Randomized Algorithms for Platform–based Design

Andrea Agostini*, Andrea Balluchi†, Antonio Bicchi*, Benedetto Piccoli‡,
Alberto Sangiovanni–Vincentelli†§ and Katarzyna Zadarnowska¶
* Centro Interdipartimentale di Ricerca "Enrico Piaggio". Università di Pisa, 56100 Pisa, Italy
† PARADES, Via S.Pantaleo, 66, 00186 Roma, Italy
‡ Istituto per le Applicazioni del Calcolo, Viale del Policlinico 137, 00161 Rome, Italy
§ EECS Dept., University of California, Berkeley, CA 94720, USA
¶ Institute of Engineering Cybernetics, Wroclaw University of Technology, 50-372 Wroclaw, Poland

***Abstract*— The design of automotive control systems is becoming increasingly complex as the level of performance required by car manufactures grows continuously and the constraints on cost and development time imposed by the market become tighter. A successful design, without costly and time consuming re-design cycles, can be achieved only by using an efficient design methodology that allows for component re-use and evaluation of platform requirements at the early stages of the design flow. In this paper, we illustrate a control-implementation design methodology for the development of embedded controllers by composition of algorithms picked up from libraries. Randomized algorithms and hybrid system theory are used to develop techniques for functional and architecture evaluations, which are implemented in a prototype tool.**

I. Introduction

Today, to meet increasing customers' expectations, car manufactures have to renew their products more frequently, reducing the design cycle and introducing innovation in each renewal, while keeping costs affordable. Most of the renewals require either significant extensions of the functionalities of automotive controllers or the introduction of ones. In fact, today 80% of car innovation is in electronic components. Hence, the pressure of competitiveness on automotive embedded systems design is extremely high. 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 re-use and evaluation of platform requirements at the early stages of the design flow.

In [1] and [2] the authors proposed a general methodology to bridge the gap between functional design and implementation of embedded control systems. The proposed design methodology is based on the principles of *platform-based design* described in [3]. A platform, in this context, is a layer of abstraction that hides the *unnecessary* details of the underlying implementation and yet carries enough information about the layers below to prevent design iterations. The application of the methodology presented in [2] to the design of an engine control unit (ECU) for motorcycles was first described in [4]. In the proposed approach, an integrated control-implementation design approach allows the designer to evaluate how closed–loop performance and robustness of a given control algorithm are affected by the implementation, which is represented in an abstract form. The importance of developing efficient design approaches for the top layers of the design flow is due to the fact that most of the critical design choices are taken in the early stages of the design flow and missteps in these stages produce costly and time consuming re-design cycles. Efficient re–use of components is essential to meet the tight constraints on development time and cost and should be fostered at all levels of the design flow. To support re–use at the functional layer, it is necessary to develop methodologies and tools that allow evaluation of "off the shelf" control algorithms, available from previous product developments, with respect to the customer requirements for the design at hand. Often, if direct re–use is not possible, requirements can be met with minor re–designs. The architecture of control algorithms should be conceived is such a way to maximize re–use, by choosing the correct granularity of partitioning for instance.

In this paper, we refine the design methodology proposed in [4] and present a prototype tool, denominated InterCIDe, that supports re–use in the design of automotive embedded systems. InterCIDe is written in Matlab and employs the modelling facilities of Simulink and Stateflow. It is conceived to support functional and architectural exploration for the development of embedded controllers by composition of algorithms picked up from libraries. InterCIDe allows the designer to formally describe the architecture of the control system and the requirements for each control algorithm, as well as to automatically evaluate different control algorithms and different HW/SW implementations, characterized in an abstract form. The procedures developed for functional and architecture evaluations are based on randomized techniques and hybrid system techniques. More precisely, performance criteria are tested for the control algorithms over parameters spaces both via randomized algorithms, i.e. letting the parameters vary according to a given probability distributions, and stochastic algorithms, i.e. letting the parameters evolve in time according to a random coefficient stochastic differential equation.

The collaboration with Magneti Marelli Powertrain

The work has been conducted with partial support by the EU Project IST-2001-33520 CC (Control and Computation) and the Multipartner "Marie Curie" Training Site "CTS" HPMT-CT-2001-00278.

