) and reachability analysis. Timed versions of these descriptions are used for quantitative analysis, performance studies and optimisation.

3.1.1 Petri nets

An (ordinary) Petri net [18] is a graphical representation consisting of a finite set of places $P=\{p_1,p_2,...,p^n\}$, a finite set of transitions $T=\{t_1,t_2,...,t_n\}$, a set of arcs going from places to transitions *Pre:* $Px\,T\,\{0,1,2,3,...\}$, and a collection of arcs from transitions to places *Post:* $Tx\,P\,\{0,1,2,3,...\}$. Circles and bars graphically represent places and transitions, respectively, that are interconnected by arcs. An integer value, called the marking, is connected to every place, that represent the number of tokens in that place (e.g. the number of busses (= tokens) at a station (= place), or the number of products at a machine or in a queue).
Tokens can move from one place to another via a transition. A transition t is enabled, when all places p contain at least $Pre(p,t)$ tokens. If the transition t is fired (executed), the marking of place p is increased by $Post(t,p)-Pre(p,t)$. Note that $Pre(p,t)=0$ indicates that there is no arc from p to t and similarly, $Post(t,p)=0$ indicates that there is no arc from t to p.
As an illustration, consider a factory consisting of several stacks or queues (= places) storing all kinds of semi-finished products and raw materials (= tokens).

These materials are processed by machines (= transitions) to fabricate new (semi-)finished products. The weightings of the arcs given by *Pre* indicate how much raw materials or semi-finished products are needed by a machine to start its production, and *Post* indicates how many finished products are generated and sent to specific stacks.

Petri nets form a graphical representation and a mathematical tool for DES. It allows concurrency, synchronisation and resource sharing and is widely used for modelling communication protocols, transportation networks and manufacturing systems.
There has been a large effort to include continuous behaviour in the Petri net, yielding the continuous Petri net (places with continuous variables), the hybrid Petri net (both discrete and continuous places), and timed Petri nets (time associated to the arrival of tokens in a place or a transition). With the hybrid Petri net both discrete-event processes can be modelled and some, simple continuous time models, such as integrators to represent buffers (with constant time derivative).
Simple P-controllers, together with all their safety measures, mode switching, exception handling, supervision, etc. can now be modelled as a hybrid Petri net. In [20] the differential Petri net has been introduced. Besides the discrete elements also complex time-continuous dynamics can be incorporated. Using numerical time-stepping methods the solution of continuous differential equations can be approximated and included as time-discrete processes in the model.
The timed, differential and hybrid Petri nets result in DES models that can be analysed (deadlock detection, safety verification, reachability) and simulated using DES software. By their broad applicability and extendibility Petri nets are popular. They yield a common language between different application fields.
However, Petri nets are not structured modelling tools. Moreover, the added flexibility is paid by a considerable increase in complexity and a reduction of the ability to do thorough analysis.

3.1.2 Hybrid Automata

A universal approach (including e.g. Brockett's model described below) has been described in [13,32]. Hybrid automata are extended from finite state machines by replacing simple clock dynamics inside each discrete state by more involved differential and algebraic equations. In [13] also the term "controlled general hybrid dynamic system" has been used.
A controlled hybrid automaton H_c is given by (particular notation taken from [13])

$$H_c = (Q, \Sigma, A, G, V, C, F)$$

with Q the set of discrete states, $\Sigma=\{\Sigma q\}_{q\in Q}$ the collection of controlled dynamical systems, A the set of autonomous jump sets, G the autonomous jump transition map, V the transition control set, C the controlled jump sets and F the controlled jump destination maps.
The model switches between a collection of dynamical systems (parameterised by a mode $q\epsilon Q$). Inside mode q the system is governed by the dynamics Σ_q given by $\dot{x} = f_q(x,u)$, where the control input u can be chosen from an a priori specified set. When the state $x(t)$ reaches A_q at time t, a mode transition and state jump must occur according to $(x(t+),q+) := G_q(x(t-),v)$, where $v\epsilon V_q$ is an available discrete control action. In case $x(t)$ reaches C_q, then we may choose to jump and if so, we may choose a destination (a new state and mode) $(x(t+),q+)$ from the set $F_q(x(t-))$.
In both situations the process continuous. Note that it is possible that multiple mode transitions and jumps happen at one time instant. This model description encloses many of the other time-continuous, discrete-event and hybrid models. In spite of its intriguing simple appearance it can model quite complex phenomena. Branicky et. al. [13] derived conditions for the existence of optimal controls given a cost function and stated sufficient conditions for optimality in terms of generalised quasi-variational inequalities (extensions of the classical Hamilton-Jacobi-Bellman equation).
However, rather complex mathematics is involved and solving the inequalities has a high computational complexity. Although the concept of hybrid automata stems originally from computer science, the techniques used in [13] are clearly from control and systems theory.

