The Car and The Cloud: Remote Automotive Electronic Controller Unit Diagnostics, Testing, and Reprogramming

Rahul Mangharam, University of Pennsylvania

In 2010, over 20.3 million vehicles were recalled. Software issues related to automotive controls such as cruise control, anti-lock braking system, traction control and stability control, account for an increasingly large percentage of the overall vehicles recalled. There is a need for new and scalable methods to evaluate automotive controls in a realistic and open setting. We have developed AutoPlug, an automotive Electronic Controller Unit (ECU) architecture between the vehicle and a Remote Diagnostics Center to diagnose, test, update and verify controls software. Within the vehicle, we evaluate observer-based runtime diagnostic schemes and introduce a framework for remote management of vehicle recalls. The diagnostics scheme deals with both real-time and non-real time faults, and we introduce a decision function to detect and isolate faults in a system with modeling uncertainties. We also evaluate the applicability of “Opportunistic Diagnostics", where the observer-based diagnostics are scheduled in the ECU's real-time operating system (RTOS) only when there is slack available in the system, i.e. it can work with existing hardware in current vehicles. This aperiodic diagnostics scheme performs similar to the standard, periodic diagnostics scheme under reasonable assumptions. This approach works on existing ECUs and does not interfere with current task sets. The overall framework integrates in-vehicle and remote diagnostics and serves to make vehicle recalls management a less reactive and warranty management a cost-intensive procedure.

The increasing complexity of software in automotive systems has resulted in the recent rise of firmware-related vehicle recalls due to undetected bugs and software faults.

In 2009, Volvo recalled 17,614 vehicles due to a software error in the engine-cooling fan control module [1] According to the NHSTA report, the error could result in engine failure and could possibly lead to a crash. In August 2011, Jaguar recalled 17,678 vehicles due to concerns that the Cruise Control in those vehicles may not respond to normal inputs and once engaged could not be switched off [2]. In November 2011, Honda recalled 2.5 million vehicles to update the software that controls their automatic transmissions [3].

While there is a significant effort for automotive software testing and verification at the design stage [4], not all possible runtime faults throughout the vehicle's lifetime can be detected. A systematic approach and infrastructure for post-market runtime diagnostics for the control software is lacking in current automotive systems. Once the vehicle leaves the dealership lot, its performance and operation safety is a black box to the manufacturers and the original equipment providers. For the over 100 million lines of code and over 60 ECUs in a vehicle [5], there are only about 8 standard Diagnostic Trouble Codes (DTCs) for software, and those too are generic (e.g. memory corruption)[6]. Of the DTCs for software, none target the *control software* in the ECUs even though control systems like stability, cruise, and traction control are safety-critical systems.

While this effort includes electric vehicle drivetrains, vehicle-to-vehicle networking, automotive Plug-n-Play systems and large-scale vehicle traffic management, the focus of this paper is on in-vehicle architectures of the future.

A. Runtime in-vehicle Diagnostics and Recalls Management

The current approach for handling vehicle recalls is reactive where the manufacturers announce a recall only after the problem occurs in a significant population of deployed vehicles and all vehicles of that particular year/make/model are recalled. A software recall

involves the vehicle being taken to service center and a technician either manually replaces the ECU which contains the faulty code, or reprograms the code onboard the ECU with the new version provided by the manufacturer. One problem with this method is that the decision to recall vehicles involves word-of-mouth or manually logged information going from the service centers to the manufacturer, which takes time and in the meanwhile may result in a malfunction within a safety critical system. This wait-and-see approach to recalls has a significant cost in both time and money and has a negative impact on the vehicle manufacturers reputation.