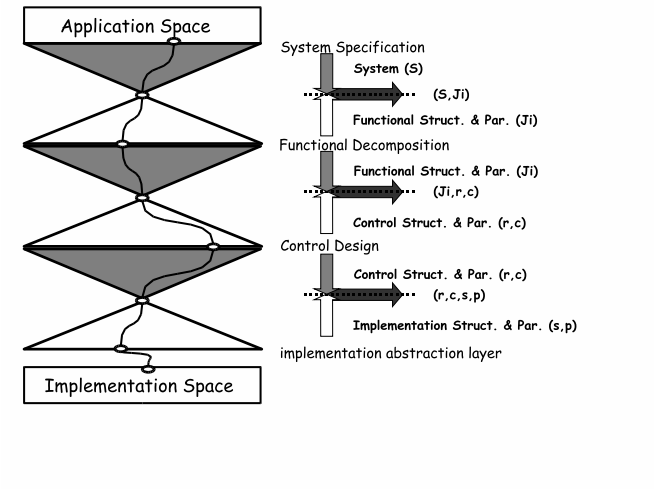

Fig. 1. Upper part of the platform stack.

(Bologna, Italy) was fecund for the definition of the tool specifications.

The rest of the paper is organized as follows. In Section II, we briefly recall the main concepts of platform–based design. In Section III, we present randomized and stochastic algorithms developed for functional and architecture explorations, giving an explicit example. Then, the tool InterCIDe and an application to ECU design are described in Section IV and in Section V, respectively. Some concluding remarks are given in Section VI.

II. Platform-based design methodology

The basic tenets of the Platform-based Design Methodology as exposed in [3] are:

- Regarding design as a "meet-in-the-middle process" where successive refinements of specifications meet with abstractions of potential implementations;
- The identification of precisely defined layers where the refinement and abstraction process take place.

These layers support designs built upon them allowing the designer to be freed from lower-level details but letting enough information transpire about lower levels of abstraction to allow design space exploration with a fairly accurate prediction of the properties of the final implementation. The information should be incorporated in appropriate parameters that annotate design choices at the present layer of abstraction. These layers of abstraction are called *platforms*. A platform is defined to be *an abstraction layer in the design flow that facilitates a number of possible refinements into a subsequent abstraction layer (platform) in the design flow*. The abstraction layer contains several possible design solutions, but limits the design exploration space. During the design process, at every step we choose a *platform instance* in the platform space. Every pair of platforms, the tools and methods that are used to map the upper layer of abstraction into the lower level one is a *platform stack*. Key to the application of the design principle is the careful

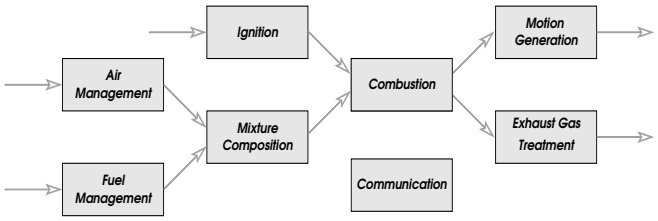

Fig. 2. Functional decomposition.

definition of the platform layers. Some layers are more important than others in the overall design trade-off space. In particular, the articulation point between functional design and implementation is a critical one for design quality and time.

In the platform-based design paradigm, the effects of the actual implementation is represented by an abstract model characterized by idealized parameters. Each choice of these parameters identifies an implementation platform. In this view, control design is a platform mapping with as many implementation details as exposed by the implementation platform.

A schematic view of the upper part of the design flow for ECUs in automotive applications is shown in Fig. 1. The design process includes: formalization of system specifications, functional decomposition and deployment, controller algorithm design and modeling of abstractions of potential implementations.

A. From System Specification to Functional Decomposition

To tackle complexity, the system is decomposed into a number of interacting simpler sub-systems, called *functions*. The decomposition is based on the understanding of the physical process of interest. This first stage 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). By the decomposition process, the objectives and constraints that define the system specification are distributed among the components, so that the composition of the behaviors of the components, made feasible and possibly optimal with their own constraints and cost functions, is guaranteed to meet the constraints and the objectives of the overall controlled system.

1) Functional platform: In our methodology, this design step represents a first refinement of the system specifications into a platform abstraction capturing a structure of the implementation. For engine control, we model the platform at this layer with eight main functions, as described in Fig. 2 (see [1]). Since, in general, it is difficult to decompose the system into independent parts, the determination of the local objectives and constraints has to be the result of a careful trade-off between the desire of optimality at the global level and the ease of design for each component.

