Hindawi Publishing Corporation
International Journal of Vehicular Technology
Volume 2012, Article ID 103530, 14 pages
doi:10.1155/2012/103530

Research Article

Modeling and Deployment of Model-Based Decentralized Embedded Diagnosis inside Vehicles: Application to Smart Distance Keeping Function

Othman Nasri,[1] Hassan Shraim,[1] Phillippe Dague,[1] Olivier Heron,[2] and Michael Cartron[2]

[1] *LRI, University Paris-Sud 11, CNRS & INRIA Saclay Île-de-France, Bât 490, 91405 Orsay Cedex, France*
[2] *CEA LIST, Saclay, 91191 Gif-sur-Yvette Cedex, France*

Correspondence should be addressed to Hassan Shraim, hassan.shraim@gmail.com

Received 9 October 2011; Accepted 9 January 2012

Academic Editor: David Fernández Llorca

The deployment of a fault diagnosis strategy in the Smart Distance Keeping (SDK) system with a decentralized architecture is presented. The SDK system is an advanced Adaptive Cruise Control (ACC) system implemented in a Renault-Volvo Trucks vehicle to increase safety by overcoming some ACC limitations. One of the main differences between this new system and the classical ACC is the choice of the safe distance. This latter is the distance between the vehicle equipped with the ACC or the SDK system and the obstacle-in-front (which may be another vehicle). It is supposed fixed in the case of the ACC, while variable in the case of the SDK. The variation of this distance depends essentially on the relative velocity between the vehicle and the obstacle-in-front. The main goal of this work is to analyze measurements, issued from the SDK elements, in order to detect, to localize, and to identify some faults that may occur. Our main contribution is the proposition of a decentralized approach permitting to carry out an on-line diagnosis without computing the global model and to achieve most of the work locally avoiding huge extra diagnostic information traffic between components. After a detailed description of the SDK system, this paper explains the model-based decentralized solution and its application to the embedded diagnosis of the SDK system inside Renault-Volvo Truck with five control units connected via a CAN-bus using "Hardware in the Loop" (HIL) technique. We also discuss the constraints that must be fulfilled.

1. Introduction

In order to respond to the increasing demands of safety and driving comfort, more and more electronic functions are embedded in the vehicles such as engine control (to optimize fuel economy and to reduce pollution), Antilock Braking System (ABS), and Electronic Stability Program (ESP), Each global safety or comfort system contains one or more functions which may be distributed on several Electronic Control Unit ECUs. Most of these functions are modular and respecting some norms (such as AUTOSAR). They exchange information (e.g., vehicle speed) with other functions via communication interfaces. That means that if the system is not equipped with a certain diagnosis strategy, any fault generated from a function may influence all the functions are related to it. This fact highlights the problem of fault propagation and the need of an on-board (i.e., on running car) fault diagnosis in the vehicle.

An increasing number of vehicles are being equipped with adaptive cruise control (ACC). The ACC adjusts the brake and/or throttle, within limited ranges, to maintain a constant headway from any other vehicle that intrudes upon the path of the driver's vehicle. While ACC provides a potential safety benefit in helping drivers to maintain a constant speed and headway Davis [2], as with other types of automation, there is the potential for misuse and disuse Parasuraman and Riley [3]. It provides assistance to the driver in the task of longitudinal control of his vehicle during motorway driving. For ACC to be effective, drivers need to understand the capabilities of the technology, which include braking and sensor limitations. Based on this understanding, they must be able to intervene when a given situation exceeds ACC capabilities. However, drivers have difficulties in understanding how ACC functions Stanton and Marsden [4]. As a result, they tend to rely on the system inappropriately. For instance, Nilsson [5] showed that drivers

failed to intervene when approaching a stopped queue of vehicles because they believed that the ACC could effectively respond to the situation. Stanton et al. [6] introduced an unexpected acceleration into the ACC system during routine driving conditions, which resulted in a collision 33 percent of the time. Whether or not drivers can respond effectively when automation fails depends on their understanding of the type of failure that occurs and the context in which it occurs Lee et al. [7].

To ensure safe and effective use, ACC limits of operation should be identifiable and interpretable Goodrich et al. [8]. One approach to help drivers to detect and to respond to these limits is to match the limits of the ACC algorithm to the natural boundaries drivers use to switch between car-following and active braking behaviors, as defined by environmental cues (e.g., time headway (THW) and time to collision (TTC)). In order to avoid several problems that may be produced because of the misuse or the disuse of the ACC, the Smart Distance Keeping (SDK) system is proposed. (The SDK system is an advanced Adaptive Cruise Control (ACC) system implemented in a Renault-Volvo Trucks vehicle to increase safety by overcoming some ACC limitations.) This system must be understood as a function to enhance the driver's capability to manage his longitudinal environment and is dedicated to a use on highways or expressways (straight line, low curvatures, oneway roads). The SDK is based on the immediate front environment sensing on one hand, and on the automated management of the truck longitudinal actuators (brakes, engine, gearbox) on the other hand, all this being monitored and controllable at any time by the driver through the in-cabin human machine interface and the conventional driving commands (pedals, switches). This system is composed of many subsystems (micro controllers, cables, CAN or FlexRay bus, sensors, actuators, etc.) coming from different suppliers. That is why its diagnostic is a challenging task.

In this work, we will focus on the modeling and the fault diagnosis of the SDK system. Faults study is limited to those that may be produced on sensors measurements due to their direct influence on the SDK system decision. So, they should be always checked to ensure that they are within their expected operating range. Simple checks on the recent rate of change or variance of the output can also be incorporated. Faults which cause the sensors to have an offset or altered gain will affect the control system but may not be detected by this first level approach. The traditional approach to sensor fault checking is to include hardware redundancy for sensors. If two sensors measure the same quantity disagree, there is likely to be a fault in one of them, and if three or more measurements are available, the fault is likely to be in the sensor which disagrees most. But due to the high cost of providing direct hardware redundancy for sensors, the analytic redundancy techniques were proposed. Conceptually, this equates to creating virtual sensors from other available measurements, to compare with the one being monitored. Analytic redundancy is used in available passenger car control systems.

Considering the size and the complexity of the SDK system, a centralized on-board diagnosis is not adequate because it requires the establishment of a global model of the system and too much communication and memory resources and prevents to act immediately each time a diagnosis could be found at local level. To detect and isolate possible faults in the SDK system and manage its architectural complexity, we have chosen to apply the model-based fault diagnosis approach (FDI) Patten et al. [9] in a decentralized manner. A local diagnoser is associated with each component of the SDK system based on a modular modeling of the plant elements. All local diagnosis decisions are transmitted via CAN-bus and merged by a dedicated supervisor in order to obtain a global decision and carry out any recovery action. A strategy for applying this merging operation was developed in order to be efficient.

The paper is organized as follows. In Section 2, we present a general description and modeling of the SDK controller with its environment system. The modeling in this work includes a simplified mathematical model of the wheels and the engine. Then, in Section 3, decentralized algorithm and strategy for the detection and the isolation of sensors faults that may affect the overall system are developed. Section 4 explains how to deploy this approach in order to achieve on-line diagnosis of the SDK system. Then, we evaluate the proposed diagnosis approach in Section 5. Finally, we conclude with a discussion on the related work.

2. System Description and Modeling

2.1. Smart Distance Keeping System. Smart Distance Keeping (SDK) or "enhanced Adaptive Cruise Control (ACC)" is a system which automatically controls the vehicle's longitudinal velocity, by acting on the engine, gearbox, retarder, and braking system. This requires the vehicle to be equipped with a radar system connected to a dedicated control unit as shown in Figure 1.

The global SDK system may be decomposed into two main parts, the SDK controller and the SDK physical system displayed in Figure 2.

The main functions of the SDK controller (the block "SDK Function") are to

(i) receive the distance between the truck and the object-in-front,

(ii) find the deceleration (acceleration) needed to realize the correct functioning of the SDK system (maintaining a minimal safety distance with the vehicle-in-front),

(iii) use a control algorithm for acting through engine, braking system, and so forth in order to adjust the velocity of the truck.

