Challenges and opportunities for hybrid systems in the automotive design flow

Andrea Balluchi
PARADES
Via S. Pantaleo, 66, 00186 Roma, Italy
balluchi@parades.rm.cnr.it

Luca Benvenuti
DIS, Università di Roma “La Sapienza”
Via Eudossiana 18, 00184 Roma, Italy
luca.benvenuti@uniroma1.it

Alberto L. Sangiovanni–Vincentelli
EECS Dept., Univ. of California at Berkeley,
CA 94720, USA
PARADES, Roma, Italy
alberto@eecs.berkeley.edu,alberto@parades.rm.cnr.it

First Draft - March 1, 2005

Abstract— **Automotive is certainly one of the most attractive and promising application domains for hybrid system techniques. Indeed, hybrid models and algorithms have already been successfully applied for automotive control designs. On the other hand, despite the significant advances achieved in the past few years, hybrid methods are in general still not mature enough for their effective introduction in the automotive industry design processes at large. In this report, we take a broad view of the development process for embedded control systems in the automotive industry with the purpose of identifying challenges and opportunities for hybrid systems in the design flow. We identify critical steps in the design flow and extract a number of open problems where, in our opinion, hybrid system technologies could play an important role.**

I. Introduction

The lack of an overall understanding of the interplay of the sub–systems and of the difficulties encountered in integrating very complex parts, system integration has become a nightmare in the automotive industry. Jurgen Hubbert, in charge of the Mercedes-Benz passenger car division, publicly stated in 2003: ”The industry is fighting to solve problems that are coming from electronics and companies that introduce new technologies face additional risks. We have experienced blackouts on our cockpit management and navigation command system and there have been problems with telephone connections and seat heating.” We believe that this state is the rule, not the exception, for the leading Original Equipment Manufacturers (OEMs) in today’s environment. The source of these problems is clearly the increased complexity but also the difficulty of the OEMs in managing the integration and maintenance process with subsystems that come from different suppliers who use different design methods, different software architecture, different hardware platforms, different (and often proprietary) Real-Time Operating Systems. Therefore, the need for for standards in the software and hardware domains that will allow plug-and-play of subsystems and their implementation are essential while the competitive advantage of an OEM will increasingly reside on essential functionalities (e.g. stability control).

To deliver more performing, less expensive, and safer cars with increasingly tighter time-to-market constraints imposed by worldwide competitiveness, the future development process for automotive electronic systems must provide solutions to:

- The design of complex functionality with tight requirements on safety and correctness;
- The design of distributed architectures consisting of several subsystems with constraints on non functional metrics such as cost, power consumption, weight, position, and reliability;
- The mapping of the functionality (often implemented as OEM application Software) onto the components of a distributed architecture with tight real-time and communication constraints.

Most of the car manufacturers outsource the design and production of embedded controllers to suppliers (so–called Tier–1 companies), which in turn buy IC components and other devices by third parties (so–called Tier–2 companies). Embedded controllers are often developed by different Tier–1 companies and are requested to operate in coordination on a same model of a car. Moreover, in the development of an embedded controller, the supplier has to integrate some IPs (Intellectual Properties) provided by the car manufacturer at different levels of details (algorithms, legacy code) and, in the near future, possibly by third parties.

To cope with this challenging context, the design flow has to be significantly improved. Hybrid systems techniques can have an important role in this respect.

Successful approaches to design of control algorithms using hybrid system methodologies had been presented in the literature, e.g. cut-off control [7], intake throttle valve control [8], actual engaged gear identification [5], adaptive cruise control [15]. However, despite the significant advances of the past few years, hybrid system methodologies are not mature yet for an effective introduction in the automotive industry. On the other hand, hybrid system techniques may have an important impact on several critical open problems in the overall design flow that go beyond the classical controller synthesis step. In this paper, we analyze the design flow for embedded controllers in the automotive industry, with the purpose of identifying challenges and opportunities for hybrid system technologies. We present a detailed description of the typical design flow for embedded controllers adopted by the automotive industry with particular emphasis on the Tier–1 supplier problems as they are more amenable to the use of hybrid techniques. For each design step, we identify critical phases and bottle-neck problems and we extract relevant open problems that hybrid system technologies may contribute to solve.

The rest of the paper is organized as follows. In Section II, an overview of the automotive embedded controller design scenario is presented, emphasizing constraints and boundaries. The design flow and the integration and validation flow are respectively discussed in Section III and Section IV. Some concluding remarks are given in Section V.

II. Design scenario and design flow

In today cars, the electronic control system is a networked system with a dedicated ECU (Electronic Control Unit) for each subsystem: e.g. engine control unit, gear–box controller, ABS (Anti–lock Braking System), ESP (Electronic Stability Program), dashboard controller, and VDC (Vehicle Dynamic Control). The ECUs are connected via a network specifically designed for automotive applications, such as CAN, and Flexray. In Section II-A, a standard design flow adopted by Tier–1 companies for the development of each single ECU is described. Due to the increased number of subsystems equipped with electronic control, this one–to–one mapping of functionalities to ECUs is no longer feasible. In Europe, the AUTOSAR initiative has been established to define common standards and methodologies to allow better mapping of functionality on the embedded systems. The objectives of this initiatives are briefly recalled in Section II-B. In Section II-C, we describe the peculiarities of customer requirements for automotive Tier–1 companies that are quite different with respect to other application domains.

A. Tier–1 companies' design flow

The standard design flow adopted by Tier–1 companies is the so–called "V–cycle" shown in Figure 1. The top–down left branch represents the synthesis flow. The bottom–up right branch is the integration and validation flow.

The synthesis flow is articulated in the following steps:

- *System specification*: formalization of system specification; coherence analysis; evaluation of feasibility; completion of underspecified behaviors; abstraction of lower layers customer's requirements.
- *Functional deployment*: system decomposition; definition of subsystems' specifications; design of control algorithm architecture; definition of specifications for each control algorithm
- *Control system*: plant modeling (model development, identification, validation); controller synthesis (plant model and specifications analysis, algorithm development, controller validation); fast prototyping.
- *HW/SW components*: formal specifications for implementation; design of hardware and software architectures; hardware design; software development and automatic code generation; RTOS (Real–time Operating System); etc[1].