2) Functional refinement: The mapping of system specification to the platform model (previously described) defines

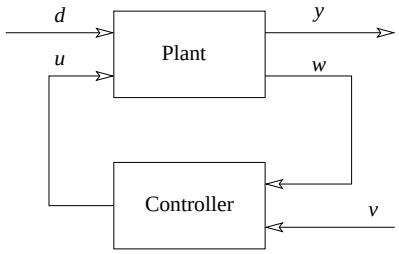

Fig. 3. General scheme for the functional design.

the behavior for each functional component. This refinement can be cast into the classical robust control representation depicted in Fig. 3, where the controller represents the behavior of the function under design and the plant represents the remaining part of the system (plant plus other functions). Furthermore, d models measurable and unmeasurable disturbances to be rejected, v denotes reference signals and commands, w stands for feedforward and feedback signals, u represents the control inputs and y denotes the system outputs.

The behavior of the function is represented by a functional specification defined in terms of a number N of inequalities of the type:

$$\bar{J}_i\left(\mathcal{J}_i\{y\}|_{x\in\mathcal{X}_i}\right) \leq 0 \qquad \text{for } i = 1 \ldots N \tag{1}$$

where:

- the system of inequalities may be related to different operating modes of the system and – possibly – different requirements for each operating mode;
- $\mathcal{J}_i : \mathcal{Y}_i \rightarrow \mathbb{R}$ is a functional measuring the performance of the controlled system on a particular evolution $x \in \mathcal{X}_i$ and the operator $\bar{J}_i(\cdot)$ collects the overall performances in $\mathcal{X}_i$;
- x denotes the state of the plant, and $\mathcal{X}_i$ is the family of system evolutions of interest[1];
- y denotes the system outputs on which the functional is applied and $\mathcal{Y}_i$ is the family of output evolutions.

B. From Functional Decomposition to Control Strategies

As represented in Fig. 1, the next step in the design flow consists of a refinement of the functional decomposition obtained in the previous step into a set of control strategies using given control platforms.

1) Control platforms: On the basis of a model of the plant interacting with the functional component, a set of candidate control strategies are devised for each function. Different control strategies are conceived to allow for exploration of different solutions. These strategies correspond to different choices of physical variables in d, u, w and v, and different algorithms. Therefore, the platforms at the control strategies layer are described by

- a number R of different controller structures;
- a set X_C^r of *control parameters* for each controller structure $r \in \{1, \ldots, R\}$.

2) Control refinement: A particular control strategy, resulting from the mapping of a functional solution into a control platform, is identified by selecting a controller structure and an admissible value for the control parameters. Let $\tilde{y}$ and $\tilde{x}$ respectively denote the representation in the given model of the physical variables y and x. The functional specification (1) is guaranteed for a control structure r, control parameters c, and a given plant model if

$$\begin{aligned}\bar{J}_i\left(\mathcal{J}_i\{y\}|_{x\in\mathcal{X}_i}\right) &\leq \bar{J}_i\left(\mathcal{J}_i\{\tilde{y}(r,c)\}|_{\tilde{x}\in\mathcal{X}_i}\right)\\ &\equiv J_i(r,c) \leq 0 \qquad \text{for } i = 1 \ldots N.\end{aligned} \tag{2}$$

Note that, while $\mathcal{J}_i\{\cdot\}$ is a functional that is applied to the system outputs, $J_i : \{1, \ldots, R\} \times X_C^r \rightarrow \mathbb{R}$ is a function of the controller structures and control parameters. To guarantee system specification (1), the model that produces $\tilde{y}(r, c)$ has be conservative w.r.t. functionals $\mathcal{J}_i\{\cdot\}$.

C. From Control Strategies to Implementation Abstract Model

Finally, referring again to Fig. 1, we describe the third step of the methodology in which control strategies are refined in an implementation abstract model.

1) Implementation platforms: The essential issue for representing implementation platforms in an abstract way is to determine the effect of implementation platforms on the controlled system performances. Accuracy of measurements and actuations, and how to represent the fact that computation and communication take time and may be affected by errors are important in this respect. The main effects of a particular implementation on the behavior of the controlled system must be carefully classified and characterized.