A realistic representation of the SDK controller is given by the Input Output Symbolic Transition System (IOSTS) model as shown in Figure 3. This model describes exhaustively the relevant driving situations where the distance control intervenes. Apart from the "Refresh (R)" mode (initial state), three categories of "basic" modes can be distinguished: the "Cruise Control (C)" mode, the "Approch (A)" mode, and the "Follow (F)" mode. The variables of this

Figure 1: Radar installed on the Renault Magnum truck.

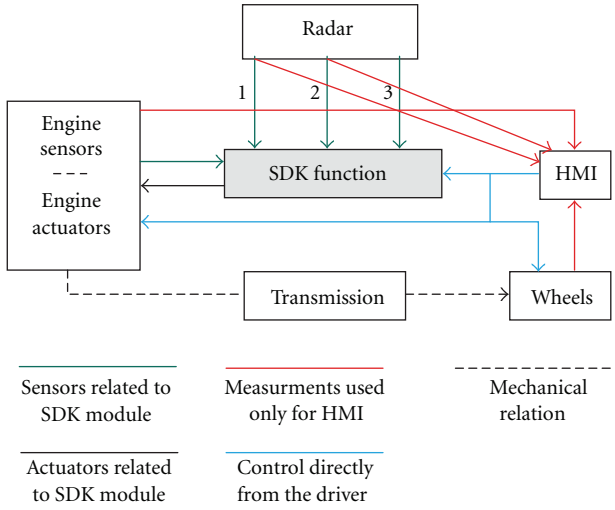

Figure 2: Environment of the SDK system.

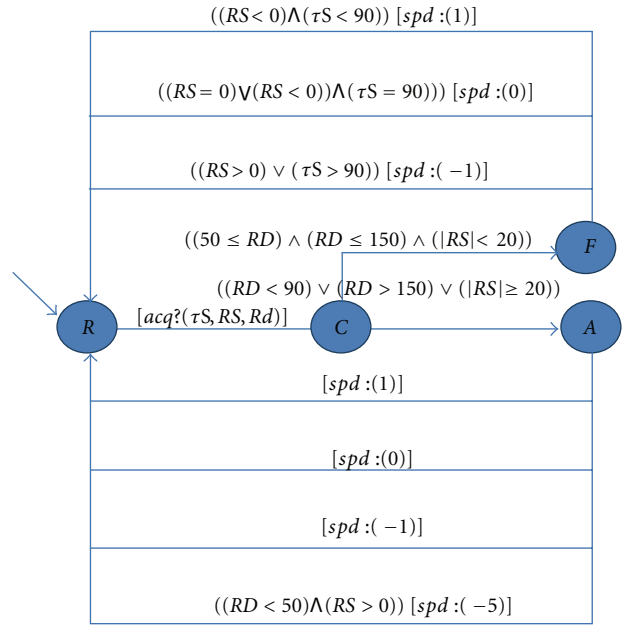

Figure 3: IOSTS system of the SDK controller.

IOSTS system are TS(= TruckSpeed), RS(= RelativeSpeed), and RD(= RelativeDistance), while the communication channels are acq(= *acquisition*) and spd(= *speed*) where $acq?(TS, RS, RD)$ and $spd!(-5, -1, 0, 1)$.

The decision of the SDK controller depends essentially on the data issued from some sensors: truck velocity sensor, wheels angular velocity sensors, radar, and engine sensors.

2.2. The Radar. The SDK needs to be informed about the object-in-front presence, and about its relative position and velocity. Within this work, the sensor is a 3-beam Doppler effect ARS100 Radar. This radar monitors the traffic in front of the vehicle using three stationary independent millimeter waves.

Moving and stationary objects are detected and their distance and relative velocity are measured and processed sixteen times per second.

Due to its physical nature, the radar sensor is offering excellent performance characteristics even in adverse weather conditions.

Since the data issued from the radar depend on the external object, then in order to realize any simulation, several scenarios should be prepared for the movements of the SDK vehicle and of the object-in-front. In this work, we suppose that the distance and the relative velocity between the SDK vehicle and the vehicle-in-front are the inputs to the system (depending on the scenarios that we are choosing).

2.3. The Wheels. The linear truck velocity, which is one of the important inputs to the SDK controller, is calculated based on the angular velocities of the six wheels.

The wheel rotational dynamics is given in (1) by applying Newton's Law:

$$I_{w_i}\dot{w}_i = R_i F_{X_i} + \text{Torque}(m_i) - \text{Torque}(b_i), \qquad (1)$$

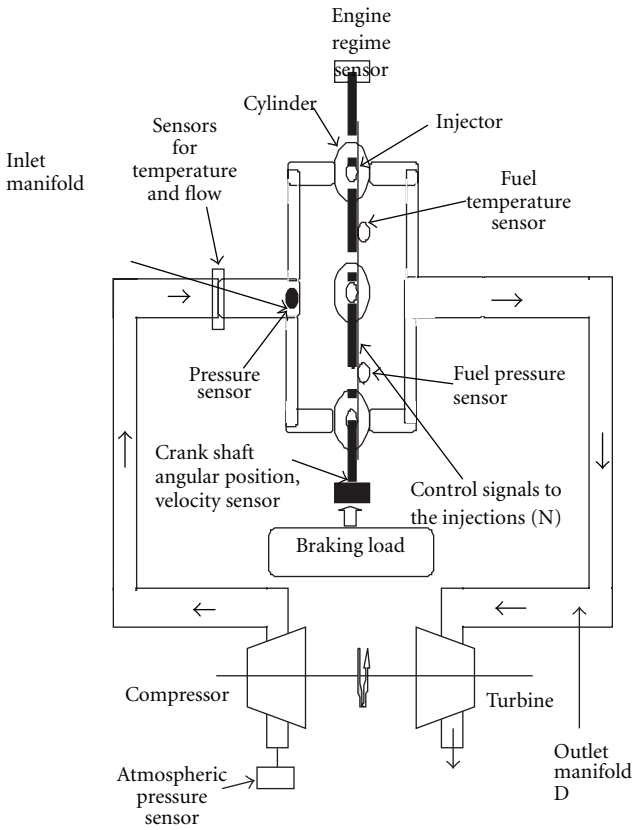

Figure 4: Engine Architecture Peysson et al. [1].

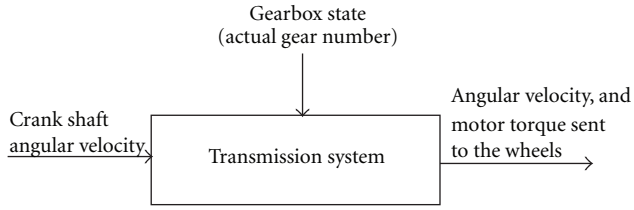

Figure 5: Transmission block.

where I_{w_i} is the moment of inertia of the wheel number i, R_i its effective radius, w_i is the angular velocity of the wheel, F_{X_i} is the friction force, Torque(m_i) is the applied tractive torque, and Torque(b_i) is the braking torque.

2.4. Motor and Power Train. Modern diesel engines are essentially made up of the following subsystems.

We present in this section the simplified model for the diesel engine (see Figure 4) that we have developed, in order to be used by the SDK controller. Based on Peysson et al. [1], the dynamics of motor rotation is given by.

$$J_e \dot{w}_e(t) = M_{\text{ind}}(t - \tau_i) - M_f(t) - M_{\text{load}}(t), \tag{2}$$

where w_e is the crank shaft angular velocity, M_{ind} is the indicated torque, τ_i is the delay, M_f is the friction torque, M_{load} is the torque due to the load, and J_e is the effective inertia of the engine.

In this work, the transmission system is represented as in the Figure 5. For simplification purposes, the "Transmission System" block is composed only of several constants, depending on the gearbox state.