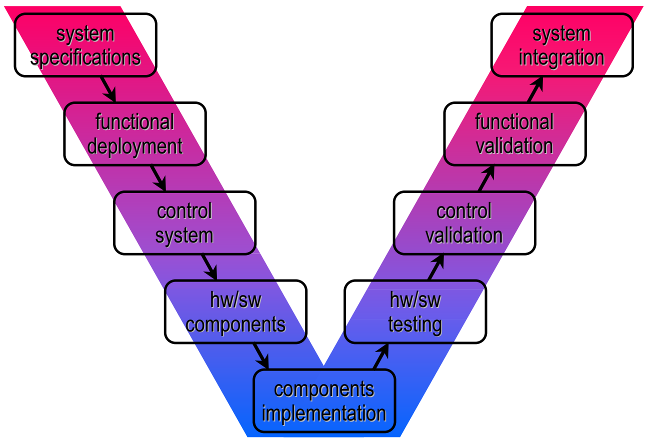

Fig. 1. Design and integration flow.

The synthesis flow terminates with the development of the components (components implementation). From this point starts the integration and validation flow that goes through the following steps:

- *HW/SW testing*: emulation; software / processor / hardware–in–the–loop testing.
- *Control validation*: in–vehicle validation of the control algorithm integration.
- *Functional validation*: in–vehicle validation of the subsystem behavior;
- *System integration*: validation with respect to system specification; calibration (off–line and on–line DoE screening, refinement and optimization, pre-calibration and fine tuning); testing of integration with other embedded controllers.

B. AUTOSAR

The rigid partition one subsystem — one ECU adopted today in the automotive industry is by no means cost appeal-

[1]This layer is only sketched, since low relevant to hybrid systems applications.

ing and often it is not efficient in terms of communication and synchronization. There is a clear necessity to have, in the near future, a more flexible approach that allows to map more functionalities on a same ECU and distribute other functionalities on several ones. The advantages of the new approach are apparent: cost reduction, flexibility, redundancy, etc. Moreover, the new approach should allow to support functionalities developed by either different suppliers or by the OEM, on a same ECU. In fact, currently electronic architectures are based on proprietary solutions, which seldom allow the exchange of applications between both automotive OEMs and their suppliers.

The objective of the AUTOSAR initiative [13], [2], promoted by leading OEMs and Tier–1 suppliers, is establishing an open standard for automotive electric/electronic architectures. The standard will serve as a basic infrastructure for the management of functionalities, within both future applications and standard software modules, allowing collaboration between companies on the development of basic functionalities while providing a platform that encourages competition on innovative functionalities.

The scope of AUTOSAR includes all vehicle domains. Its declared goals are:

- Implementation and standardization of basic system functionalities as an OEM wide "Standard Core" solution;
- Scalability to different vehicle and platform variants;
- Transferability of functionalities throughout network;
- Integration of functional modules from multiple suppliers;
- Consideration of availability and safety requirements;
- Redundancy activation;
- Maintainability throughout the whole "Product Life Cycle";
- Increased use of "Commercial off the shelf hardware";
- Software updates and upgrades over vehicle lifetime.

The AUTOSAR standard will serve as a platform upon which future vehicle applications will be implemented and will also serve to minimize the current barriers between functional domains. It will, therefore, be possible to map functionalities and functional networks to different ECUs in the system, almost independently from the associated hardware. To achieve the technical goals modularity, scalability, transferability and re-usability of functionalities, AUTOSAR will provide a common software infrastructure for automotive systems of all vehicle domains based on standardized interfaces for the different layers. In addition to the technical question, there are serious implications on IP exchange and protection to be solved.

C. Customer requirements

The customer requirements, issued by the OEM, define the desired behavior of the vehicle that should be achieved by the design of the control system. The specifications regard

- *performance and driveability* - dynamic behavior of the vehicle, driver assistance, detection and suppression of critical dynamic vehicle states, comfort;
- *fuel consumption*;
- *legal requirements* - environment and safety.

Ideally, requirements should define *what* the controlled system must, do, i.e. they should express at the system specification level of the design flow the desired closed–loop specification. However, in the automotive industry, often customer requirements define also in part *how* the requested behavior has to be obtained. For instance, OEM requirements may impose some details on the control algorithm architecture at the functional level, some particular control algorithms at the control system level, the use of OEM legacy code or the choice of a particular micro-controller at the HW/SW component level. In sum, customer requirements contain specifications that are not only at the system specification level, but are spread over all levels of the design flow. That is, they are heterogeneous – because they are related to different levels of abstraction – and often not complete. *Customer requirement management* analyzes OEM requirements and clearly identifies this heterogeneity to allocate and handle each requirement at the correct phase of the design flow. Heterogeneity of customer requirements is a critical issue in the design flow for Tier–1 companies and it is peculiar to the automotive industry. In the near future, Tier–1 companies will be requested to integrate on a single ECU functionalities and legacy code provided by OEMs and third parties. In our opinion, handling this composition of IPs at different levels of abstraction, developed independently by different companies, is nearly impossible. In fact, software components can not be composed with guaranteed closed–loop behavior if their interaction is not represented, formalized and tested at the algorithm level. Hence, the right level of composition is at the control system level.

Due to the automotive design cycle and the demanding constraints on ECU development time, at the beginning of the design, only a draft of the customer requirements is handed out to the Tier–1 companies. This first draft is usually incomplete in many parts; it will be completed later in parallel to the development of the ECU. To complicate matters further, most of the requirements are given in natural language and are not formally specified.

III. Synthesis flow

The design of automotive ECUs is subject to very critical constraints on cost and time–to–market. Successful designs, in which costly and time consuming re–design cycles are avoided, can only be achieved using efficient design methodologies that allow for component reuse and evaluation of platform requirements at the early stages of the design flow. Design methodologies must foster component reuse at all layers of the design flow from system specification to control algorithm and HW/SW components (see [1], [6]). To do so, design methodologies should provide means for the:

- evaluation of the compliance of the reused component with the new context requirements;
- correct integration with other (either reused or not) components;
- cost evaluation.