They can be represented in terms of perturbations on the controller input/output channels, as illustrated in Fig. 4. Disturbances n_u, n_w, n_r and blocks Δ_u, Δ_w, Δ_r represent, respectively, value and time domains perturbations due to the implementation and acting on the control inputs u, feedback outputs w and reference signals v. Depending on the selected platform, these perturbations can be represented by different models and characterized by abstract parameters p. A set of implementation platforms with the corresponding exported parameters is defined by:

- a number S of different platform structures;
- a set of parameters X_P^s for each platform structure $s \in \{1, \ldots, S\}$;
- a set of platform constraints[2]

$$J_v(s, p) \leq 0, \qquad \text{for } v = 1 \ldots V. \tag{3}$$

For a given platform structure $s \in \{1, \ldots, S\}$, elements $p \in X_P^s$ are referred to as the *platform parameters*.

[1]Which may depend on uncertain and time-varying parameters, as well as initial and final conditions.

[2]Typically, the schedulability and latency constraints are added.

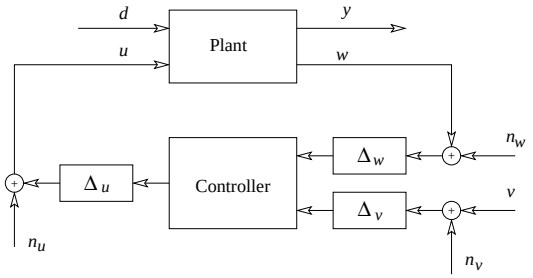

Fig. 4. Abstract representation of the effects of implementation non–idealities.

2) Implementation abstract model refinement: In the control parameters and platform parameters product space, feasible mappings are given by the set

$$\begin{aligned} U \quad = \quad & \{(r,c,s,p) \mid r \in \{1,\dots R\},\ c \in X_C^r, \\ & s \in \{1,\dots S\},\ p \in X_P^s,\ \text{such that} \\ & J_i(r,c,s,p) \leq 0,\ \text{for } i = 1 \dots N+V\} \end{aligned} \quad (4)$$

where J_i include both conservative expressions for (2), including the effects of the implementation platform modeled by (s,r), and the platform constraints (3).

The best implementation platform, among those described by parameters inside the set U in (4), can be selected by introducing a suitable objective function, representing an estimation of the implementation cost to be minimized. It is often the case that the cost model depends only on a subset of the parameters. The parameters belonging to the orthogonal space can be abstracted away. A very powerful method for the abstraction of non–relevant parameters is the use of formulas with existence and universal quantifiers.

As an example, consider a PI controller with gains K_P and K_I designed to control the speed of a crankshaft, with nominal inertia J and inertial uncertainty δ_J. Parameters K_P, K_I, J, and δ_J can be abstracted using the following formula:

$$\forall J\ \exists\ K_P, K_I\ \forall\ \delta_J \quad (\hat{c}, J, K_P, K_I, \delta_J, \hat{p}) \in U\ , \quad (5)$$

where $\hat{c}$ and $\hat{p}$ are other control and platform parameters. By applying quantifier elimination to (5), the exploration is reduced to the subspace $(\hat{c}, \hat{p})$, where the cost model $H(\hat{c}, \hat{p})$ is defined. In [2], the use of *flexibility*

functions as cost models was proposed. The underlining idea is to guide the exploration towards parameters that can be easily achieved by platforms at lower levels of abstraction. In this way, the risk of expensive design cycles that span several platforms is minimized and a better platform choice is offered while approaching the implementation level.

III. Randomized algorithms

The determination of feasible maps (4), in the control parameters and platform parameters spaces, happens to be an hard task, which in many real cases is not affordable by deterministic robust control methods. The high computational complexity is a common phenomenon in the determination of feasible regions in robust control, see [5], that motivated the use of Randomized Algorithms as a general tool [6], [7]. In our case, this issue is even more delicate because of the hybrid and nonlinear nature of the involved control systems. For a fixed controller structure $\bar{r}$ and platform structure $\bar{s}$, the problem boils down to a performance indices verification of the type:

$$J_i(\bar{r}, c, \bar{s}, p) \leq 0, \qquad i = 1, \dots, N+V,$$

for c, p varying in the appropriate spaces $X_C^{\bar{r}}, X_P^{\bar{s}}$.
In Randomized Algorithms the exploration of the feasible region in the parameters space is operated choosing the sampling according to assigned probability distribution functions $\tilde{c} : \Omega \mapsto X_C^{\bar{r}}$, $\tilde{p} : \Omega \mapsto X_P^{\bar{s}}$, where Ω is the event space. It is natural to assume that such functions are of the known types as Gaussian, Poisson or other. This method permits to conduct an efficient exploration, while keeping the computational costs at an acceptable level.
However, the corresponding probability distribution functions (briefly pdfs) of the indices J_i happen to be quite scattered reflecting the choice of parameters which are fixed, once sampled, and do not vary as the control system evolves in time. For example, in the idle control problem the level of the battery is a parameter which is expected to have a prescribed mean value. The use of this resource by many of the car components leads to a stochastic behavior whose associated distribution is expected to be normal. The sampling method of a Randomized Algorithm chooses a value in the parameter space which is kept fixed as the system evolves: such choice represents a battery having a regime level which is not the expected one (probably due to a battery production failure). The resulting pdfs of the performance indices may thus be not satisfactory.
Moreover, even if the battery is perfectly performing, the level at a certain instant of time may be different from the average and this behavior is better represented by a stochastic evolution. Hence, beside the implementation of Random Algorithms, we propose the use of stochastic processes to model parameters evolution. From mathematical point of view this amounts to simulate the solutions to a random coefficients stochastic differential equation:

$$dp = g(t,p,\omega) + \tau(t,p,\omega)dW$$

where $\omega \in \Omega$ and, as usual, W is a standard Brownian motion (and a similar equation for c). The necessity of modelling the parameters behavior with random coefficients comes from the need of keeping the parameters in a given bounded set, e.g. $X_P^{\bar{s}}$, while maintaining the typical feature of a noise signal, see the example below and [8].

Example: FILM control system.
The control algorithm we consider is in charge of compensating the evaporation of the fuel puddle, which affects the amount of fuel entering the cylinder during intake. The fuel puddle is produced by the condensation of the injected fuel in the intake runner, on the wall in front of the injector. The aim is to apply both randomized algorithms and stochastic processes to determine the sensitivity of the given control system with respect to the parameters changes as a first step

for determination of feasible regions.

The fuel puddle phenomena is usually modelled by the mean–value model

$$\begin{cases} \dot{m}_v = \frac{1}{\tau} m_1 + (1-\chi) q \\ \dot{m}_1 = -\frac{1}{\tau} m_1 + \chi q \end{cases} \tag{6}$$

where m_v is the mass of vaporized fuel, m_1 is the mass of the fuel puddle and q is the injected fuel flow rate. The evaporation time constant τ and the condensation coefficient $\chi \in (0,1)$ depend on the engine temperature T_e and such dependence is compensated by the FILM algorithm. The uncertainties on the expressions $\tau(T_e)$ and $\chi(T_e)$ are represented in the plant model respectively by additive disturbances δ_τ^p and δ_χ^p. The computation of $\tau(T_e)$ and $\chi(T_e)$ in the implementation of the FILM algorithm is affected by errors due to the implementation platform bounded precision, modelled as additive disturbances δ_τ^c and δ_χ^c.

We set $J(\bar{r}, c, \bar{s}, p) = |err_{mbenzT}|$, the error in the mass of vaporized fuel, and want to bound it by $\varepsilon = 0.2$. In order to check the sensitivity of the system (6) the following indices are determined:

1) Dependence (in percentage) of $J(\bar{r}, c, \bar{s}, p)$ on a given parameter $\bar{p}$

$$d = \frac{100 \max_{\bar{p} \in [p_l, p_r]} \{\max_{\sigma \in \Sigma} J(\bar{r}, c, \bar{s}, p) - \min_{\sigma \in \Sigma} J(\bar{r}, c, \bar{s}, p)\}}{E[J(\bar{r}, c, \bar{s}, p)]}, \tag{7}$$

where $[p_l, p_r]$ is the range of $\bar{p}$ and Σ is the set of samples. (The same definition for given $\bar{c}$.)

2) Probability that fulfilment of the performance criterion depends on a given parameter

$$P = \frac{\sum_{\sigma \in \Sigma} 4y(1-y)}{\#\Sigma} \tag{8}$$

where $y = P(\max_{\sigma \in \Sigma} J(\bar{r}, c, \bar{s}, p) \leq \varepsilon)$ and $\#$ denotes the cardinality.