2.4.1. The Admission and Intake Manifold. The temperature T_{im} of the intake manifold is assumed to remain constant due to the intercooler. Therefore, the analysis will be based essentially on the variation of the pressure P_{im}. In this study, the input flow is characterized by the output flow $\dot{m}_c$ of the compressor (see (3)):

$$\dot{P}_{\text{im}} + \frac{\tau_v V_d N_e}{120 V_{\text{im}}} P_{\text{im}} = \dot{m}_c \frac{R T_{\text{im}}}{V_{\text{im}}}, \tag{3}$$

where V_{im} is the volume of the intake manifold, τ_v is the volumic efficiency, V_d is the exchange volume in the engine, and N_e is the rotational velocity of the engine in rpm ($N_e = 60 w_e / 2\pi$).

2.4.2. The Indicated Torque M_{ind}. In order to calculate the indicated torque, we should calculate firstly the indicated efficiency. Normally, this efficiency is a specific characteristic to the engine and it is found from empirical data. This efficiency is higher when the mixture is light, and it may be approximated by (4)

$$\mu_{\text{ind}} = a + b\lambda + c\lambda^2. \tag{4}$$

The coefficients (a, b, c) are found by identification (three different tests have been made) and λ is defined by (5)

$$\lambda = \frac{(P_{\text{im}} w_e V_d / 4\pi R)\tau_v}{T_{\text{im}} \dot{m}_f}. \tag{5}$$

Then the indicated torque M_{ind} can be found by (6)

$$M_{\text{ind}} = \dot{m}_f p_{c_i} \mu_{\text{ind}}, \tag{6}$$

where $\dot{m}_f$ is the flow of fuel and p_{c_i} is a characteristic for the diesel ($40000000\,\text{JKg}^{-1}$).

2.4.3. The Injection. The injection system controls the quantity of fuel that will be introduced into the combustion chamber. The mixture Air/fuel should be capable to auto ignite by the effect of temperature and the high pressure. The calorific power of combustion is related to the quantity of fuel injected. The following model gives the flow of fuel $\dot{m}_f$ in function of the position x_p of the accelerator pedal and the engine speed (see (7)) Peysson et al. [1]:

$$\begin{aligned} \dot{m}_f &= i_0 + \Delta\dot{m}_{(\text{SDK})} + \Delta\dot{m}_f, \\ \Delta\dot{m}_f &= w_e\left(i_1 + i_2 x_p + i_3 x_p^2 + i_4 w_e\right), \end{aligned} \tag{7}$$

where i_0 characterizes the minimal injection flow (greater than zero, when the engine is idle) and $\Delta\dot{m}_f$ models the variations of the flow around i_0. $i_1, \ldots, i_4$ are constants obtained by identification following simplification on the flow control model (regulator slide).

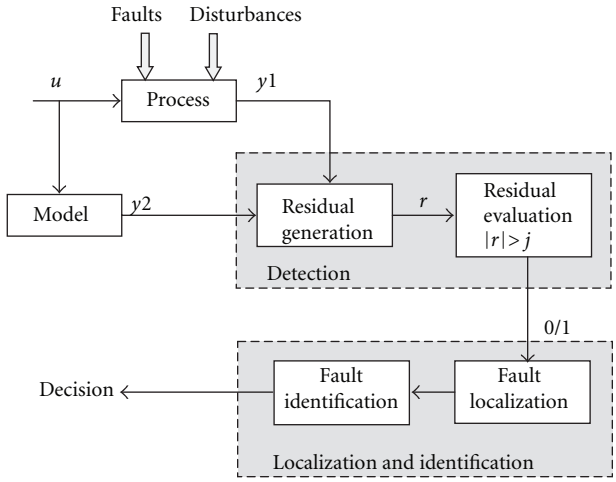

Figure 6: Model-based fault diagnosis using residual approach.

2.4.4. The Friction Torque M_f. The friction torque may be calculated by

$$M_f = \frac{(c_0 + c_1 w_e + c_1 w_e^2) V_d}{2\pi n r}. \quad (8)$$

2.4.5. The Load Torque M_{load}. The torque M_{load} depends on the type of the road, the vehicle velocity, the turnings, and so forth. In this work we will suppose that this torque is an input to the system and has a constant value.

3. Decentralized Diagnosis

In most automated systems, the command part (which implements the control of the operative part) is generally represented through a model to be applied to the operative part (mechanical components which should be controlled by means of actuators, such as engines). Realizing a diagnosis requires also to be able to represent the state of the operative part using a model that can be integrated to the one of the command part, separated or mixed. Thus, when a fault occurs, it is possible to get information regarding the process and to compare model and process. This is called model-based diagnosis, more particularly Fault Detection and Isolation (FDI) Darkhovski [10].

The method of FDI is based on the use of model-based analytical redundancy, that is, relations among the measured variables (see Figure 6). It can be divided into several steps Patten et al. [9].

(i) *First*, data containing information about the process states are transmitted to the residual generation module. This module generates a vector that carries information about symptoms and particular possible faults.

(ii) *Second*, the successive generated vectors are evaluated and filtered in order to extract the primary cause of the observed evolution, that is, to achieve fault localization and identification,. The structured residual approach Chow et al. [11] has been chosen in order to perform this stage.

3.1. Motivations. A decision-making structure for fault diagnosis must be defined to face combinatorial explosion and real-time problems and/or communication problems between various components of a process. The choice of a structure depends on the distribution of the available information (model and observation): centralized or distributed, and on the nature of the process: simple with only one control unit or complex with several local control units. There are therefore three main structures of decision-making methods for diagnosis: centralized, decentralized, and distributed.

The centralized structure consists in associating to one global model of the process a single diagnostic module (called diagnoser). It collects the different process information before making its final decision about the operating status of the process Sampath et al. [12]. Although successful in terms of diagnosis for simple systems, the centralized structure is difficult to use for large systems. Indeed, the acquisition of a global model of the process rises difficulties and often leads to combinatorial explosion problem, when it is not just impossible due to the presence of several manufacturers for the different parts of the process and privacy rules.

The decentralized structure is based on several local independent diagnosers that are associated to one global model of the process. Each diagnoser receives the observations which are specific to it and takes a local decision based on its local observations. However, this structure involves problems of indecision when some global specifications require consistency checking between decisions of local diagnosers. To solve these cases, each diagnoser sends its local decision to a coordinator (or supervisor) which will manage the different problems of ambiguity between these diagnosers and will take the final decision Wang et al. [13].

In the distributed structure, the process is modeled through its components (or subsystems) by several local models. Each one is equipped with a local diagnoser responsible of it. In the case of global specifications, a communication protocol allows directly the communication between the different diagnosers to manage conflict decision Qiu [14]. Each diagnoser makes its decision based on its own local observation and that reported by other local diagnosers as answers to its queries. This structure permits to throw off the construction of a coordinator but implies the definition of a protocol for decision making through communication between diagnosers, often impractical because without guarantee of convergence in bounded time and generating more important communication traffic and delays.

In this paper, we adapt the decentralized/distributed structure. In other words, a local diagnoser is associated with each component of the process based on a modular modelling of the plant elements. All local diagnosis decisions are transmitted via a communication environment and merged by a dedicated supervisor in order to obtain a global decision and carry out any recovery action (see Figure 7).

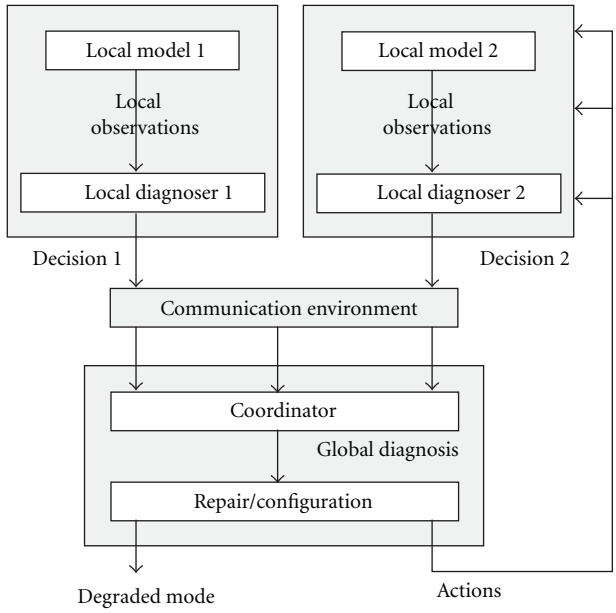

Figure 7: Model-based decentralized principle with 2 local diagnosers.