Furthermore, to manage the complexity of the design and obtain ECUs with guaranteed performances and reduced cost, a model-based design approach has to be adopted at all levels of the design flow. Specifications, functional architectures, algorithms, and implementation architectures should be represented formally by models. The complementary and interoperability of tools supporting the model description at the different levels of the design flow is a key issue. The design chain should be refined to achieve higher degrees of integration and standardization. In this section, we describe the synthesis part of the automotive design flow covering the levels of system specification, functional deployment, control system and HW/SW components. The discussion will focus on the design flow for an engine control unit, which is taken as an example. Emphasis will be placed on the aspects which we believe hybrid system techniques may have relevant impact on, while details of the design with no relation to hybrid systems will be slightly mentioned.

A. System specification

System specification are defined in terms of a number of operation modes characterized by different controlled variables and objectives. Specifications are given in terms of requirements on performance, driveability, fuel consumption, emissions and safety.

System specification regards both discrete and continuous behaviors. Discrete specifications are often given in natural language and only sometimes formalized in some discrete modeling framework. The specifications defines modes and switching conditions. Continuous specifications are given following classical methodologies in terms of steady-state/transient response, frequency domain, robustness and parameter sensibility, disturbance rejection, control effort, cost functions, and constraints.

Operation modes are organized in a hierarchical structure. Some of them are introduced to specify smooth transitions between different operating conditions.

Often, for both discrete and continuous specifications, requirements are given by specifying requested behaviors on hybrid input/output evolutions. In addition, critical maneuvers for which the behavior requested by the specifications should be guaranteed (possibly up to some allowed degradation) are also identified.

The degree of detail given by the OEMs in describing system specifications is not uniform. Depending on the importance placed by the OEM on each single behavior, functionality or constraint, a different degree of accuracy in describing the requirement itself is used. In particular, some behaviors may result only vaguely specified: in this case, under-specified system requirements are completed by the Tier–1 supplier on the basis of its own experience while trying to maximize reusability for future developments. On the other hand, there are also behaviors that are very detailed in the customer requirements to the the point that the OEM imposes not only a system level requirement but also a particular solution to satisfy it, resulting in an undesirable (at least from an ideal point of view) over-specification. In *customer requirement analysis*, these customer requirements are inserted at the right level of abstraction as constraints that limit the degrees of freedom of the designers active at that level. Since these constraints are often the result of decisions based on insufficient analysis, the feasible design space may be empty thus causing unnecessary design cycles. We do believe that care must be exercised when constraint are entered at abstraction levels that are non appropriate with respect to the role of the company that specifies them.

Hybrid system contributions. Hybrid methods have impact in the following areas:

- Entering designs at the correct level of abstraction does require the specification of models that contain both continuous and discrete components as they often refer to the properties of the closed loop system where the model of the plant is continuous or hybrid itself.
- Tools for system specification, requirement management and system design, validation and verification must be developed to deal with hybrid models.
- Since customer requirements contains details regarding several levels of the design flow, to achieve a complete representation of the system at system specification level, abstraction techniques that deal with hybrid systems for projecting lower–levels specifications back to upper–levels must be developed.
- Hybrid techniques and supporting tools to perform coherence and feasibility analysis at system specification levels have to be developed as well.

B. Functional deployment

In the first stage of the design, functional deployment, the system is viewed as a collection of interacting components. The decomposition is based on the understanding of the physical process of interest. The decomposition process is clearly a key step towards a good quality design, since it leads to a design process that can be carried out as independently as possible for each component (see [1] for more details). A typical decomposition for engine control is shown in Figure 2. The objectives and constraints that define the system specification are distributed among the components by the functional deployment process so that the composition of the behaviors of the components is guaranteed to meet the constraints and the objectives required for the overall controlled system.

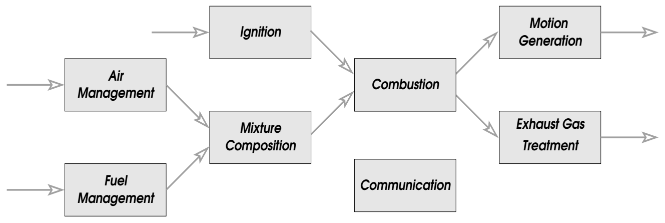

Fig. 2. Functional decomposition.

In the second stage of the functional deployment, the control algorithms architecture is defined. Namely, the set of control algorithms to be developed for each function and the topology of interconnection are determined. Furthermore, for each control algorithm, desired closed–loop specifications are defined to achieve the requested behavior for each functional component. This process is mainly guided by the experience of system engineers, with little support of methodologies and tools. Moreover, the sets of measurable and actuated quantities, which will constitute the sets of, respectively, inputs and outputs to the ECU, are often defined by the OEM and are specified in the customer requirements. In fact, the OEM often defines also sensors and actuators to be used, since they have a major impact on the cost of the control system. In addition, customer requirements include details on the topology of the control algorithms architecture that further constrains the functional deployment process.

The results of the functional deployment design stage are: the control algorithms architecture and the desired closed–loop specification for each control algorithm. The traceability of customer requirements guarantees that each requirement has been handled. Correct implementation of traceability during functional deployment is fundamental, since it establishes the relations between each customer requirement and the specifications of the relevant control algorithms.

Hybrid system contributions. Hybrid techniques could help addressing the following issues in functional deployment:

- Hybrid formalisms are required to support the description of
  - the functional decomposition and the desired behavior for each functional component;
  - the architecture of control algorithms, sensors and actuators, for each functional component;
  - the desired requirements for each control algorithm obtained from the functional deployment process.
- Development of methodologies and tools for the evaluation, the exploration and possibly the synthesis of functional behaviors from system specification and control algorithm requirements from desired functional behaviors. The methodologies should foster re–use at the functional level, obtained by exposing description of existing solutions, characterized by appropriate cost models that allow comparisons among different solutions. The tools should be integrated with requirement management and system engineering tools.
- Methodologies and tools for the validation of selected functional behaviors against system specification and control algorithm specifications against desired functional behaviors should also be developed.

C. Control system

At the control system level, the algorithms to be implemented in the architecture defined at the functional level are designed. All control algorithms have to meet the assigned specification, so that their composition within a functional component exhibits the required behavior defined during functional deployment.

In general, the design process for each control algorithm involves

- Plant modeling:
  - model development;
  - identification;
  - validation;
- Controller synthesis:
  - plant model and specifications analysis;
  - algorithm development;
  - controller validation;
- Fast prototyping.