3.1.3 Transformation from CT to DES

In [37] one proposes to replace a CT model by a DES model. Each variable is divided into a number of regions, for example three: below minimum, between minimum and maximum and above maximum. Only the presence of the variable inside a region is detected. Leaving a region (crossing a region border) realises an event, which assigns the «discrete state» or mode of the model to another discrete value.
So, only the time of a border crossing and the discrete value of the present region are used to describe the dynamic behaviour.
Advantages of this approach are that a DES model of CT-plant can be extended with all DES-like functionality of an embedded controller, and that one can rely on standard DES tools. Of course, approximating a CT model by a DES model introduces loss of information and a considerable increase of complexity. A tenth-order CT model with 5 regions for each of its state variables has already 105 different regions or (operating) modes.

3.2 Approaches starting from CT

3.2.1 Variable Structure Systems (VSS)

Sliding mode control [47] is an approach that allows the inclusion of switches in a time-continuous model description. The control law is formulated as to reach a switching line (plane) in the state space of the continuous system and, once reached this plane, to keep the state at this plane. In theory, infinitely fast switching is required.
Once this sliding mode has been achieved, the system exhibits attractive performance such as complete reduction of disturbances and insensitivity to parameter variations.
Moreover, the order of the system is reduced. This objective is achieved by using different control laws in the individual regions separated by the switching plane. The theoretical requirement of infinite fast switching has to be relaxed in practice such that some of the features are less advantageous. Theoretical understanding of VSS allows analysis and synthesis of (sliding) mode switchings within a CT description.
Switching control schemes using e.g. gain scheduling, relays and piecewise linear systems are quite popular considering the many studies encountered in the literature, see e.g. [12,28,30,43] and the references therein.

3.2.2 Complementarity systems

Complementarity systems consist of a set of differential and algebraic equations

$$0 = F(\dot{z}(t), z(t))$$

$$y(t) = g(z(t)) \epsilon R^k$$

$$u(t) = h(z(t)) \epsilon R^k$$

with t the time variable, $z(t)$ a state variable and $u(t)$ and $y(t)$ variables satisfying the complementarity conditions

$$u_i(t) \geq 0,\ y_i(t) \geq 0, \{u_i(t) = 0 \vee y(t) = 0\} \text{ for all } i=1,2,...k$$

The complementarity conditions are similar as in the linear complementarity problem [17] of mathematical programming. The relation "$\vee$" is meant to be a nonexclusive or. The complementarity conditions comply with the characteristics of an ideal diode.

Either the current through or the voltage across the diode is equal to zero, and the current through and voltage across the diode are both nonnegative. Hence, certain types of switches can easily be incorporated.
However, the complementarity formalism is more general than that. Further examples include (see [23]) constrained mechanical systems (with collisions), piecewise linear systems (including saturation, deadzones, relays, Coulomb friction, etc.), systems of equations resulting from optimal control problems with state/control constraints, hydraulic systems with one-way valves, projected dynamical systems and so on.
Based on a solid system theoretic knowledge, analysis concerning well posedness, simulation methods and characterisation of the behaviour are possible [24,40]. Future research will be concerned with controllability and stability analysis, and controller design.

3.2.3 Brockett's model

Many researchers have attempted to obtain a «unified» model (like the hybrid automata) for describing a large class of hybrid systems. A disadvantage is that the richer the model, the more complicated the analysis and synthesis becomes.
Brockett [14] introduces a model with a continuous variable $x(t)$ and a discrete variable z, where the events are triggered by a time-continuous non-decreasing process $p(t)$. For the variable $p(t)$ we denote the greatest integer less than $p(t)$ by $\lfloor p(t) \rfloor$ and $\lceil p(t) \rceil := \lfloor p(t) \rfloor + 1$. When $p(t)$ crosses and integer value, an event occurs such that the discrete event part of the model can be recalculated. Brockett's model looks like:

$$\dot{x}(t) = f_1(x(t), u(t), z\lfloor p \rfloor)$$

$$\dot{p}(t) = f_2(x(t), u(t), z\lfloor p \rfloor)$$

$$z\lceil p \rceil = f_3(x(t), u(t), z\lfloor p \rfloor)$$

with $u(t)$ an external input. Brockett tries to capture many practical models into one common framework that might be used as a starting point for solving many problems by one general methodology.
Analysis or synthesis problems are not treated in [14].