This fusion can be realized by a coordinator based on a set of rules. The goal of this coordinator is to solve the problem of decision conflict and/or ambiguity among local diagnosers in order to obtain a diagnosis performance equivalent to that of a centralized diagnoser. This approach allows carrying out on-line diagnosis without computing the global model and overcoming both the combinatorial and communication traffic explosion problems.

3.2. SDK Algorithm. As shown previously, the decision of the SDK controller depends essentially on the data issued from some sensors (wheels angular velocity sensors, radar, and transmission sensor), which means that any faulty data will influence the SDK system decision. That is why one local diagnoser is associated with each one of these sensors in order to diagnose an SDK fault by using the decentralized model-based approach (see Figure 7).

3.3. Decentralized Diagnosis of the SDK. A model of the SDK environment, that is, the part of the vehicle required to close the loop, is necessary to perform diagnosis (see Figure 7). So, by applying the laws of dynamics, a simplified model of the diesel engine has been developed. It permits to identify the angular velocities of the six wheels and the crank shaft angular velocity in response to an action (on braking system or accelerator pedal or gearbox).

3.3.1. Wheels Diagnoser. The six wheels angular velocities w_i are the inputs of a local algorithm able to detect and isolate any fault that occurs in the wheels velocity sensors. Indeed, it is possible by using these redundant measurements to generate a set of structured residuals and afterwards detect and isolate single or multiple fault.

Table 1: Angular velocity comparison for the wheels 1, 3, and 5.

Difference	$w_1 - w_3$	$w_1 - w_5$	$w_3 - w_5$
$f_{\text{odd } i}$	f_{13}	f_{15}	f_{35}

Table 2: Angular velocity comparison for the wheels 2, 4, and 6.

Difference	$w_2 - w_4$	$w_2 - w_6$	$w_4 - w_6$
$f_{\text{even } i}$	f_{24}	f_{26}	f_{46}

Table 3: Comparison of the angular velocity of the wheels: 1, 2, 4, and 6.

Difference	$w_1 - w_2$	$w_1 - w_4$	$w_1 - w_6$
f_{1i}	f_{12}	f_{14}	f_{16}

Table 4: Comparison of the angular velocity of the wheels: 3, and 2, 4, 6.

Difference	$w_3 - w_2$	$w_3 - w_4$	$w_3 - w_6$
f_{3i}	f_{32}	f_{34}	f_{36}

Table 5: Comparison of the angular velocity of the wheels: 5, and 2, 4, 6.

Difference	$w_5 - w_2$	$w_5 - w_4$	$w_5 - w_6$
f_{5i}	f_{52}	f_{54}	f_{56}

Two cases are considered based on steering angle.

(i) *Case of Straight Line Motion*. In this case, we suppose that the angular velocities of the six wheels should be approximately equal. Then in order to apply this strategy, we suppose that we have two groups: group 1 (for the wheels: 1, 3, 5), and group 2 (for the wheels: 2, 4, 6), and we calculate the differences in the angular velocities as shown in Tables 1 and 2. Then if $(w_i - w_j) < \epsilon$, we suppose there is no fault, and $f_{ij} = 0$, else we have a fault and $f_{ij} = 1$. To localise the fault in the case of $f_{ij} = 1$, a small algorithm is realized. This algorithm is able to localise from 1 to 4 faults. In the case of more than four, it gives a signal that all the wheels are faulty The realization of this algorithm is based on Tables 1, 2, 3, 4, 5. By completing these tables, the localization of the fault will be evident.

(ii) *Case of a Curve Motion*. in this case, we follow the same strategy proposed in the previous case, with the five tables, but the main difference here, in the case of the curve, is when we compare a wheel in group 1 (for the wheels: 1, 3, 5) to a wheel in group 2 (for the wheels: 2, 4, 6), we should take in consideration a small difference that can be calculated geometrically based on the Ackerman angle theory and based essentially on the steering angle. So we should replace ϵ with ϵ'.

The outputs of this algorithm are the state (normal/abnormal) s_i of the wheels sensors. In addition, it

computes the longitudinal speed TS_w of the truck which is approximated based on the nonfaulty sensors:

$$(s_1, s_2, s_3, s_4, s_5, s_6, TS_w) = \text{wheels}(w_1, w_2, w_3, w_4, w_5, w_6). \tag{9}$$

3.3.2. Transmission Diagnoser. Since there is no redundancy measurement, the algorithm just computes the longitudinal truck speed TS_t by using the value of the crank shaft angular velocity w_e (the gear box output) and the transmission rate number n:

$$TS_t = \text{transmission}(w_e, n). \tag{10}$$

3.3.3. Radar Diagnoser. The basic data detection requirement is to measure distance, relative speed, and reflection signal amplitude of moving and stationary objects in three beams. Angular position is calculated by the interpolation algorithms based on signal levels in adjacent beams.

Several scenarios for the radar fault detection analysis ar as follows:

(i) *First*, if the radar is faulty and does not detect any object. So, without the help of another device, we can do nothing,

(ii) *Second*, if the radar works but gives incorrect distances (with a certain shift of x meters): for example ($150\,\text{m} \rightarrow (150 - x)\text{m}$) where x is a constant term, then, we cannot detect this fault.

(iii) *Third*, if the relative velocity and the distance between the vehicles are measured separately (two different measurement tools), then it is important to check at each period (e.g., 2 seconds) if the variation in the distance corresponds to the variation in the relative velocity. If there is a difference then we say that there is a fault.
Exemple: suppose that we initially have the relative velocity RS and the distance RD between the SDK truck and the vehicle-in-front (see Figure 9), so if we consider that the period (that we choose for checking) is equal to 2 seconds, then we should obtain

$$d(t) - d(t-2) = 2 * \text{Avera}(Rs), \tag{11}$$

where Avera(RS) is the average value of RS during the period of 2 seconds.

(iv) *Fourth*, suppose that the radar was detecting a vehicle-in-front (see Figure 9).

As we have shown before, the relative speed RS is a measurement given by the radar. And also, the SDK vehicle velocity TS is measured from other sensors (wheels angular velocities or vehicle velocity); then we can find the velocity of the vehicle-in-front: $FS = RS + TS$.

Getting the velocity of the front vehicle, we can analyze as follows: if there is a strong sudden variation (and then its acceleration (deceleration) is not realistic), then we have one of the three following cases:

(i) there is a fault in the radar sensor;

(ii) the value of the SDK vehicle velocity is faulty;

(iii) an intruder vehicle comes in front of the SDK vehicle (see Figure 10).

In all the aforementioned cases, it is important to observe the velocity of the front vehicle for several points before taking any decision.

In some of the previous cases, the calculation of the front vehicle acceleration (deceleration) may give a nonrealistic values. A study about the maximum (minimum) possible acceleration (deceleration) can be given as follows:

$$|a_x|_{\max} = \max\left|\frac{\sum F_{x_{ij}}}{m}\right| = \max\left|\frac{\mu \cdot N_z}{m}\right| = \max\left|\frac{\mu \cdot m \cdot g}{m}\right| \leq \mu_{\max}.g. \tag{12}$$

A maximum friction coefficient ($\mu_{\max}$) determines maximum acceleration or deceleration. In order to estimate $\mu_{\max}$, sliding mode observers can be applied. A hierarchical observer is needed for this estimation. In the first step, an observer based on the dynamical equation of the wheels should be developed. This observer takes as an input the applied motor torque (which is estimated statically (existing maps)) and the braking torque, which can be easily found based on the hydraulic pressure sent to the wheels shraim [15]. Then, in parallel to this observer, a sliding modes observer is used to estimate the vertical forces. This observer is based on the suspension system modeling. Then by calculating the longitudinal force and the vertical force, we apply the following formula to estimate the adherence coefficient:

$$\mu_{\max} > \frac{\max\left(\sum_{i=1}^{n} F_{x_i}\right)}{\min\left(\sum_{i=1}^{n} F_{z_i}\right)}, \tag{13}$$

where n is the number of the wheels (equals 6), and F_{x_i} and F_{z_i} are, respectively, the longitudinal and the vertical forces applied at the wheel i.