However, if part of the algorithms are re–used from previous designs, the entire three–step flow is often only partially performed.

Models of the plant interacting with the control algorithm under design may already be available from previews designs. In this case, only some adjustment of sensitive parameters, along with a coarse validation, could be sufficient to obtain reliable models. If not, rigorous identification and validation has to be performed. The complete plant modeling phase is necessary for the refinement of un-satisfactory existing models, when major changes in the plant have been made or for the development of new functionalities.

Model–based design is universally recognized to be a good approach to control algorithm synthesis and is increasingly used in the automotive industry. Using block diagram–based modeling tools, such as Matlab/Simulink, control algorithms are designed and initial validation in off–line simulation is performed. Then, models of the control algorithms are the basis for all subsequent development stages. The advantages are obvious:

- sharing models reduces the risk of mistakes and shortens the development cycles;
- design choices can be explored and evaluated much faster and more reliably;
- the result of a model–based development process is an optimized and fully tested system.

In addition, model–based design allows efficient, time-saving and cost-effective reuse of control algorithms. Re–use cannot be based on paper documentation of software; it does require executable models of the control algorithms. In fact, in many applications, most of the control algorithms are re–used from past designs. Modularization and standardization of control algorithms and their interfaces are essential here. In the plant model and specifications analysis stage, we first analyze whether an available algorithm can meet the specification or a new design is needed. If the algorithm is obtained from designs done in the past, then the algorithm development stage may involve some minor changes to the re–used algorithm in order to completely cover the new specification. In these cases, the controller validation stage is the most important step to ensure that re–use was successful.

Design of new control algorithms, necessary when re–use cannot be applied due to major changes in the specification or when new functionalities have to be developed, require the performance of the entire three-step flow for controller synthesis.

Fast prototyping is adopted when either control algorithms are designed for new functionalities or major redesigned has occurred to meet more stringent specifications.

In the following sections, each design step is discussed in details.

1) plant modeling: In this section, the three steps of the plant modeling phase, namely model development, identification, and validation, are illustrated. Aspects relevant to the introduction of hybrid system modeling techniques in automotive applications are outlined.

a) model development: Traditionally, control engineers adopt mean–value models to represent the behavior of automotive subsystems. However, the need for hybrid system formalisms to model the behavior of systems in automotive applications is apparent in many cases.

To demonstrate that this is indeed the case, let us consider for instance the behavior of an internal combustion engine, and the one of the fuel–injection and spark–ignition subsystems. An accurate model of the engine has a natural hybrid representation because the cylinders have four modes of operation corresponding to the stroke they are in (which can be represented by a finite-state model) while power–train and air dynamics are continuous-time processes. In addition, these processes interact tightly. In fact, the timing of the transitions between two phases of the cylinders is determined by the continuous motion of the power–train, which, in turn, depends on the torque produced by each piston. In [3], we showed that the engine can be modeled using a hybrid system composed of interacting finite–state machines, discrete–event systems and continuous–time systems. The hybrid nature of the behaviors is also evident if we look at the different types of input and output signals for the internal combustion engine, and the fuel injection and spark ignition systems, summarized in Table I. The hybrid nature of the behaviors is not limited to the input–output interfaces of the models. For instance, the model of an automotive drive line has several internal discrete–continuous interactions. In [4], a detailed model with up to 6048 discrete state combinations and 12 continuous state variables was presented. The hybrid model accurately represents discontinuities distributed along the drive line due to engine suspension, clutch, gear, elastic torsional characteristic, tires, frictions and backlashes. Finally, models of automotive subsystems are often highly nonlinear. In engine modeling, nonlinearities arise from fluid–dynamics and thermodynamics phenomena (e.g. volumetric efficiency, engine torque, emissions) and are usually represented by piece–wise affine maps.

To conclude this brief discussion, we mention the increasing importance of human factors in automotive control design. Since the closed–loop system is a man–in–the–loop system, to design and validate correctly control algorithms that interacts with the driver, it is necessary to understand and model the behavior of the driver. This approach will become increasingly important with the expansion of x–by–wire applications (driving, steering, braking) that require a fine design of man–machine interfaces. The models of the driver must include perception (interesting also for passengers in some applications), actuation, open-loop and closed-loop control actions. As described in [14], the behavior of the driver in the longitudinal control of the car can be characterized using 13 main operation modes, such as idle, tip in, acceleration, and pedal steering control. The driver's behavior in each operation mode is further decoupled in a number of tasks, which define about one hundred operation sub–modes. For instance, the idle mode is split in after cut off, after highway run, after vehicle stop, after hot start, after air-conditioning switching on. As far as the longitudinal motion of the car is concerned, the driver and the passengers are sensitive to vehicle shuffle, pedal response, oscillations of engine speed and torque. Performance and driveability of the car in each operation mode are assessed by experienced test drivers that quote on a scale from 1 to 10 the driving feeling. Driver's models are then used to represent

- the behavior of the driver in open–loop and closed–loop control for the design of the man–machine interface in x–by–wire control systems;
- driveability perception to obtain analytical and repeatable specifications for driveability controllers design.

The development of models for car driveability perception requires intensive study of human perception and assessment criteria, different vehicles and different drivers, driver interviews (during and after driving), data recording and analysis.

Plant models and driver models have to be characterized by expressing both their nominal behaviors and main uncertainties due to parameter and time-varying perturbations, representing production diversity and aging. Relevant disturbances should also be modeled. Depending on the context, either deterministic or statistic/stochastic approaches could

Internal Combustion Engine Hybrid Model

	discrete time		*continuous time*	
	discrete value	*continuous value*	*discrete value*	*continuous value*
inputs	spark ignition	injected fuel air charge exhaust gas conc.		engine speed
outputs	crankshaft events	air-to-fuel ratio		engine torque engine temperature engine exhaust gas

Direct Injection Fuel System Hybrid Model

	discrete time		*continuous time*	
	discrete value	*continuous value*	*discrete value*	*continuous value*
inputs		pressure valve cmd	injection signal	
outputs		injected fuel		rail pressure fuel temperature

Spark Ignition Hybrid Model

	discrete time		*continuous time*	
	discrete value	*continuous value*	*discrete value*	*continuous value*
inputs	spark command		ignition coil cmd	
outputs	spark ignition			

TABLE I

TIME DOMAIN AND VALUE DOMAIN CLASSIFICATION OF SIGNALS FOR INTERNAL COMBUSTION ENGINE MODELING.

be employed.