Consequently, there is an urgent need for systematic post-market in-vehicle diagnostics for control systems software such that issues can be detected early. The in-vehicle system would be responsible for data logging of sensor values and runtime evaluation of controller states. To complement this, a Remote Diagnostics Center (RDC) would receive this data, over a network link, to prepare an appropriate Fault Detection and Isolation response (see Fig. 1). This would normally be in the form of sending a custom Dynamic Diagnostic Code which observes the ECUs and controller tasks in question. Once sufficient data is captured, the RDC, using a model of the plant, is able to execute a grey-box structured system identification to build a plant model of the particular vehicle. Using this vehicle-specific plant model, the RDC develops a fault-tolerant controller for the issue and the vehicle is remotely re-programmed via a code update. While this approach is difficult in practice as it would require extensive runtime verification of the patched controller, we present the early design of

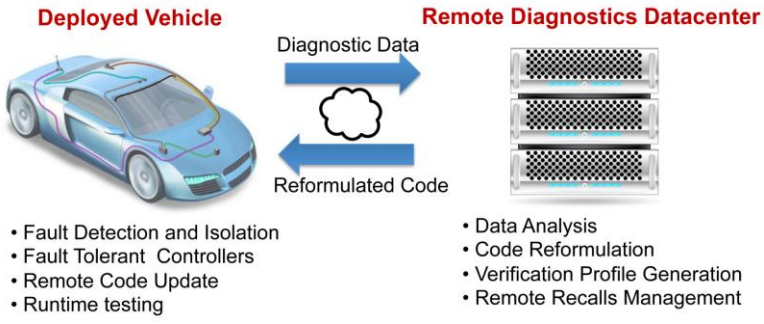

Fig. 1. Remote Diagnostics of automotive control systems

such a system with AutoPlug.

B. Overview of AutoPlug

The AutoPlug automotive architecture aims to make the vehicle recalls process a less reactive one with a runtime system for diagnosis of automotive control systems and software. Our focus is on the on-line analysis of the control system and control software within the vehicle ECU network and between the vehicle and the RDC. We assume the network link between the two is available.

The runtime system *within the vehicle* is responsible for:

(i) Fault Detection and Isolation: Sensor, actuator and controller states are logged for the specific ECU. This data is analyzed locally and a summary of the states are transmitted to the RDC.

(ii) Fault Tolerant Controllers: Once a fault is detected, the high-performance controller is automatically replaced with a backup controller.

(iii) ECU re-programming for remote code updates: Upon reception of reformulated

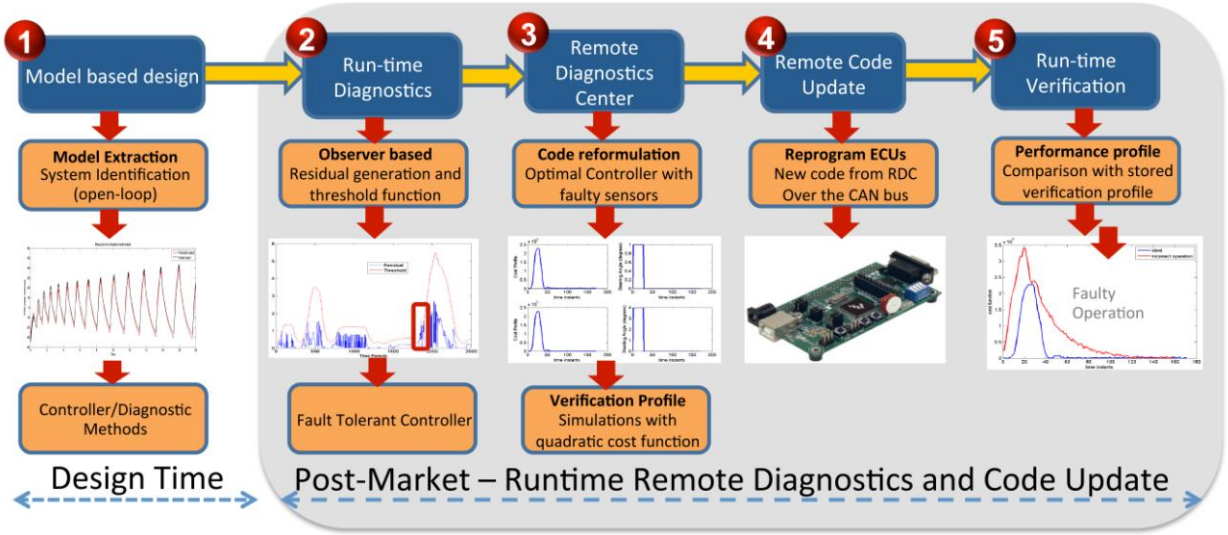

Fig. 2. End-to-end stages of the AutoPlug automotive architecture

controller code from the RDC (which will guarantee the stability and safety of the particular vehicle), the runtime system re-flashes the particular controller task(s) with the updated code. This can be done over a cellular or wireless communications link.