If a nonrealistic acceleration (deceleration) value is found, then if there is no intruder vehicle, we can suppose that there is radar fault. Thus, the outputs of the local algorithm are the radar state (normal/abnormal) s_r and the velocity of the object-in-front:

$$(s_r, FS) = \text{radar}(RD, RV, TV, s_{\text{inf}}, \mu_{\max}), \tag{14}$$

where s_{inf} is a signal precising if the truck velocity is faulty or not.

3.3.4. Global Diagnoser (or Supervisor). It takes as inputs the outputs of the local diagnosers and its goal is to do some global consistency checking and to merge the local diagnosis decisions in order to obtain one global diagnosis decision and carry out the appropriate recovery action:

$$(c, TS) = \text{global}(s_1, s_2, s_3, s_4, s_5, s_6, s_r, TS_w, TS_t). \tag{15}$$

c and TS represent the control (recovery action) and the truck speed, respectively.

Several fault scenarios and recovery actions have been analyzed. The first class of scenarios is composed of wheel sensors failures and/or sensor failure on the rotation speed of the shaft engine (transmission fault). The global diagnosis and recovery actions are described as follows.

(a) If 1 to 3 wheels sensors are faulty out of the 6 (determined by the wheels diagnoser), the recovery action consists for the SDK function in using the truck speed TS_w calculated as average values provided by the correct sensors (between 3 and 5). The global diagnoser compares TS_t to TS_w and, in case of discrepancy, concludes also to a faulty transmission sensor.

(b) If 4 wheels sensors are faulty (determined by the wheels diagnoser), a comparison between the speed TS_w computed from the two ones assumed to be correct and TS_t is performed by the global diagnoser. In case of consistency, the recovery action decided by the global diagnoser consists in using the truck speed calculated as average values provided by the correct wheel sensors and the transmission sensor (3 out of 7).

(c) In all other cases, that is, when at most 2 sensors out of 7 provide consistent measurements, the recovery action decided by the global diagnoser consists in disabling the SDK function.

The second scenario category includes the radar failure. It may pass unnoticed, in particular if RD and RS measurements are not independent, because we cannot provide analytical redundancies between the truck and the front vehicle. In absence of such redundancy, the only check we can do is to verify that the behavior of the front vehicle, in terms of velocity and acceleration, as deduced by the global diagnoser from the truck's behavior (TS) and the radar measurement, is not physically impossible, that is, does not violate the laws of dynamics. In case of physical impossibility detected, then the recovery action taken by the global diagnoser consists in disabling the SDK controller.

Obviously both scenarios can combine, allowing multiple fault diagnosis of the wheels and transmission sensors and of the radar.

4. Diagnosis Deployment

In this section, we propose a deployment solution of the decentralized model-based fault diagnosis strategy (see Figure 8) in the electronic architecture of a Renault Truck's vehicle. The SDK electronic architecture part is composed of three Electronic Control Units (ECUs) that communicate between them through a bus topology. The ECUs exchange messages that follow the Control Area Network (CAN) protocol, which is low cost and very wide spread in the road transport domain. The three ECUs (ECU1, 2, and 3) are, respectively, linked to wheels, transmission and radar (+SDK algorithm) control functions. We also assume that

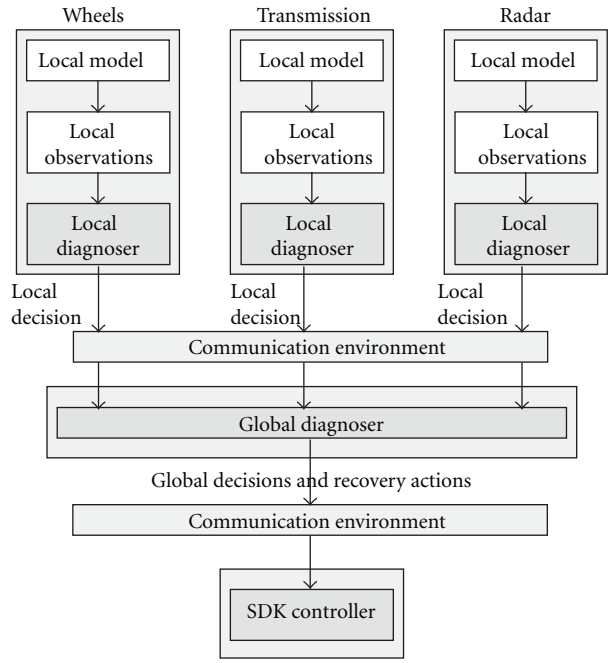

Figure 8: Decentralized model-based fault diagnosis of the SDK system.

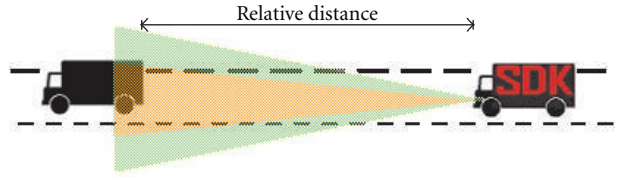

Figure 9: SDK Vehicle and the vehicle-in-front.

there is at least another ECU (ECU4) in the vehicle for other high level control functions, which is a very likely situation in modern vehicles. ECU4 will be used for embedding the global diagnoser and recovery functions.

We first list the manufacturing constraints that motivate the adopted deployment solution. We next present the classical on-board diagnosis techniques in vehicles. Finally, we describe the principle of our on-board diagnosis strategy, based on the previous electronic architecture. The next section will present an integration and validation platform.

4.1. Motivations. From an end users point of view, breakdowns and malfunctioning can lead to a loss of safety for the driver and his environment—a major breakdown which necessitates an emergency stop and repair—or a company's performance penalty, due to the need of a vehicle maintenance. From the vehicle manufacturer point's of view, these risks must be avoided because they can mostly cause a loss of corporate image and in-field yield.

Beside that, the road transport industry is a very competitive market and the integration of innovative features, such as diagnosis, is highly driven by economical considerations. That is why most of control organs of modern fuel vehicles

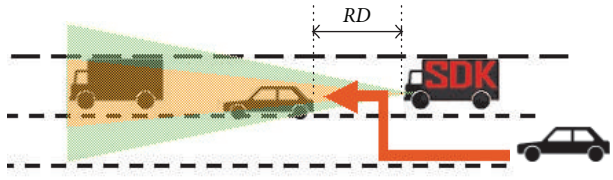

Figure 10: SDK Vehicle and the front vehicle AV with intruders (vehicle).

are embedded and architectured around a limited number of ECUs and communication buses. This evolution towards a very wide use of ECUs was also pushed by the need to reach new challenges such as environmental, performance, security, and driving assistance requirements.

As a result, the following four major requirements have driven our diagnosis deployment strategy:

(i) the number of ECUs has to remain unchanged;

(ii) the number of buses has to remain unchanged and the same communication protocol should be used if possible;

(iii) the additional bus load related to diagnosis information must not affect the current real time performance;

(iv) the diagnosis procedure must be achieved within a bounded time.

Note that, here, we do not address the problem of ECUs load sharing and balancing between the control functions already embedded in them and the diagnosis control ones that will be embedded. Nevertheless, the response time of a control function mainly depends on the total communication delay along the bus lines.