Hybrid system contributions. Plant model development requires extensive use of hybrid modeling techniques:

- Hybrid deterministic and stochastic formalisms, including FSM, DES, DT, CT, PDA, for representing interacting behaviors of different nature are essential.
- Such hybrid formalisms should be supported by appropriate tools for hybrid model description and simulation.

b) identification: In current practice, parameter identification is mostly based on steady–state measurements, obtained using either manually defined set–points or automatic on–line screening. More details on these approaches are given below in the description of the calibration process. Dynamic parameters are often either obtained analytically (e.g. intake manifold model) or from step responses. However, step response and other classical identification methods can be used to identify models representing standard continuous evolutions only, such as those exhibited by mean–value models. When applied to hybrid models, classical techniques can only be used to identify the plant model separately in each discrete mode. They hardly succeed in identifying parameters related to switching conditions and cannot be applied to black–box hybrid model identification. A major drawback of this approach to identification is that it requires a relevant amount of experimental data. Hence, it is very time–consuming and costly.

A common problem in engine control is the representation of nonlinearities. Nonlinearities are usually represented by piece–wise affine functions, with the following limitations:

- the partition of the domain is based on non-uniform grids (independent breakpoints for each axis);
- parameter identification is obtained from steady-state experimental data only;
- only $R \to R$ and $R^2 \to R$ nonlinearities are expressed (higher dimension nonlinearities are modeled as their products).

Hybrid system contributions. Among the many open problems in hybrid system identification, the following ones are the most relevant to automotive applications:

- The availability of hybrid system identification techniques using transient data, including mode switching, would allow to increase identification accuracy, reduce the amount of experimental data needed and identify all parameters in hybrid models. Efficient identification techniques for hybrid systems will also give the opportunity for modeling more complex hybrid behaviors that are currently abstracted due to the difficulties in the identification process.
- Efficient hybrid techniques for the representation and identification of nonlinearities, as either piece–wise affine functions (see [10]) or piece–wise polynomial functions, would produce majors impact in the design:
  - domain partition could be optimized (possibly not grid-based), achieving increased accuracy and reducing model complexity;
  - parameter identification accuracy could be improved;
  - higher dimension nonlinearities $R^p \to R$ could be

represented and identified.

c) validation: The selection of test patterns for model validation is a crucial issue in the validation process. Classical techniques allow to assess the richness of sets of test patterns for the validation of continuous models. These techniques can be used in automotive applications to assess richness of validation patterns for continuous evolutions of the plant. However, the problem remains open for hybrid model validations. This topic is further discussed in Section III-C.2.c, where automatic test pattern generation for controller validation is analyzed.

Hybrid system contributions. Validation of hybrid models is a very complex task not sufficiently investigated in the literature. In particular, the following open problems must be addressed:

- Methodologies for automatic generation of extensive validation patterns for hybrid models;
- Techniques for the assessment of the completeness of validation patterns. This problem can be formalized in the framework of reachability analysis. Interesting approaches have been proposed using the concepts of structural coverage and data coverage []
  - condition coverage: how many transition guards have been tested?
  - decision coverage: how many locations have been tested?
  - modified condition/decision coverage: how many input combinations of guard conditions have been tested?

2) controller synthesis: In this section, the activities related to controller synthesis are presented by discussing: the analysis of experimental data, plant model and specifications; algorithms development; controller validation.

a) analysis of experimental data, plant model and specifications: Typically the design process of a control algorithm for a new application starts with the analysis of some experimental data obtained either with open–loop control or with some very elementary closed–loop algorithm. Open loop simulation of the plant model is also very useful in this phase. Often the model represents a partially controlled plant and contains the effect of some inner–loop controllers. Open–loop simulation of hybrid models requires the definition of discrete-time, event-based and continuous time input actions, representing either hybrid inputs and references or perturbations. The assessment of classical structural properties, such as reachability, observability, stabilizability, passivity [9], on the open–loop plant model is of interest in this phase. In addition, quantitative analysis is very useful to understand the strengths and weaknesses of the design. It is interesting to obtain by performance and perturbations/uncertainties analysis an evaluation of quantities such as stability margins, most critical perturbations/uncertainties, robust stability margins, reachability and observability measures in the state space.

Hybrid system contributions. Classical concepts and techniques for system analysis cannot be applied to hybrid systems (e.g. switching systems stability has no direct relation with subsystems poles). Unfortunately hybrid system theory has not been developed to a point to be trusted for model analysis:

- Some fundamental properties have not been formally defined yet and tests are not available for verifying most of the properties.
- Efficient implementation of tests will be necessary for automatic evaluation, since often manual testing is prohibitively expensive for hybrid system properties.
- Analysis tools must be integrated with standard system engineering tools.

b) algorithms development: We begin by illustrating some general characteristics of the algorithms implemented in automotive ECUs. For example, consider engine control units. These ECUs have usually more than one hundred I/O signals, implement up to two hundreds algorithms, share with the other related ECUs approximately fifty signals. Typically, an ECU for automotive applications is not a stand–alone system controlling a mechanical subsystem in the car, but interacts with other ECUs by asynchronous communication over a communication network. Sometimes this interaction is very tight, as in the case of gear–box controllers and the engine controllers. Each ECU is a multirate control system composed of nested control loops, with frequency and phase drifts between fixed sampling–time actions and even driven actions. The controller performs both continuous and discrete computations, where the discrete part is often more important than the continuous one. Often control algorithms are characterized by many operation modes, that are conceived to cover the entire life–time of the product: starting from in–factory operations before car installation, configuration, first power–on, power–on, functioning, power–off, connection to diagnostic tools. During standard functioning, control strategies can be either at the nominal operation mode or at one of several recovery modes. A significant number of algorithms are dedicated to the computation of switching conditions and controller initializations.