(iv) Patched Controller runtime-verification: The updated code is monitored with continuous checks for safety and performance.

While the on-board system provides state updates of the specific controller, the *Remote Diagnostics Center (RDC)* provides complementary support by:

(i) Data analysis and fault localization: Using grey-box structured system identification, a plant model of the particular vehicle is created. The existing controller is evaluated on this model to isolate faulty behavior.

(ii) Reformulating Control and Diagnostics Code: A new controller is formulated for the specific plant model and further diagnostics code is dispatched.

(iii) Recalls Management: Reformulated controller code is transmitted to the vehicle.

(iv) Generating Controller Verification profiles: The updated controller is probed for performance and safety.

A more descriptive view is provided in Fig. 2. This system is capable of diagnosing and reformulating controllers with real-time faults (delay, jitter, incorrect sampling rates) and system faults (stuck-at faults, calibration faults and noise in sensors/actuators).

C. AutoPlug Test-bed

In order to design and validate the proposed architecture we developed the AutoPlug test-bed that consists of a Hardware-in-the-Loop simulation platform for ECU development and testing (see Fig. 3). The hardware is in the form of a network of ECUs interfaced by

CAN, on which we implement the control and diagnostic algorithms. Each ECU runs the nano-RK RTOS [7], a resource kernel with preemptive priority-based real-time scheduling. Instead of a real-vehicle, our plant uses The Open-source Race Car Simulator (TORCS). This provides physics-based high-fidelity vehicle models and different road terrains. The testbed provides us with the realism of using a real vehicle, and also has enough flexibility to implement our own code. In addition, we can introduce faults that are not covered by set of standard Diagnostic Trouble Codes (DTC). We have tested out basic control algorithms, running as real-time tasks on nano-RK, for Anti-Lock Braking System (ABS), Traction Control, Cruise Control and Stability Control to see that the testbed indeed performs like a real vehicle would. AutoPlug is free and open-sourced.

D. Discussion

The main contributions of this effort are: (a) An architecture is introduced which uses both in-vehicle and remote diagnostics for remote recalls management of deployed vehicles; (b) We present a modification of the traditional observer-based FDI scheme for in-vehicle *opportunistic diagnosis*, as well as an experimental thresholding scheme for fault detection and isolation in presence of modeling uncertainties; (c) Finally, we implement