Due to the robustness of the CAN-based bus technology, it is a good candidate for enabling the deployment of a decentralized diagnosis strategy. The electronic architecture remains unchanged (no additional ECU and bus). In order to achieve the two last requirements, we developed an *SW*-based diagnosis service that is an *SW* middleware built on the top of any ECU operating system. It enables the communication of diagnosis information between the local diagnosers and the global diagnoser (as shown in Figure 8) over a loaded CAN bus. It also processes any alert with a bounded time whatever the bus load. It enables the message exchanging with a bounded time while the real-time requirements are verified (no more than 2% of delay time) whatever the bus load.

4.2. Classical Diagnosis Approaches. On-board vehicle diagnosis (OBD) refers to vehicle self-detection, localization, identification, and reporting capabilities (O'Reilly [16] and Greening [17]). Early OBD versions for fuel vehicle managed by electronic simply switched on a malfunction indicator light in the vehicle if any problem was detected. The diagnosis was next performed by an operator in a garage with the aid of a terminal connected to the vehicle electronic. Efficient diagnosis tools were developed for tracking the problem such as Ressencourt [18].

Modern OBD provides real-time diagnosis data in addition to standardized diagnosis trouble codes (DTCs) which rapidly allow the vehicle to self-identify and, possibly, self-repair by itself the problem during the driving. Otherwise, some vehicles activate a downgraded mode that allows the vehicle driving in safe conditions even in the presence of problems Fromion [19]. In parallel, the vehicle switches on a driver indicator light that points the need of an emergency maintenance (the driver must reach the closest garage) or stop.

Current SAE (Society of Automobile Engineers) and ISO (International Standardization Organization) standards specify the hardware (connector, network) and communication protocol (Open System Interconnection model) for exchanging diagnostic data over the ECUs and external terminals. Some engineering companies propose tools that allow the ECU original equipment manufacturers (OEMs) implement the standardized DTC and customer-specific diagnosis requirements in their ECU, such as Frank et al. [20] (diagnostic-oriented process flow).

4.3. Deployment of the Decentralized Diagnosis. The decentralized diagnosis system, described in Section 3, is deployed in an electronic architecture of four ECUs which communicate over a single CAN-based bus.

The local diagnosers only transmit boolean signals, which take two states: *normal* or *abnormal*, to the global diagnoser. Note that the local diagnosers do not exchange directly diagnosis information with each other. A local diagnoser outputs a *normal* state whenever no fault is detected. During this *normal* state, no additional bus load is due to the diagnosis protocol. Conversely, an *abnormal* state indicates that a problem has likely occurred in a sensor, and when it appears, the local diagnoser must immediately send diagnosis information to the remote global diagnoser in order to compute a global diagnosis and, if the need arises, to perform a recovery procedure.

For enabling this event-based procedure, we insert a Local Diagnosis Service (LDS) in every ECU that embeds a local diagnoser. An LDS reads the outputs of the local diagnoser (*normal*/*abnormal*). When an event "*normal* → *abnormal*" occurs, it virtually creates a high-priority communication channel between the local diagnoser and the global diagnoser, so that the former can immediately transmit the diagnosis information to the latter within a bounded time. The diagnosis information is embedded in CAN messages. In addition, a CAN message contains a control header that especially defines the transmitter and receiver identifiers and the priority level of the communication. On the opposite side, a Supervision Diagnosis Service (SDS) is inserted in the global diagnoser ECU. It monitors the bus load and gathers the diagnosis information sent by any LDS. It next triggers the global diagnoser. Note that the LDS and SDS can periodically check if, respectively, the SDS or any LDS is safe and ready.

Figure 11 illustrates the different behavior phases of both LDS and SDS before and after the occurrence of a problem.

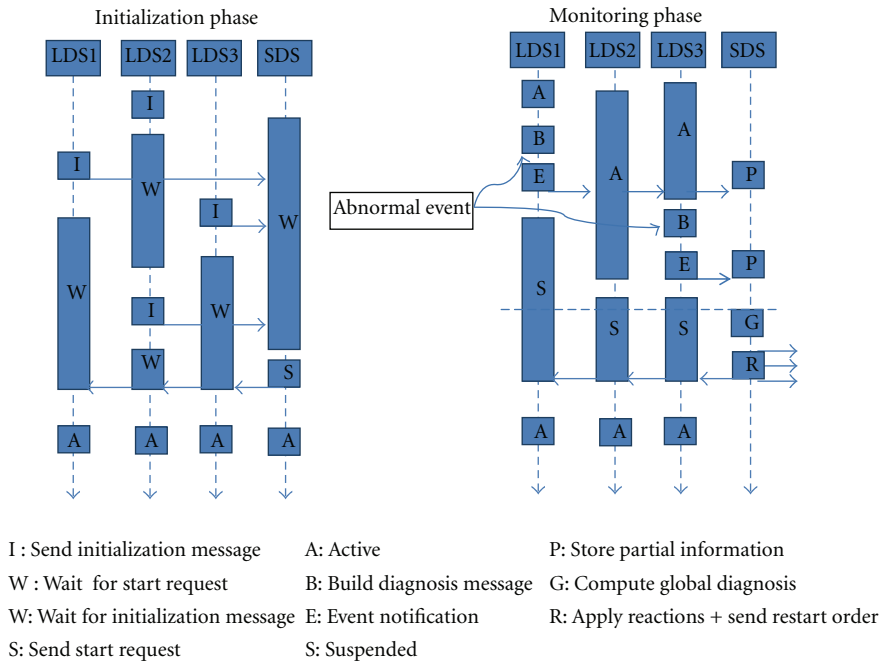

Figure 11: Decentralized diagnosis protocol over a CAN bus.

Initialisation Phase. The first phase consists in the establishment of the list of available LDSs. Each LDS sends to the SDS a specific CAN message for initialization which includes its identification number. When the SDS has received the initialization message from all LDSs, the protocol enters the monitoring phase. This instant is materialized by the broadcasting of a specific message from the SDS to all LDSs.

Monitoring Phase. During this phase, no messages are exchanged between the LDSs and the SDS. This phase corresponds to a situation where the system is operating normally, without local diagnosis event from the local diagnosers. This implies that there is not any overtraffic due to the diagnosis during the normal operation of the system.

Alert Phase. This phase begins when an abnormal event is detected by a local diagnoser. At this moment, the LDS of the concerned ECU sends a specific alert event message to the SDS. The instant of the first alert event message emission materializes the beginning of a period when the SDS is waiting for other alert event messages that would complete the information for the global diagnosis. At the end of this period, the SDS gives the order to the global diagnoser to compute the global diagnosis. When this is done, the SDS broadcasts a specific message to all LDSs for entering again in the monitoring phase.

Vivacity Check. While the protocol is in monitoring phase, the SDS can initiate a vivacity check: the SDS broadcasts a specific CAN message to all LDSs. When the LDSs receive this message during their monitoring phase, they send an acknowledgement message for proving that the on-board diagnosis is correctly running.

5. Evaluation of the Diagnosis

5.1. Evaluation of the Diagnosis Algorithms

5.1.1. Presentation of the Scenarios. Several fault scenarios were chosen to valid the diagnostic approach for this SDK system. The selection was done according to

(i) relevance of fault during SDK system operation,

(ii) ease of inducing the fault,

(iii) exclusion of danger possibly caused by the induced fault.

Such faults in the SDK system are failures of the wheels, the transmission, and the radar. All these failures can occur intermittently. Moreover, single or multiple faults may be considered.