A short and by no–means exhaustive list of control actions for which hybrid system design is particularly interesting is as follows: fuel injection, spark ignition, throttle valve control (especially with stepper motor), electromechanical intake/exhaust valve control, engine start-up and stroke detection, crankshaft sensor management, VGT and EGR actuation (hysteresis management), emission control (cold start-up, lambda on/off sensor feedback), longitudinal oscillations control (backlash and elasticity discontinuities), gear–box control (servo-actuation in traditional gear shift systems), cruise control and adaptive cruise control, diagnosis algorithms (signals and functionalities on-line monitoring), algorithms for fault-tolerance and safety and recovery (degraded mode activation). Automotive applications are characterized by challenging aspects. For example:

- highly non–linear behaviors;
- input functions vanishing at the equilibrium points preventing the use of linearization techniques;
- adaptation with lack of persistence of excitation;
- control loops affected by long delays that are often uncertain and/or time-varying;
- lack of direct feedback of many significant physical signals that have to be estimated.

A special discussion should be devoted to diagnostic algorithms, since they represent a major part of the strategies implemented in automotive ECUs. For engine control, the implementation of diagnosis algorithms is enforced by the legislation: OBDII (On Board Diagnosis II) in USA and EOBD (European On Board Diagnosis) in EU. In general, these requirements specify that every fault, malfunction or simple component degradation that leads to pollutant emissions over given thresholds should be diagnosed and signaled to the driver. This requirement has a significant impact on ECU design, since it implies the development of many on–line diagnostic algorithms [12]. Failures in on–line diagnosis have serious consequences. If an engine failure is present and is not detected according to what is imposed by the legislation, then the car may not receive the necessary authorization to be offered on the market. On the other hand, excess of false alarms in diagnosis may result in very annoying situations for the driver, such as engine lock even when everything is working properly. Next-generation x-by-wire vehicle systems (e.g. brake-by-wire and steer-by-wire) will require additional design efforts to achieve the requested levels of fault tolerance and safety.

Both specifications and accurate models of the plant are often hybrid in automotive applications. The methodology currently adopted for algorithm development is rather crude and can be summarized as follows. The continuous functionalities to be implemented in the controller are designed based on mean–value models of the plant, with some *ad hoc* solutions to manage hybrid system issues (such as synchronization with event–based behaviors). If the resulting behavior is not satisfactory under some specific conditions, then the controller is modified to detect critical behaviors and operate consequently (introducing further control switching). The discrete functionalities of the controller are designed by direct implementation of non–formalized specifications. Design methodologies and corresponding tools for the synthesis of discrete systems (FSM, DES, etc) are usually not employed. The discrete behavior of the controller is not obtained from automatic synthesis of a formalized specification, as for instance it is done in hardware design. If the algorithm is not designed from scratch, but is obtained by elaborating existing solutions, as is often the case, then additional operation modes may be introduced to comply with the new specification. This results in a non–optimized controller structure. Structured approaches to the integrated design of the controller that allow to satisfy hybrid specifications considering hybrid models of the plant are not adopted as yet even though they have obvious advantages over the heuristics that permeate the present approaches.

The disadvantages of the current methodology can be summarized as follows:

- long development time due to redesign cycles;
- un-satisfactory performances;
- extensive testing needed;
- time-consuming and expensive calibration;
- no guaranteed behavior and low reliability (testing can never reach complete coverage; the more frequent malfunctioning and the ones with more serious consequences are related to the discrete functionalities);
- low or non guaranteed robustness with respect to product diversity, aging, perturbations;
- complex solutions with high implementation cost.

Hybrid system contributions. Hybrid system techniques can significantly contribute to the improvement of control algorithm design in automotive applications. The introduction of hybrid synthesis techniques should be aimed to:

- shorten the algorithm development time;
- reduce testing effort;
- reduce calibration parameters and provide automatic calibration techniques;
- improve closed–loop performances;
- guarantee correct closed–loop behavior and reliability;
- achieve and guarantee desired robustness;
- reduce implementation cost.

Most of the analytical approaches so far proposed for controller design using hybrid system techniques are quite complex. Usually, the application of these techniques requires designers that are trained in hybrid systems and necessitates long development times. As a consequence, the hybrid system design process results too expensive for the human resources commonly deployed in automotive system engineering. Hence, for a profitable introduction of hybrid system design techniques, it is essential that methodologies be supported by efficient tools that allow fast and easy designs. Hybrid model predictive control is a good example in hybrid system research where the development of the methodology was supported by a good effort in design tool development [11]. Efficient tools for both validation (possibly verification) and calibration are also mandatory as discussed below.

c) controller validation: Control algorithms are validated in extensive simulations of the closed–loop models. The Mathworks Inc. tool–suite is widely used in the automotive industry for simulation purposes. However, since the semantic of the Simulink–Stateflow simulation environment is not well–defined and documented, the simulation of hybrid models may be problematic at times. Validation via extensive simulation is time–consuming and hence expensive.

The designers, based on their experience, devise critical trajectories to test the behavior of the closed–loop system in the perceived worst–case conditions. Some of the critical maneuvers may be provided by the system specifications. Furthermore, a rough investigation on the robustness properties of control algorithms is obtained by screening the most critical parameters and uncertainties and applying critical perturbations. In the current design flow, there is no automatic approach to the validation of performance specifications. Some approaches for automatic test patterns generation are under investigation. To the best of our knowledge, there is no tool available in the market for performance analysis, robust stability and robust performance analysis. Formal verification for both continuous and for discrete specification is not adopted.

Hybrid system contributions. Due to complexity of the plant–controller interactions, the non negligible effects of the implementation, the large uncertainties in the plant given by product diversity and aging, validation of control algorithms is one of the hottest topics in automotive industry. Today, the quality of the validation step is not satisfactory and important improvements in validation will be necessary to cope with the safety issues that will be raised by next generation x–by–wire systems. Ideally, validation and formal verification should be completely automatic. Hybrid system techniques can contribute significantly to the improvement of the validation process:

- Validation has to be supported by tools for
  - efficient simulations of hybrid closed–loop models;
  - stability and performance analysis;
  - robust stability and robust performance analysis;
  - invariant set and robust invariant set computations.
- Methodologies and tools should be developed for
  - automatic validation against formalized hybrid performance specifications;
  - automatic validation of safety relevant conditions;
  - automatic optimized test patterns generation reaching specified level of coverage.
- Interesting validation problems are related to the computation of conservative approximations for the largest sets of
  - parameter uncertainties,
  - calibration parameters,
  - implementation parameters (e.g. sampling–period, latency, jitter, computation precision, etc.),

  for which the desired performances are achieved.
- Some classes of algorithms that require intensive and complex validation are
  - diagnosis algorithms;
  - safety critical algorithms;
  - algorithms preventing the system to stall (e.g. idle speed control).