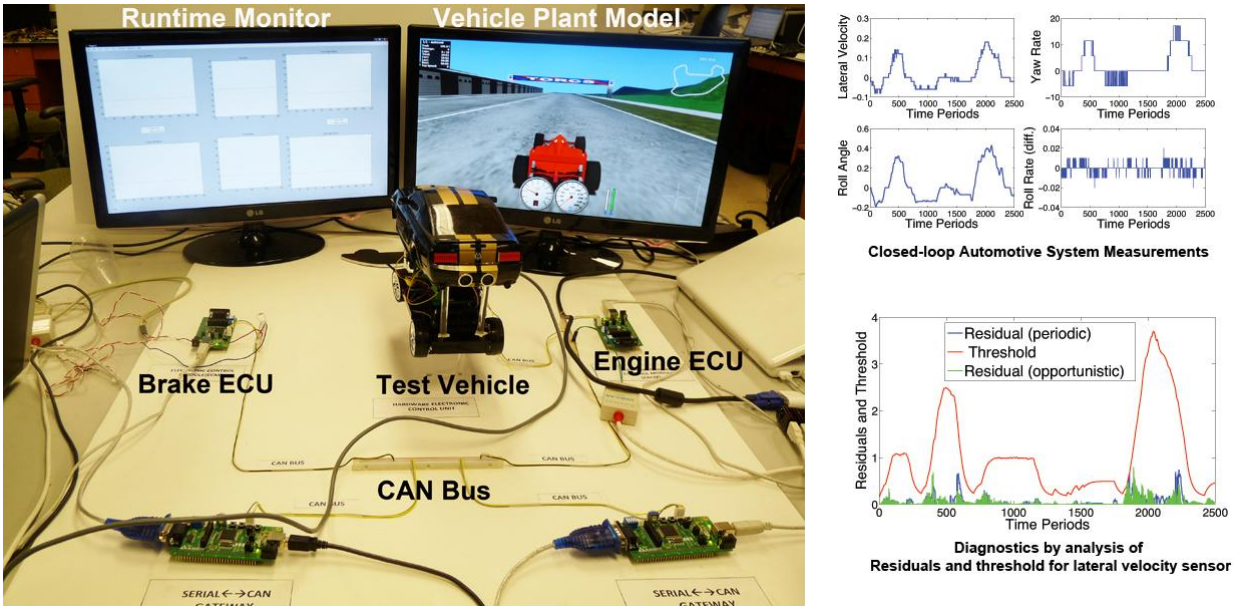

Fig. 3. AutoPlug hardware-in-loop testbed showing real-time system state monitoring and diagnostics via residual analysis

and evaluate these schemes on real ECUs on the AutoPlug test-bed for Hardware-In-Loop simulation. This effort is a step in the direction for safer automotive software by facilitating post-market diagnostics, testing and reconfiguration from a remote datacenter.

REFERENCES

[1] NHTSA Campaign ID: 09V218000. www.safecar.gov

[2] Jaguar Software Issue May Cause Cruise Control to Stay On. http://spectrum.ieee.org

[3] Honda recalls 2.5 million vehicles on software issue. http://www.reuters.com.

[4] AUTOSAR Homepage. http://www.autosar.org/

[5] J. Schaufalle and T. Zurawka. *Automotive Software Engineering.* SAE International, 2005.

[6] On-board Diagnostic Codes. http://www.obd-codes.com

[7] nano-RK Sensor RTOS. http://nanork.org

ADDITIONAL READING:

[1] U. Drolia and Z. Wang and Y. Pant and R. Mangharam. AutoPlug: An Automotive Test-bed for Electronic Controller Unit Testing and Verification. 14th IEEE Conf. on Intelligent Transportation Sys., 2011.

[2] K. Pattipati et. al. Fault Diagnosis and Prognosis in a Network of Embedded Systems in Automotive Vehicles. NSF-NIST-USCAR Workshop on Cyber-Physical Systems, 2011.

[3] P. Ioannou Y. Huo and M. Mirmirani. Fault-Tolerant Control and Re- configuration for High Performance Aircraft: Review. CATT Technical Report, 2001.

[4] X. Liu, K. Lee, Q. Wang, and L. Sha. ORTEGA: An Efficient and Flexible Software Fault Tolerance Architecture for Real-Time Control Systems. IEEE Transactions on Industrial Informatics, 2008.

[5] E. Eyesi and J. Bai and D. Riley and J. Weng and Y. Xue and X. Koutsoukos and J. Sztipanovits. NCSWT: an integrated modeling and simulation tool for networked control systems. 15th ACM Conf. on Hybrid Systems: Computation and Control, 2012.

[6] Keep Connected Cars Up to Date with FOTA (Firmware Over-the-Air) Technology. http://www.qnx.com/news/webseminars/fota.html.

[7] T. Flach and N. Mishra and L. Pedrosa and C. Riesz and R. Govindan. CarMA: Towards Personalized Automotive Tuning. 9th ACM Conf. on Embedded Networked Sensor Systems, 2011.

[8] M. Blanke, C. W. Frei, F. Kraus, R. J. Patton, and M. Staroswiecki. What is Fault Tolerant Control.

[9] R. Patton and P. Frank and R. Clark. Fault Diagnosis in Dynamic Systems. Prentice Hall, 1989.

[10] D.C. Fosth R.N. Clark and V.M. Walton. Detecting Instrument Malfunctions in Control Systems. IEEE Transaction on Aerospace and Electronic Systems, 1975.

[11] H. E. Tseng, B. Ashrafi, D. Madau, T. A. Brown, and D. Recker. The Development of Vehicle Stability Control at Ford. IEEE Trans. on Mechatronics, 4(3):223–234, 1999.