3.2.4 Mixed Logical Dynamic models (MLD)

In [7] the Mixed Logical Dynamic (MLD) model is presented to cope with both time-continuous and logic variables. The first are needed to describe regulation and tracking of e.g. chemical plants or electromechanical devices.
The latter are required for describing the supervision, planning, mode switching and conflict resolution (DES) among separate processes. The MLD model can be used for

- linear systems
- constrained linear systems
- sequential logic systems such as finite state machines, automata
- some discrete-event systems
- non-linear systems modelled by piecewise linear systems

Variables can be either real (R^n) or logic such as the statement "x≥0" that is either true or false $\{T, F\}$. This can be represented by a Boolean variable $\delta \in \{0, 1\}$. Logic statements are converted into sets of linear inequality constraints. As an example, consider the following system.

$$x(t+1) = a\,x(t) + u(t) \quad \text{if} \quad x(t) \geq 0$$
$$x(t+1) = b\,x(t) + u(t) \quad \text{if} \quad x(t) < 0$$

with $m \leq x(t) \leq M$. To rewrite these equations in a MLD model, $x(t)<0$ is replaced by $x(t) \leq -\varepsilon$, where ε is a small positive tolerance (typically the machine precision) at which the violation of constraints can be detected.

By introducing the auxiliary variable $y(t) = \delta(t)x(t)$, it can be verified that the following equivalent model arises [7]:

$$x(t+1) = (a - b)\,y(t) + b\,x(t) + u(t)$$
$$y(t) \leq M\delta(t)$$
$$y(t) \geq m\delta(t)$$
$$y(t) \leq x(t) - m(1-\delta(t))$$
$$y(t) \geq x(t) - M(1-\delta(t))$$
$$-m\delta(t) \leq x(t) - m$$
$$-(M+\varepsilon)\delta \leq -x - \varepsilon$$

This model and many others can be cast in the general form of a MLD system given by

$$x(t+1) = A_t x(t) + B_{1t} u(t) + B_{2t}\delta(t) + B_{3t} z(t)$$
$$y(t) = C_t x(t) + D_{1t} u(t) + D_{2t}\delta(t) + D_{3t} z(t)$$
$$E_{2t}\delta(t) + E_{3t} z(t) \leq E_{1t} u(t) + E_{4t} x(t) + E_{5t}$$

The variables x, y and u contain a real-valued part as well as a logical part, δ is a logical and z is a real variable. Some mathematical tools are available to test whether such a system is well posed (existence of a unique solution for $\{x(t+1), y(t)\}$ given $\{x(t),u(t)\}$) or completely well posed (existence of a unique solution for $\{x(t+1),y(t),z(t), \delta(t)\}$ given $\{x(t),u(t)\}$). Although the MLD model has broad applicability, it is not easy to analyse stability or design a fixed controller.
This model is more suitable for model predictive control (MPC) that does not design a controller with fixed settings and structure for an infinite time. Instead, each time step the values of the controller $u(t)$ are calculated, depending on a criterion for a finite time horizon $(t=1, .. , H)$, the model and many constraints like $u_{min} \leq u(t) \leq u_{max}$. Only the first calculated value $u(1)$ is implemented, and the optimisation problem is solved again for a receded horizon.
Because the MLD model is formulated as a time-discrete model with control input $u(t)$, $t=1,..., H$ with a real and logic part, and logic variables $\delta(t)$, $t=1,..., H$ $(\delta(t) \in \{0, 1\})$, mixed integer quadratic programming problems (MIQP) arise. With the exception of special cases, mixed integer problems are NP-complete, indicating an exponential growth of solution time with the number of variables.
However, several algorithms have been proposed for solving such combinatorial optimisation problems [22], which work reasonably well in practice (although needing non-polynomial time in worst case).