3) fast prototyping: Control algorithms either significantly redesigned or developed for the first time are usually tested using fast prototyping. Fast prototyping equipments allow to perform in–vehicle experiments with minor configuration efforts using a bypass method. The software implementing the algorithm under development is off–loaded from the production ECU to the fast prototyping unit that is connected to the ECU. The flow of data is diverted from the ECU to the prototyping system and back, while the ECU runs the legacy code. The bypass method is ideally suited to replace, step–by–step, obsolete ECU code by new one. Completely new developments also benefit from the bypass method, as the I/O interfaces of an existing ECU can be used. Connecting the fast prototyping unit to a notebook enables the control engineer to control the experiment from the passenger seat.

D. HW/SW components

The design of HW/SW implementation of ECUs follows the standard methodologies for hardware and software development. It is interesting to discuss the phase of definition of the specifications for the HW/SW implementation of the control algorithms, because hybrid formalisms may have interesting applications in this regard. This phase of the design flow is very critical since it is located at the boundary between different disciplines, namely control engineering and software/hardware design. The methodologies and the design tools in the control domain and the HW and SW implementation domains are often not very well integrated and this is often the cause of many design errors. The specifications for the HW/SW implementation must include all details necessary for a correct implementation of the algorithms i.e., they must provide:

- complete description of the algorithm;
- specification of the computation accuracy
  - in the value domain: precision for each computation chain (for fixed–point arithmetic implementation), threshold detection bounds, etc;
  - in the time domain: bounds for latency, jitter, delay in event detection, etc.
- execution order and synchronization;
- priorities in case of resource sharing (cpu, communication, etc);
- communication specifications;
- data storage requirements (e.g. variables in EEPROM).

In addition, the specifications for the HW/SW implementation have to be derived from executable models, according to the model–based design approach. These models should be integrated with tools for automatic code generation. Finally, the specifications for the HW/SW implementation should ideally provide executable acceptance tests that can guarantee that the computation accuracy obtained in the HW/SW implementation is good enough.

Hybrid system contributions. Hybrid formalisms can successfully support specifications for the HW/SW implementation:

- Hybrid tools suitable for the description of the implementation requirements of the algorithms have to:
  - support the specification of the algorithm behavior, the computation accuracy and the other implementation requirements and constraints mentioned above;
  - support description of implementation acceptance tests;
  - to be efficiently integrated with software and hardware development tools and tools for automatic code generation.
- Methodologies and tools for defining and validating implementation constraints should be developed:
  - the degradation of the execution of control algorithms due to the implementation on bounded resource platforms has to be exported and modeled at the control system level to obtain constraints for the implementation;
  - these constraints should be formally specified in the HW/SW implementation requirements along with executable acceptance tests;
  - tools should support the validation of the HW/SW implementation by running the acceptance tests.

IV. Integration and testing

[...]

V. Concluding Remarks

Assuming the design methodology and the infrastructure for design chain integration are all in place, what will be the implication on the industrial segment structure? Today, the roles of car makers, Tier 1 and 2 Suppliers are relatively stable but they are undergoing a period of stress due to the increased importance of electronics and its added value. We mention the desire of car makers to gain a stronger grip on the integration process and on the critical parts of the electronics subsystems. At the same time, there is evidence that sharing IPs among car makers and Tier 1 suppliers could improve substantially time-to-market, development and maintenance costs. The essential technical problem to solve for this vision is the establishment of standards for interoperability among IPs and tools. AUTOSAR has this goal very clear in mind. However, there are technical and business challenges to overcome. In particular, from the technical point of view, while sharing algorithms and functional designs seems feasible at this time once the semantic platform issues are squared away, the sharing of real-time software is difficult even assuming substantial improvements in design methods and technology, if run-time efficiency has to be retained. The issues are related to the interplay that different tasks can have at the RTOS level. The timing of the software tasks depend on the presence or absence of other tasks. A scheduling policy that could prevent timing variability in presence of dynamical changing task characteristics can be conceived but it will carry heavy overhead thus requiring powerful microprocessors even when they are not strictly needed. This is the standard trade-off between efficiency and reliability but it has more important business implications than usual. In fact, if software from different sources has to be integrated on a common hardware platform who will be responsible for the correct functioning of the final product?

Whoever will take on this responsibility would need a very strong methodology and an iron fist to make suppliers and partners comply with it. This may not be enough, in the sense that software characteristics are hard to pin down and with the best intentions of this world, one may not be able to guarantee functional AND timing behavior in the presence of foreign components. The ideal approach would be a tool that could map automatically the set of tasks onto the platform guaranteeing the correct functionality and timing with optimal resource utilization. This tool should take the design description at the pure functional level with performance and other constraints and the architecture of the platform and produce correct settings for the RTOS and optimized code. We are still far from this ideal. It is likely, then, that the responsibility for subsystem integration will still rest with the car manufacturers but the responsibility for integrating software components onto ECUs will be assigned to Tier 1 suppliers. In this case, the burden of Tier 1 suppliers will be increased at a possibly reduced premium because of the perceived reduction in added value. This is likely to be an unstable model and major attention should be devoted to find a common ground where both car makers and suppliers find their economic return.