We perform two stages: on the *first stage* we let all the parameters be constant except the chosen one, which evolves inside the range $[p_l, p_r]$ as

$$p(t, \omega) = G(W(t))$$

where $G(t,x) = \frac{p_r - p_l}{\pi} \arctan x + \frac{p_r - p_l}{2}$. This corresponds to the solution to the stochastic differential equation

$$\begin{cases} dp = g(t,p,\omega)dt + \tau(t,p,\omega)dW(t) \\ p(0,\omega) = 0, \end{cases}$$

where (see [8], [9] for details)

$$g(t,p,\omega) = \frac{(p_l - p_r) W(t)}{\pi(1+W^2(t))^2}, \qquad \tau(t,p,\omega) = \frac{p_r - p_l}{\pi(1+W^2(t))}.$$

We then compute d and P and, if $d \leq 50$ and $P = 0$, we pass to the next stage. At the *second stage* the other parameters are assumed to be random variables chosen according to $G(N(0,t))$, where $N(0,t)$ denotes normal distribution. The maxima of d and P are computed over 40 samples for the other parameters.

For both stages the range $[p_r, p_u]$ is divided into 10 small intervals, $p(0,\omega)$ is set to be center of the small interval and 50 samples are produced for the time horizon $T = 20$.

TABLE I

SENSITIVITY OF FILM CONTROL SYSTEM.

parameter	stage I	stage II
$m_c \in [-1;1]$	$d = 40.53$ $P = 0$	$d = 42.08$ $P = 0.84$
$eff_{pla} \in [-0.1;0.1]$	$d = 207.23$ $P = 0.88$	-
$eff_{asp} \in [-0.1;0.1]$	$d = 276.81$ $P = 0.80$	-
$\delta_\tau^p \in [-0.05;0.05]$	$d = 6.26$ $P = 0$	$d = 6.53$ $P = 0$
$\delta_\tau^c \in [-0.05;0.05]$	$d = 9.03$ $P = 0$	$d = 5.32$ $P = 0$
$\delta_\chi^p \in [-0.05;0.05]$	$d = 3.99$ $P = 0$	$d = 9.89$ $P = 0$
$\delta_\chi^c \in [-0.05;0.05]$	$d = 11.90$ $P = 0$	$d = 10.80$ $P = 0$

In order to execute the first stage, the constant values of parameters are: $\delta_\tau^c = 0.025$, $\delta_\tau^p = 0.025$, $\delta_\chi^c = 0.025$, $\delta_\chi^p = 0.025$, $m_c = 0.05$, $eff_{pla} = 0.05$, $eff_{asp} = 0.05$. Table (I) collects numerical results: it is worth observing that eff_{pla} and eff_{asp} influence the system the most, in fact the fulfilment of the performance criterion strongly depends on these parameters. On the contrary, $J(\bar{r}, c, \bar{s}, p)$ depends only slightly on δ_τ^c, δ_τ^p, δ_χ^c and δ_χ^p.

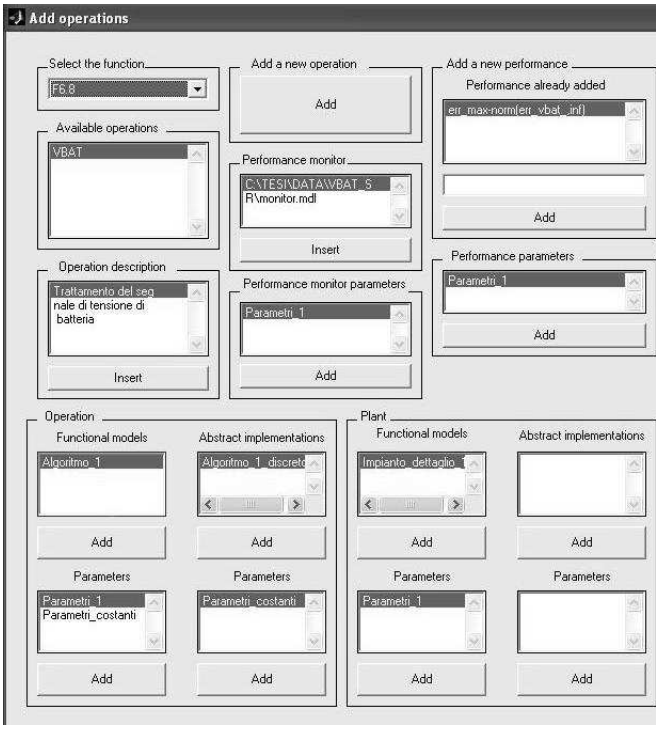

Fig. 5. Main window for the description of control algorithms and plants.