3.2.5 CT processes and asynchronous measurements

As mentioned before, the choice of a suitable framework is a trade-off between modelling and decisive power. In the previous subsections quite general model classes are described encountering several problems like computational intractability or little decisive power. In this subsection we describe a special situation allowing full synthesis and analysis of controllers in spite of the hybrid nature of the sensor.
The particular problem was raised by the company Buhrs-Zaandam B.V. (The Netherlands), which builds mailing machines. Mailing machines create post parcels consisting of a journal with many different inserts. These parcels are created with a speed of about 20.000 parcels per hour. Essential is that all parts (conveyor belt, sheet-feeders, packaging and labelling module) of this machine operate synchronously.
To reduce the costs, low-resolution encoders for the motors, that drive the individual tools, have been used. However, this low resolution results in a limited amount of interrupts and low accuracy. The measurement updates are asynchronous in time, since they depend on the speed of the motor. Present day theory does not support the analysis and design of systems with asynchronous measurements and control updates. However, by introducing a transformation from the time domain into the position domain, the problem changes into a non-linear model description with synchronous measurements and control updates [25].

The applications of standard control theory can now be used to design an appropriate, position-synchronous but time-asynchronous controller. Simulation results and application of these controllers in practise yield impressive results.
An asynchronous controller outperforms the synchronous controller with a fixed and much higher sample frequency. Other techniques based on asynchronous measurements using time-varying Luenberger observers can be found in [36].

Many other attempts have been made to formulate a model that can deal with time-continuous models and parts of DES. It turns out that there is, up to now, no easy solution that combines the conflicting requirements of modelling and decisive power.
Computational intractability or decidability problems occur for even the most elementary hybrid systems as demonstrated by the work of Blondel and Tsitsiklis [8]. So, in many situations simulation may help in finding a solution.

4 Simulation

Simulation is the ultimate escape when mathematics lacks an analytical solution. Almost anything that can be described can be included, such as different signals, systems, components, disturbances and non-linearities.
Flexibility is the big advantage of simulation. However, simulation has to be considered as just executing experiments. The answers obtained are only valid for the experiments carried out. Reliable extrapolation of these results to other operating conditions is not guaranteed. In general, a simulation can only prove that a design does not meet the specifications. It cannot justify a proper design in all cases.
The reason is that designing experiments to cover all possible operating modes can be a tedious and time-consuming activity and is almost always impossible.
For hybrid systems convenient simulation environments exist that allow the description of both time-continuous and discrete-event models inside one language. Early attempts have been made either from the discrete-event world (e.g. Siman, Prolog, Prosim [41]) or the time-continuous world (e.g. Psi [10], Omola/Omsim [3], Matlab/Simulink/StateFlow, Modellica [33]). Nowadays integrated simulation languages are available such as the (language [6]. All the mentioned packages can be considered as "event-driven methods" (term taken from [34]). Event-driven methods are based on the procedure used to describe the evolution of hybrid automata in section 3.1.2.
The evolution inside a mode is approximated by standard integration routines for DAE. Indicators are monitored to determine the event times (i.e. the times entering the jump sets), often given by zero-crossings of certain (combinations of) variables. At the event time a state jump (re-initialisation) and a mode transition (mode selection) has to be computed.
After possibly multiple events, the simulation of the DAE corresponding to the new mode can be continued. This cycle of DAE-simulation, event detection, mode selection and re-initialisations is repeated until the end time is reached. Besides the event-driven method, Moreau [34] discusses also the following simulation techniques for rigid body dynamics.

- Smoothing methods: replacing hard impacts and bounds with soft impacts and bounds by introducing artificial damping and stiff repulsion laws)
- Time-stepping methods (Moreau calls them "contact dynamics"): directly replacing the describing equations by discretised equivalents by approximating derivatives by numerical integration routines and enforcing all algebraic relations to hold on each time-step.

As mentioned before, these methods are tailored for constrained mechanical systems. Extending them is only possible for subclasses of hybrid systems with a special structure. Smoothing methods require insight in the system's behaviour to come up with suitable regularisations.