If the strategy followed by car makers in AUTOSAR succeeds, then it is likely that a global restructuring of the industry will take place by creating an environment where Tier 1 plevels with small market share will find themselves in a difficult position unless they find a way of competing on a more leveled ground with the major stake holders. In this scenario, Tier 2 suppliers including IP providers may find themselves in a better position to entertain business relations directly with the car manufacturer. Tool providers will be in a more strategic position as providers of mapping tools that make the business model feasible. Hence, it is likely that a shift of recognized value will take place from Tier 1 suppliers towards tool providers and Tier 2 suppliers. The redistribution of wealth in the design chain may or may not be a positive outcome for the health of the industrial sector. If the discontinuities are sharp, then there may be a period of instability where much effort will be required to keep the products coming out with quality and reliability problems that may be larger than the ones observed lately. However, if it is well managed, then a natural shake-up with stronger plevels emerging will have a double positive: more quality in the products at lower cost. An additional benefit from a real plug-and-play environment will be the acceleration of the rate of innovation. Today, the automotive sector is considered conservative and the innovations in design methods and electronic components are slow to come. For example, if a well-oiled mechanism existed to migrate from one hardware platform to another, the “optimal” solutions would be selected instead of the ones that have been traditionally used. In this case, the Tier 2 market place will also be rationalized and the rate of innovation will likely be increased.

As a final consequence, the introduction of new functionalities will be a matter of algorithm and architecture rather than detailed software and hardware selection. The trend in electronics is clear: less customization, more standardization. This is indeed the reason why platform-based design and supporting tools [16] has appealed to a wide variety of electronic industry plevels. For a subsystem supplier, the choice will be richer in terms of platforms but it will not require heavy investment in IC design or RTOS development. For a car manufacturer, the granularity of its choices will be also richer because of interoperability. He will have the choice of selecting entire macro systems or components that could be integrated in a large automotive platform. The choice will be guided by cost, quality and product innovation.

The final goal of the strategy is rather clear. The way of getting there is not as clear and the road has many bumps and turns that are difficult to negotiate. A positive outcome will have to come from a process of deep business and technical cooperation among all plevels in the design chain as well as the research community. It is a unique opportunity and a great challenge.

VI. Acknowledgments

We wish to thank Alberto Ferrari and Pierpaolo Murrieri from PARADES; Gabriele Serra, Giacomo Gentile and Walter Nesci, from Magneti Marelli Powertrain (Bologna, I); Paolo Ferracin from CNH (Modena, I); Gilberto Burgio from Ford Forschungszentrum (Aachen, G) for the many discussions on the topic. This work has been partially supported by the E.C. Project IST-2001-33520 CC (Control and Computation). A. Balluchi and A. Sangiovanni–Vincentelli are members of the “HyCon” Network of Excellence, EC grant IST-511368.

VII. REFERENCES

[1] M. Antoniotti, A. Balluchi, L. Benvenuti, A. Ferrari, C. Pinello, A. L. Sangiovanni-Vincentelli, R. Flora, W. Nesci, C. Rossi, G. Serra, and M. Tabaro. A top-down constraints-driven design methodology for powertrain control system. In *Proc. GPC98, Global Powertrain Congress*, volume Emissions, Testing and Controls, pages 74–84, Detroit, Michigan, USA, October 1998.

[2] AUTOSAR. www.autosar.org.

[3] A. Balluchi, L. Benvenuti, M. D. Di Benedetto, C. Pinello, and A. L. Sangiovanni-Vincentelli. Automotive engine control and hybrid systems: Challenges and opportunities. *Proceedings of the IEEE*, 88, ”Special Issue on Hybrid Systems” (invited paper)(7):888–912, July 2000.

[4] A. Balluchi, L. Benvenuti, C. Lemma, P. Murrieri, and A. L. Sangiovanni-Vincentelli. Hybrid models of an automotive driveline. Tech. rep., PARADES, Rome, I, December 2004.

[5] A. Balluchi, L. Benvenuti, C. Lemma, A. L. Sangiovanni-Vincentelli, and G. Serra. Actual engaged gear identification: a hybrid observer approach. In *to be presented at 16th IFAC World Congress*, Prague (CZ), July 2005.

[6] A. Balluchi, M. D. Di Benedetto, A. Ferrari, G. Gaviani, G. Girasole, C. Grossi, W. Nesci, M. Pennese, and A. L. Sangiovanni-Vincentelli. Design of a motorcycle engine control unit using an integrated control-implementation approach. In *Proc. 1st IFAC Workshop on ”Advances in Automatic Control”*, pages 218–225, Salerno, Italy, April 2004.

[7] A. Balluchi, M. D. Di Benedetto, C. Pinello, C. Rossi, and A. L. Sangiovanni-Vincentelli. Hybrid control in automotive applications: the cut-off control. *Automatica*, 35, Special Issue on Hybrid Systems:519–535, March 1999.

[8] M. Baotic, M. Vasak, M. Morari, and N. Peric. Hybrid theory based optimal control of electronic throttle. In *Proc. of the IEEE American Control Conference, ACC 2003*, pages 5209–5214, Denver, Colorado, USA, June 2003.

[9] A. Bemporad, G. Bianchini, F. Brogi, and F. Barbagli. Passivity analysis and passification of discrete-time hybrid systems. 2005. Submitted.

[10] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino. A bounded-error approach to piecewise affine system identification. *IEEE Trans. Automatic Control*, 2004. Accepted for publication as a regular paper.

[11] A. Bemporad, M. Morari, and N. L. Ricker. *Model Predictive Control Toolbox for Matlab – User’s Guide*. The Mathworks, Inc., 2004. http://www.mathworks.com/access/helpdesk/help/toolbox/mpc/.

[12] J. Chen and R.J. Patton. *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Number 3 in Series on Asian Studies in Computer and Information Science. Kluwer International, 1999.

[13] H. Heinecke, K.-P. Schnelle, H. Fennel, J. Bortolazzi, L. Lundh, J. Leflour, J.-L. Mat/’e, K. Nishikawa, and T. Scharnhorst. Automotive open system architecture - an industry-wide initiative to manage the complexity of emerging automotive e/e-architectures. In *Proc. of Convergence 2004*, number 2004-21-0042, Detroit, MI, October 2004.

[14] H.O. List and P. Schoeggl. Objective evaluation of vehicle driveability. Technical Report 980204, SAE, 1998.

[15] R. M obus, M. Baotic, and M. Morari. Multi-object adaptive cruise control. In O. Maler and Eds. A. Pnueli, editors, *Hybrid Systems: Computation and Control, HSCC 2003*, volume 2623 of *Lecture Notes in Computer Science*, pages 359–374. Springer Verlag, 2003.

[16] A. Sangiovanni-Vincentelli. Defining platform-based design. *EEdesign*, February 2002. http://www.eedesign.com/.