5.1.2. Evaluation of the Scenarios. We developed a physical simulation model of the SDK environment in MATLAB/Simulink in order to evaluate our model-based diagnosis approach. The simulation serves as a virtual test bed where we can easily study a large number of fault scenarios to develop our diagnosis models and test our algorithms. We adopted a component-based modeling paradigm, where parameterized simulation models of generic components (SDK controller, radar, wheels, transmission, engine, and supervisor) were developed within a component library. The different local models are constructed by instantiating different components from the library, specifying their

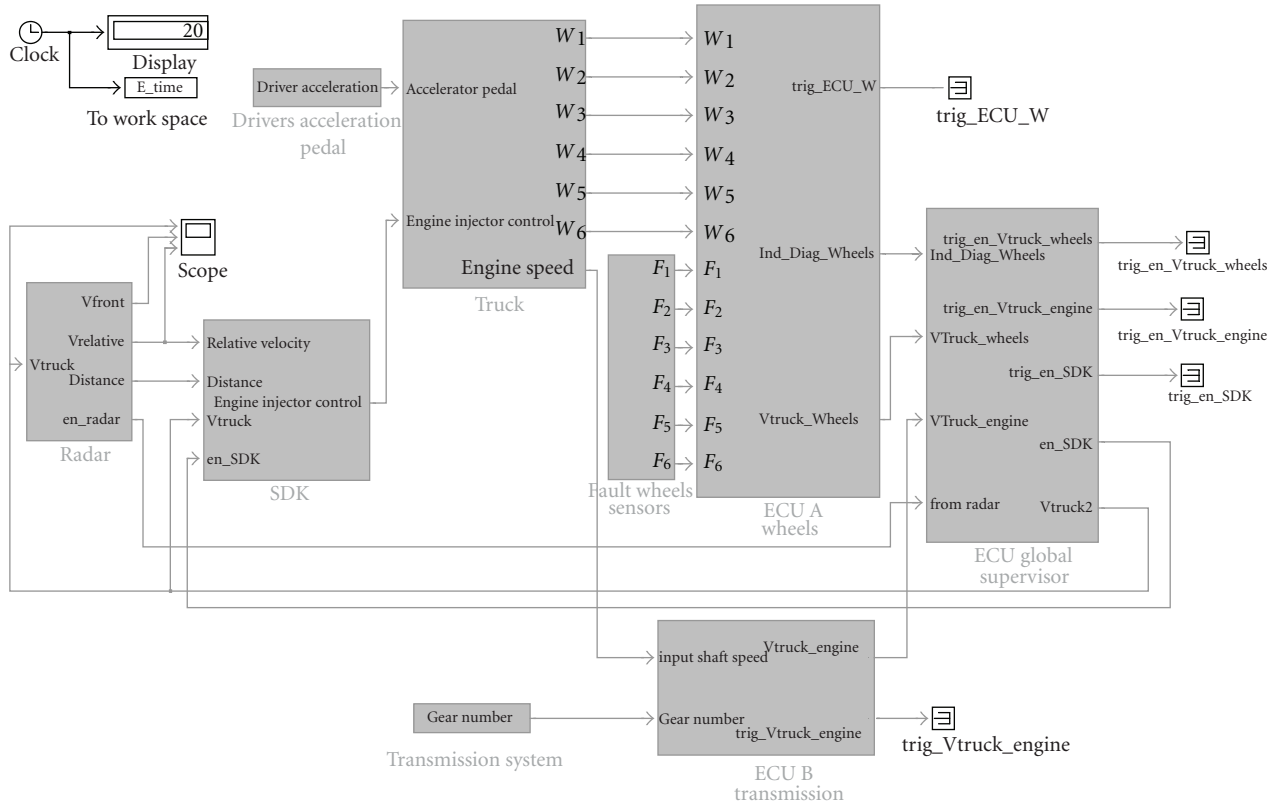

Figure 12: Simulink block diagram for detection of SDK sensor faults.

parameters, and connecting the components to each other in the appropriate fashion. The top level of the Simulink block diagram is illustrated in Figure 12.

Except the supervisor, each component model includes its associated fault modes. The fault mode, time of fault injection, and fault magnitude (where applicable) can all be specified. In general, each fault mode is mapped to a change in component mode and a fault-dependent magnitude parameter. Because each fault mode is parameterized within the Simulink model, a fault can be injected programmatically (i.e., the fault mode, injection time, and magnitude are specified) either at the beginning of the simulation or while the simulation is running.

As mentioned previously, a fault will be detected by observing residual values which should be close to zero in the nominal behavior of the process, otherwise, significantly different from zero. Ideally the residual signal should carry only information about faults but, practically, it also contains disturbances, which is the effect of model uncertainty. It is necessary in this case to establish thresholds on residuals to avoid false alarms. The fault occurs only when the residual values exceed the prescribed threshold. For this, we have analyzed the residual values in the absence of fault, which helped determine the residual magnitude in functioning healthy state. The resulting thresholds are then used to the fault detection mechanism.

By using this simulator, the fault scenarios and recovery actions presented in Section 3.3 have been performed. Then, the injected fault has been detected and localized. Finally, the appropriate recovery action has been applied.

5.2. Performance Evaluation of the CAN Integration. A prototype of the diagnosis system has been realized, according to the diagnosis organization presented in Section 3 and using the deployment scheme described in Section 4. The objective of this prototype is to validate the functionality of the diagnosis implementation and the performance of the deployment architecture. More specifically, we want to check the real-time property provided by the proposed deployment scheme. For that, we developed a validation environment that allows the emulation of both LDS and SDS modules with a high accuracy (real time simulation). In the following, a description of the experiment will be done; then we will present and analyze the results.

5.2.1. Prototype Presentation. As represented in Figure 13, some parts of the prototype are emulated with a simulation tool while others are implemented in a real hardware ECU. The electromechanical truck subsystem models and the local diagnosis algorithms are modeled with the Matlab Simulink simulation tool. Every LDS module, presented in Section 4, is modeled in the CANoe tool Frank and Schmitds [20], which is a suitable emulator of CAN-based systems. This tool simulates the behavior of a CAN-based system with a high accuracy (real time). In addition, this tool allows

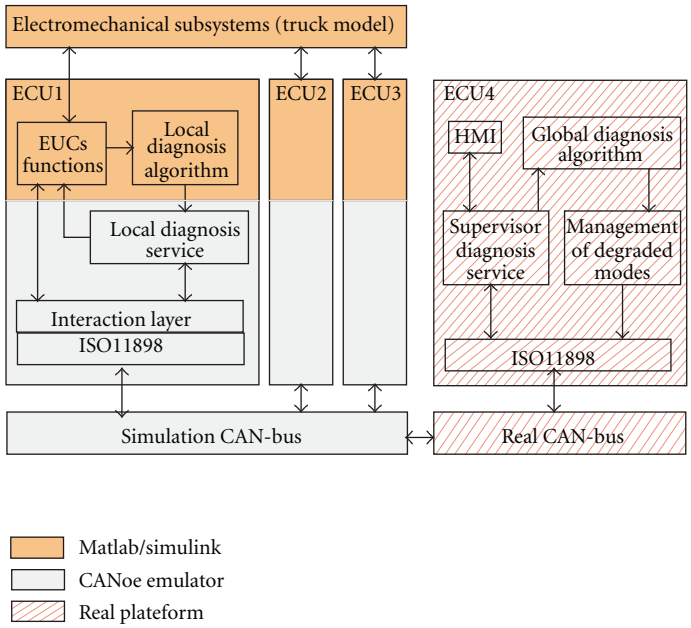

Figure 13: General diagram of the prototype of embedded diagnosis of the SDK function.

a cosimulation with a real CAN-bus and real ECU. For performance evaluation, a CAN based traffic generator was included for generating different reference loads that are adjusted for evaluating under which condition the LDS-SDS protocol under study verifies the real-time condition. In addition, the priority of the bus load can be configured as *lower* or *higher*, compared to the messages' priority of our diagnosis protocol. The global diagnosis algorithm (Section 4), the SDS module (Section 4), and a human-machine interface (HMI) are implemented in a hardware ECU which is the Freescale board MAC7100EVB, equipped with an ARM MAC7111 microprocessor, which includes 4 CAN peripherals, one of which is used by our application. The CAN bus is configured at a bit rate of 250 kbit/s.