Applying time-stepping methods (as indicated by Moreau) asks for a simultaneous treatment of the discrete-event and continuous-time parts, which is, of course, in general not possible.
However, for certain subclasses of complementarity systems [9,15,44] time-stepping methods are proven to be valid. For general hybrid systems (like hybrid automata) the event-driven approach is most appropriate as it can deal with the asynchronous occurrence of events, and separates the continuous phases from the re-initialisation and switching rules.
An event-driven simulator for hybrid systems should support (at least) the following three main facilities:

- *State events:* if a time-continuous variable crosses some level, an interrupt has to be generated such that a corresponding action (re-initialisation, mode transition) can be realised. This facility is needed to model complex physical phenomena like bumps or collisions.
- *Time events:* time is the only variable that is completely independent and autonomous in a simulator. Time events are triggered at fixed time instants or at specific instants where a certain time interval has elapsed after another event.
  The difference between state and time events lies in the cause. In both cases, actions like mode switching (change of structure of the time-continuous part) and re-initialisations must be possible in the simulation language.
- *Data exchange* between the time-continuous and the discrete part. Care has to be taken that only constants, parameters and states are changed. A modification of algebraic dependent variables will have no effect. A re-initialisation of the control value u in a state feedback law $u=Fx$ is useless without changing x accordingly, since the state x determines the value of u immediately after the event.

The issue raised in the second item prevents that a DES and a CT simulator can be coupled directly. The fact that both simulators have their own independent time variable creates many problems. An attempt to merge simulators requires that one independent variable is dominant over the other. A (hybrid) simulator can have only one real time variable.

5 Conclusions

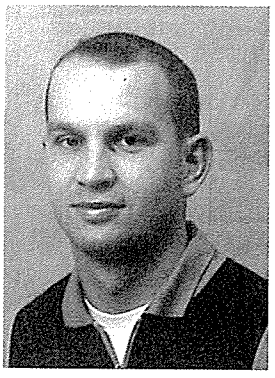
In studying embedded systems the dynamic behaviour of three interacting parts have to be analysed, to wit, the physical process, the digital controller and the software that takes care of start-up, shut-down, exception handling and mode switching.
These three parts are formulated as models of, respectively, time-continuous, time-discrete and discrete-event systems. Each model utilises its own mathematical description: differential equations, difference equations and, e.g. Petri nets or finite state automata.
To understand the dynamic behaviour of the combination of the embedded controller and the physical plant the interaction of these three models has to be analysed.
Several formulations of hybrid systems seem likely candidates to describe the combined system. Among these hybrid models, approaches starting to extend the time-continuous models with discrete-event facilities seem the most promising ones.
They maintain their roots in system theory which allows attractive analysis and synthesis methods to be utilised. Depending on the application an appropriate model has to be selected. When the application at hand fits in the class of variable structure systems (VSS), complementarity systems, CT-processes with asynchronous measurements or mixed logic dynamic (MLD) systems, associated methods are available.
Else, one of the «unified» methods (e.g. hybrid automata or Petri nets) has to be used resulting in lack of decisive power or intractable algorithms. In many cases one has to rely, once again, on simulation experiments. As a final remark, hybrid systems theory is widely unexplored and of clear industrial relevance, which will make it a major research area in future years.