IV. INTERCIDE

In this section, we present the tool InterCIDe that has been developed to support the integrated control–implementation design methodology presented in Section II and implements the randomized algorithms described in Section III. InterCIDe is developed in Matlab, uses the modeling facilities of

Simulink and Stateflow, and employs the Matlab Statistic Toolbox for the implementation of statistic functions. InterCIDe allows the designer to compose the set of control algorithms in each functional component by extracting them from libraries of existing solutions. Figure 5 reports the main window for the introduction and selection of control algorithms.

According to the methodology described in Section II, each candidate control algorithm is parametrized, in terms of control parameters and implementation parameters. The exploration between the candidate control algorithms is performed in the control parameters and implementation parameters space. Efficiency of the parameter space exploration is fundamental for the successful of the approach. Randomized algorithms allow to achieve accurate exploration within reasonable computation times. For each sample of control parameters and implementation parameters, the tool evaluates whether or not the assigned closed–loop

specifications are met. InterCIDe performs a random screening of the parameter space, labelling each sample, either *good* or *bad*, according to fulfillment of the specification. In a first stage, the control algorithm is functionally tested, screening the control parameter space. If the tool succeeds in finding a big enough region containing *good* samples, then, in a second stage, implementation details are modelled in an abstract way and the screening proceeds including the implementation parameter space.

Next, quantifier elimination is applied to project the results in the subspace where the estimation model for the implementation cost is defined. Quantifier elimination has been implemented using a griding method. By means of quantifier elimination, the samples are projected in the cost model parameter subspace. Then, the convex polyhedra $\mathcal{P}_{good}$ and $\mathcal{P}_{bad}$ are produced by computing the convex hull of the subsets of *good* and *bad* samples, respectively. The subset of parameter values for which the specification is assumed to hold is given by the (non–convex, in general) polyhedron $\mathcal{P}_{good} \setminus \mathcal{P}_{bad}$. Subsequently, the cost parameter subspaces of all control algorithms to be implemented in the electronic control unit are composed and the set $\mathcal{P}_{ecu}$ is defined by composing the sets $\mathcal{P}_{good} \setminus \mathcal{P}_{bad}$ obtained for each control algorithm. Hence, the point corresponding to the minimum cost, according to the given cost model, in the set $\mathcal{P}_{ecu}$ is obtained by an optimization algorithm. Finally, optimal values – maximizing robustness – for the remaining parameters of each control algorithm abstracted away by the quantifier elimination are selected.

V. Application

The description of an entire chain of control algorithms for fuel injection in a motorcycle application had been introduced in InterCIDe. For each control algorithm, desired closed–loop specification, and functional and implementation models had been described using the tool according to design methodology presented in Section II. Using the randomized algorithms implemented in InterCIDe, control and implementation parameter screening has been performed for each control algorithm. Then, quantifier elimination is applied to project the results of the exploration in the subspace of sampling frequency F_i and latency L_i parameters. The sampling frequency and latency subspaces of all control algorithms are composed in the space $(F_i \times L_i)^m$ and constraints on maximal latency and CPU utilization are introduced. The latter is defined as follows:

$$J_v(W_i, F_i, U_{cpu}) = \sum_{i=1}^{m} W_i F_i - U_{cpu} \leq 0$$

where W_i and F_i are, respectively, the worst case execution time and the execution frequency of the i–th algorithm, and U_{cpu} is the maximal CPU utilization.

It was obtained that the feasible set in the $(F_i \times L_i)^m$ was not empty. Then, the CPU utilization $J_v(W_i, F_i, U_{cpu})$ was used as cost model to be minimized.

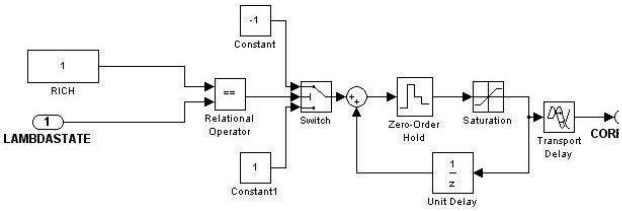

Fig. 6. GTIT control algorithm.