For stimulating the diagnosis protocol between the LDSs and the SDS, a fault is injected in the electromechanical model of the truck. The fault generates a divergence between the model and the golden model of the local diagnosis. We considered 8 scenarios that correspond to various configurations of load values and priorities. For each scenario, a series of 20 faults is applied. The different scenarios are given in Table 6.

5.2.2. Performance Results. The experimental results are presented in Table 7. We observe that if the bus load does not exceed 60%, no particular impact can be noticed on the processing latency of the diagnosis, for higher and lower load priorities. But when the load priority is high and the load is over 60%, an impact on the latency can be measured, which is growing with the bus load level. These configurations must be avoided for ensuring a reliable use of this decentralized diagnosis algorithm.

In practice, the CAN bus load of vehicles never exceeds 40% by design, for avoiding congestion problems, so considering this statement, our decentralized diagnosis algorithm verifies the real-time condition, that is, the delay between the LDS and SDS over the network remains quite constant whatever the bus traffic conditions.

Table 6: Scenarios for the evaluation of the CAN integration.

Configuration	Load priority	Load level (%)
Config0	No Load	0
Config1	Lower	2.67
Config2	Lower	8.11
Config3	Lower	62.44
Config4	Higher	2.65
Config5	Higher	8.06
Config6	Higher	62.02
Config7	Higher	97.18

Table 7: Performance evaluation of the CAN integration.

Configuration	Processing latency of the diagnosis (s)			
	Min	Max	Average	Std. dev.
Config0	20.72	21.09	20.91	0.37
Config1	20.82	22.22	21.17	0.52
Config2	20.75	22.24	21.34	0.65
Config3	20.60	21.02	20.87	0.13
Config4	20.75	22.00	21.11	0.41
Config5	20.67	21.05	20.93	0.12
Config6	20.68	28.09	21.81	2.26
Config7	21.06	101.76	32.90	30.37

6. Conclusions

In this paper, we have developed a diagnosis algorithm of the SDK system and a software architecture in order to board it inside truck vehicle in a decentralized manner. The model-based diagnosis approach has been used because it presents the advantage that no prior knowledge of possible faults or symptoms is needed. It relies only on a given model of the correct functioning of the system and proceeds by comparing the behaviors of the model and of the actual system (as known through the observations given by sensors).

The originality of the accomplished work is based on two contributions. The first one is the distribution of diagnosis algorithms on several ECUs by using the decentralized diagnosis approach (the method has only been applied, in the practical context of industrial applications, in a centralized manner). This approach uses a set of diagnosers. Each diagnoser observes a part of the SDK system and takes a local decision about the occurrence of a fault and its localization. The construction of the local diagnosers is based on a modular modeling of the plant elements. All local diagnosers decisions must be merged by a dedicated supervisor in order to obtain one global diagnosis decision and to take also any recovery action. This fusion can be realized by a coordinator. The second contribution is the design and deployment of the decentralized model-based fault diagnosis approach for the SDK system. The attention was paid to minimize the additional traffic generated by the diagnosis function and to respect the real application constraints (performance, diagnosis latency, etc.). An on-board diagnosis using Hardware-in-the-Loop scheme under specifications (constrains) near to a truck "Renault Trucks" has been performed.

The decentralized embedded diagnosis for the SDK system inside real vehicles is mature from the point of view of research and of feasibility. However, we should extend and develop algorithms robustness, take into account the protocols and details of the hardware architecture and existing software, conduct tests on many scenarios, and measure in situ the quality (correctness, precision, time delay) of the diagnosis and its compatibility with existing functions (induced traffic on the CAN-bus, transparency). Moreover, in order to verify a priori that the set of local diagnosers and supervisor is capable of diagnosing a given set of faults within a bounded delay, a notion of diagnosability must be studied.

Acknowledgments

The work reported in this paper was supported by the project DIAFORE (Diagnosis for Distributed Functions) which is funded by the French National Research Agency (ANR) and SYSTEM@TIC PARIS-REGION Cluster, with the support of French Environment and Energy Management Agency (ADEME) and French Programme of Research, Experimentation and Innovation in Land transport (PREDIT). Many thanks to industrial partners (Renault Trucks/Volvo SAS and SERMA INGENIERIE) in this project for their support upon completion of the simulation platform.

References

[1] F. Peysson, H. Noura, and R. Younes, "Diagnostic de défauts sur un moteur diesel," in *Proceedings of the Conférence Internationale Francophone d'Automatique (CIFA '06)*, Bordeaux, France, 2006.

[2] L. C. Davis, "Effect of adaptive cruise control systems on traffic flow," *Physical Review E*, vol. 69, no. 6, Article ID 066110, 8 pages, 2004.

[3] R. Parasuraman and V. Riley, "Humans and automation: use, misuse, disuse, abuse," *Human Factors*, vol. 39, no. 2, pp. 230–253, 1997.

[4] N. A. Stanton and P. Marsden, "From fly-by-wire to drive-by-wire: safety implications of automation in vehicles," *Safety Science*, vol. 24, no. 1, pp. 35–49, 1996.

[5] L. Nilsson, "Safety effects of adaptive cruise control in critical traffic situations," in *Proceedings of the nd World Congress on Intelligent Transport Systems: Steps Forward*, vol. 3, pp. 1254–1259, VERTIS, 1995.

[6] N. A. Stanton, M. Young, and B. McCaulder, "Drive-by-wire: the case of driver workload and reclaiming control with adaptive cruise control," *Safety Science*, vol. 27, no. 2-3, pp. 149–159, 1997.

[7] J. D. Lee and K. A. See, "Trust in automation: designing for appropriate reliance," *Human Factors*, vol. 46, no. 1, pp. 50–80, 2004.

[8] M. A. Goodrich and E. R. Boer, "Model-based human-centered task automation: a case study in ACC system design," *IEEE Transactions on Systems, Man, and Cybernetics Part A*, vol. 33, no. 3, pp. 325–336, 2003.

[9] R. J. Patten, P. M. Frank, and R. Clark, *Issues of Fault Diagnosis for Dynamic Systems*, Springer, London, UK, 2000.

[10] B. Darkhovski and M. Staroswiecki, "A game-theoretic approach to decision in FDI," *IEEE Transactions on Automatic Control*, vol. 48, no. 5, pp. 853–858, 2003.

[11] E. Chow and A. Willsky, "Analytic redundancy and the design of robust failure detection systems," *IEEE Transactions on Automatic Control*, vol. 29, pp. 603–614, 1984.

[12] M. Sampath, R. Sungupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete event systems," in *Proceedings of the 11th International Conference Analysis Optimization of Systems: Discrete Event Systems*, Sophia-Antipolis, France, 1994.

[13] Y. Wang, T. S. Yoo, and S. Lafortune, "New results on decentralized diagnosis of discrete event systems," in *Annual Allerton Conference on Communication, Control and Computing*, 2004.

[14] W. Qiu, *Decentralized/distributed failure diagnosis and supervisory control of discrete event systems*, Ph.D. thesis, Iowa State University, 2005.

[15] H. Shraim, *Modeling, Estimation and control for vehicle dynamics*, Ph.D. thesis, Université Paul Cézanne, Aix Marseille III, 2007.

[16] P. O'Reilly, "An overview of the potential contribution of diagnostics to improving vehicle safety and reducing vehicle emissions," in *IEEE Colloquium on Vehicle Diagnostics in Europe*, pp. 1–12, February 1994.

[17] P. Greening, "On-board diagnostics for control of vehicle emissions," in *IEEE Colloquium on Vehicle Diagnostics in Europe*, pp. 1–6, February 1994.

[18] H. Ressencourt, *Diagnostic hors-ligne à base de modèles: approche multi-modèle pour la génération automatique de séquences de tests. Application au domaine de l'utomobile*, Ph.D. thesis, 2008.

[19] A. Fromion, Apparatus for the differential transmission of information between at least two devices in a vehicle. European patent EP19960401324, 2003.

[20] H. Frank and U. Schmidts, Vehicle Diagnostics—The whole Story. Vector Informatik, Article press, 2007, http://www.vector.com/.