References

1. R. Alur, and T.A. Henzinger, *Real-time logics: complexity and expressiveness.* Information and Computation, 104, 1993, p. 35-77.
2. R. Alur, T.A. Henzinger and E.D. Sontag (eds.), *Hybrid Systems III.* Proc. of the Workshop on Verification and Control of Hybrid Systems, New Brunswick, New Jersey, October 1995. Lecture Notes in Computer Science 1066, Springer, Berlin, 1996.
3. M. Andersson, S.E. Mattsson and D. Brück and T. Schöntal, *Omsim - An integrated environment for object-oriented modelling and simulation.* In Proc. of the IEEE/IFAC joint symposium on Computer-Aided Control System Design, Tucson, Arizona, 1994, p. 285-290.
4. P.J. Antsaklis and A. Nerode (guest eds.), *Special issue on hybrid control systems.* IEEE TAC, 43(4), 1998.
5. P. Antsaklis, W. Kohn, A. Nerode and S. Sastry (eds.), *Hybrid systems IV.* Proc. of the Fourth Intern. Workshop on Hybrid Systems, Ithaca, New York, October 1996. Lecture Notes in Computer Science 1273, Springer, Berlin, 1997.
6. D.A. van Beek, S.H.F. Gordijn and J.E. Rooda, *Integrating continuous-time and discrete-event concepts in modelling and simulation of manufacturing machines.* Journal of Simulation Practice and Theory, 5, 1997, p. 653-669.
7. A. Bemporad and M. Morari, *Control of systems integrating logic, dynamics, and constraints.* Automatica, 35(3), 1999, p. 407-428.
8. V.D. Blondel and J.N. Tsitsiklis, *Complexity of stability and controllability of elementary hybrid systems.* Automatica, 35(3), 1999, p. 479-489.
9. W.M.G. van Bokhoven, *Piecewise-linear modelling and analysis.* PhD-thesis Eindhoven Univ. of Technology (Dept. of Electr. Eng.), The Netherlands, 1981.
10. P.P.J. van den Bosch, H. Butler, A.R.M. Soeterboek, and M.M.W.G. Zaat, Modelling and simulation with PSI/c. BOZA Automatisering B.V., Nuenen, The Netherlands, 1995.
11. P.P.J. van den Bosch and A.C. van der Klauw, *Modelling, Identification and Simulation of Dynamical Systems.* CRC-Press, 1994.
12. M.S. Branicky, *Stability theory for hybrid dynamical systems.* IEEE TAC, 43(4), 1998, pp. 475-482.
13. M.S. Branicky, V.S. Borkar and S.K. Mitter, *A unified framework for hybrid control: model and optimal control theory.* IEEE TAC 43(1), 1998, p. 31-45.
14. R.W. Brockett, *Hybrid models for motion control systems. In Essays in control.* H. Trentelman and J.C. Willems (eds.), Birkhauser, Boston, 1993, p. 29-53.
15. M.K. Çamlibel, W.P.M.H. Heemels and J.M. Schumacher, Dynamical analysis of linear passive networks with ideal diodes. Part II: Consistency of a time-stepping method. In preparation.
16. Zhou Chaochen, C.A.R. Hoare and A.P. Ravn, *A calculus of durations.* Information Processing Letters, 40(5), 1991, p. 269-276.
17. R.W. Cottle, J.-S. Pang and R.E. Stone, *The linear complementarity problem.* Academic Press, Boston, 1992.
18. R. David and H. Alla, *Petri nets for modelling of dynamic systems - a survey.* Automatica, 30(2), 1994, p. 175-202.
19. J. Davies, *Specification and proof in real-time CSP.* Cambridge University Press, 1993.
20. I. Demongodin and N.T. Koussoulas, *Differential Petri nets: representing continuous systems in a discrete-event world.* IEEE TAC 43(4), 1998, p. 573-579.
21. A. Deshpande, A. Göllü and P. Varaiya, *SHIFT: a formalism and a programming language for dynamic networks and hybrid automata.* In [5], p. 113-133.
22. C.A. Floudas, *Nonlinear and mixed-integer optimization.* Oxford University Press, 1995.
23. W.P.M.H. Heemels, J.M. Schumacher and S. Weiland, *Applications of complementarity systems.* In Proceedings European Control Conference, Karlsruhe (Germany), 1999.
24. W.P.M.H. Heemels, J.M. Schumacher and S. Weiland, *The rational complementarity problem.* Lin. Alg. Appl., 294(1-3), 1999., p. 93-135.
25. W.P.M.H. Heemels, A. van Zijl, R.J.A. Gorter, P.P.J. van den Bosch, S. Weiland, W.H.A. Hendrix and M.R. Vonder, *Asynchronous measurement and control: a case study on motor synchronisation.* To appear in Control Engineering Practice, 1999.
26. T.A. Henzinger, *Hybrid automata with finite bisimulations.* In Z. Fülöp and F. Gécseg (eds.) ICALP'95: Automata, Languages and Programming, Springer-Verlag, 1995, p. 324-335.
27. C.A.R. Hoare, *Communicating sequential processes.* Prentice-Hall, 1985.
28. M. Johansson and A. Rantzer, *Computation of piecewise quadratic Lyapunov functions for hybrid systems.* IEEE TAC, 43(3), 1998, p. 555-559.