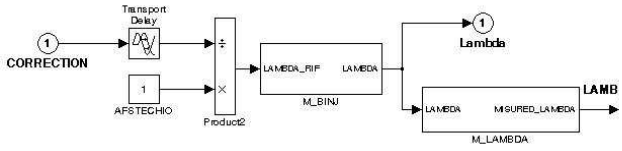

Fig. 7. GTIT plant model.

As an example, we report the result obtained for the GTIT algorithm, which is designed for air–to–fuel ratio control. The GTIT algorithm produces desired air–to–fuel ratio corrections, on the basis of the feedback from the ON/OFF lambda sensor. The specification is to keep the mean–value of the air–to–fuel ratio close to the stoichiometric value. The GTIT algorithm is depicted in Figure 6. The plant model, described in Figure 7, includes closed–loop models for the subsystems interacting with the GTIT algorithm, namely the TIT and BINJ subsystems.

The control parameters are: the sampling frequency, the latency, the

upper and lower saturation limits. Each parameter is represented as follows:

$$p_i = (1 + \delta_i) P_i 10^{Q_i}$$

where P_i and Q_i are constant and δ_i is random. Figure 8 reports the set $\mathcal{P}_{good} \setminus \mathcal{P}_{bad}$ for the GTIT algorithm. This set is defined in the sampling frequency and latency subspace and is obtained by abstracting away the upper and lower saturation limits, using quantifier elimination. The GTIT set $\mathcal{P}_{good} \setminus \mathcal{P}_{bad}$ in Figure 8, composed with the corresponding

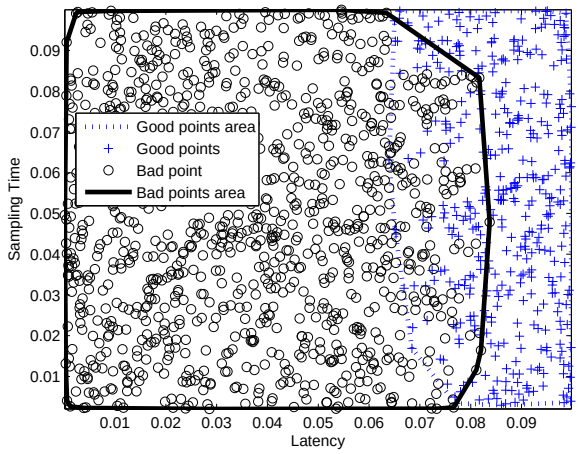

Fig. 8. Implementation parameters projection.

ones obtained for the other control algorithms, defines set $\mathcal{P}_{ecu}$ on which the optimal solution that minimizes CPU utilization is searched for.

VI. Conclusions

A control-implementation design methodology for the development of embedded controllers in the automotive industry has been presented. The proposed methodology allows the designers to reduce the development cost by maximizing re–use of algorithms available from previous designs. The methodology is supported by the prototype tool InterCIDe that allows automatic validation of functional and architecture solutions. The validation techniques implemented in InterCIDe are based on randomized and stochastic algorithms and hybrid system theory.

References

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

[2] A. Balluchi, L. Berardi, M. D. Di Benedetto, A. Ferrari, G. Girasole, and A. L. Sangiovanni-Vincentelli, "Integrated control–implementation design," in *Proc. 41st IEEE Conference on Decision and Control*, vol. 2, Las Vegas, NV, USA, December 2002, pp. 1337–1342.

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

[4] 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"*, Salerno, Italy, April 2004, pp. 218–225.

[5] G. Calafiore and F. D. (editors), *Probabilistic and randomized methods for design under uncertainty*. New York: Springer, 2005.

[6] R. Tempo, G. Calafiore, and F. Dabbene, *Randomized Algorithms for Analysis and Control of Uncertain Systems*, ser. Communications and Control Engineering. Springer, London, 2005.

[7] M. Vidyasagar, "Statistical learning theory and randomized algorithms for control," *IEEE Contr. Systems Magazine*, vol. 18, pp. 69–85, no. 6 1998.

[8] B. Piccoli and K. Zadarnowska, "Stochastic algorithms for robustness of control performances," in *Proc. 44st IEEE Conference on Decision and Control, submitted to this same session*, Sevilla, Spain, December 2005.

[9] J. Yong and X. Zhou, *Stochastic Controls. Hamiltonian Systems and HJB Equations*. New York: Springer, 1999.