29. G. Lafferiere, G.J. Pappas and S. Sastry, *Reachability analysis of hybrid systems using bisimulations.* In Proc. 37th IEEE Conf. on Decision and Control, Tampa (USA), 1998, p. 1623-1628.
30. D. Liberzon and A.S. Morse, *Benchmark poblems in stability and design of switched systems.* Nonlinear Control Abstracts, No. 1, 1999.
31. J. Lygros, D.N. Godbole and S. Sastry, *Verified hybrid controllers for automated vehicles.* IEEE TAC 43(4), 1998, 522-539.
32. N. Lynch, R. Segala, F. Vaandrager and H.B. Weinberg, *Hybrid I/O automata.* In [2], p. 496-510.
33. S.E. Mattsson, H. Elmqvist and J.F. Broenink, *Modelica: an international effort to design the next generation modelling language.* Journal A. 38(3), 1997, p. 16-19.
34. J.J. Moreau, *Numerical aspects of the sweeping process.* Preprint Elsevier.
35. A.S. Morse, S.S. Pantelides, S. Sastry and J.M. Schumacher (guest eds.), *A special issue on hybrid systems.* Automatica, 35(3), 1999.
36. A.M. Philips and M. Tomizuka, *Multirate estimation and control under time-varying data sampling with applications to information storage devices.* In Proc. American Control Conference, Evanstan Illinois, 1995, p. 4151-4155.
37. H.A. Preisig, M.J.H. Pijper and M. Weiss, *A discrete-event modelling procedure for continuous processes based on state-discretization.* Proc. IMACS Symposium on Mathematical Modelling, Vienna, 1997, p. 189-194.
38. A. Pnueli, *The temporal logic of programs.* In Proceedings of the 18th Annual Symposium on the Foundations of Computer Science, IEEE Computer Science Press, New York. 1977, p. 46-57.
39. A. Pnueli and J. Sifakis (guest eds.), *Special issue on hybrid systems,* Theoretical Computer Science 138, 1995.
40. A.J. van der Schaft and J.M. Schumacher, *Complementarity modelling of hybrid systems.* IEEE TAC 43(4), 1998, p. 483-491.
41. R.W. Sierenberg and O.B. de Gans, *Personal Prosim: a fully integrated simulation environment,* In Proceedings of the 1992 European simulation symposium, Dresden (Germany), p. 167-173.
42. U. Söderman and J.-E. Strömberg, *Switched bond graphs: multiport switches, mathematical characterization and systematic composition of computational models,* Techn. Report of Dept. of Computer and Information Science, Linköping University, Sweden, 1995. Available at http://www.ida.liu.se/ext/pur/enter/
43. E.D. Sontag, *Nonlinear regulation: the piecewise linear approach.* IEEE TAC, 26(2), 1981, p. 346-357.
44. D.E. Stewart, *Convergence of a time-stepping scheme for rigid body dynamics and resolution of Painlevé's problem.* Archive for Rational Mechanics and Analysis, 145(3), 1998, p. 215-260.
45. L.Tavernini, *Differential automata and their discrete simulators.* Nonlinear analysis, theory, methods and applications, 11(6), 1987, p. 665-683.
46. C. Tomlin, G.J. Pappas and S. Sastry, *Conflict resolution for air traffic management: a study in multiagent hybrid systems.* IEEE TAC, 43(4), 1998, p. 509-522.
47. V.I. Utkin, *Variable Structure Systems with Sliding Mode.* IEEE TAC, 22(1), 1977, p. 31-45.

Paul van den Bosch

Paul van den Bosch was born in 1948 in Rotterdam, the Netherlands. In April 1972 he graduated from the Delft University of Technology (Control Engineering) and obtained a PhD degree in 1983 with his dissertation on "Short term optimization of thermal power systems".
He was appointed professor in Control Engineering at the Delft University of Technology in 1988.
In September 1993 he was appointed to the chair of Measurement and Control at the Electrical Engineering Department of the Eindhoven University of Technology.
His interests include modelling, simulation, and control theory with applications in industrial processes and motion control.



Maurice Heemels

Maurice Heemels was born in 1972 in St. Odilienberg, the Netherlands.
In December 1995 he graduated from the Eindhoven University of Technology (System and Control Theory, Dept. of Mathematics).
He is currently working towards a PhD degree at the Eindhoven University of Technology (Dept. of Electrical Engineering) under supervision of Prof. Paul van den Bosch, Prof. Hans Schumacher and Dr. Siep Weiland. The PhD work is entitled "Linear complementarity systems: a study in hybrid dynamics."
His interests include modelling, analysis and control of hybrid systems and dynamics under inequality constraints.