Holistic Data-Driven Diagnosis for Dependable Automotive Systems

Patrick E. Lanigan
Carnegie Mellon University
Electrical & Computer Engineering
planigan@ece.cmu.edu

Priya Narasimhan
Carnegie Mellon University
Electrical & Computer Engineering
priya@cs.cmu.edu

Themes

Dependable run-time computing and communication infrastructure; Learning and adaptable systems, self-healing and self-reconfigurable systems; Characterization of unintended, unexpected, and emergent behaviors; Analysis of nondeterministic behaviors arising from integration of independently-designed deterministic components.

Abstract

Despite extensive design processes, emergent behavior will still appear at runtime in dependable automotive systems. Such behavior occurs due to unexpected or unidentified interactions and dependencies between system components. These interactions are unidentified due to a disconnect between various stages of the design process. A diagnostic advisor that synthesizes data from each stage of the product lifecycle provide a more accurate design-time characterization of the system, as well as more robust runtime operation.

1 Introduction

The automotive industry has become steadily more reliant on software-intensive distributed systems to implement advanced vehicle features. In fact, it has been estimated [1] that "up to 90% of all innovations are driven by electronics and software". It is further estimated [1] that 50-70% of an ECU's development costs come from software, with some vehicles having up to 70 ECUs. Overall, electronics and software can account for up to 40% of a vehicle's cost [1].

A growing trend is toward features that assist the driver in maintaining safe control of the vehicle under a variety of conditions. Previously, such assistance has been provided *passively* in the form of information or warnings. These features are now being given increasing amounts of authority to control the vehicle's motion by *actively* supplementing the driver's inputs. The long term trend is towards fully autonomous operation.

Because these systems are critical to ensuring the safe operation of the vehicle, they must be designed to tolerate faults and provide high levels of dependability. Typically, a systematic safety analysis is conducted during the design phase, to evaluate both the severity and likelihood of the consequences of possible faults. Formal verification methods are used to analyze system dependability. Fault-injection can play a complementary role in this analysis by providing an empirical way to study the system's dependability.

Despite these processes, systems still might exhibit emergent behavior due to unforeseen interactions and complexity between independently designed components, Such interactions are not readily apparent to the system designers, and might not be captured by system models. When problems in software-based systems are uncovered after a vehicle has gone to production, recall costs can rival development costs. For example, 2004 saw the recall of 680,000 Mercedes-Benz E-Class vehicles due to issues with the electronic brake-by-wire system [2].

2 Holistic Data-Driven Diagnosis

Most of the work in the area of diagnosis for automotive systems has focused on online detection followed by offline diagnosis. For example, diagnostic trouble codes (DTCs) are set based on errors that are detected during vehicle operation. Such errors are typically detected by mechanisms that operate at a component level. These trouble codes are then analyzed offline in a service garage or by a remote service like OnStar.

Just as quality needs to be "designed in" to the system as early as possible, the development and refinement of a diagnosis strategy must begin early in the design process. We propose a holistic diagnosis strategy that spans the entire lifecycle of a system, from specification to validation to deployment and operation (runtime instantiation). The intent is to span the disconnect between model-based diagnosis and emergent behavior at runtime.

The approach is data-driven in the sense that documents and other artifacts of the design process, including models, become data that are input into a *diagnostic advisor*. The diagnostic advisor synthesizes this data at each stage, and provides outputs to guide the next stage. In this way, the advisor becomes more omniscient as you progress through the cycle. The design-time diagnostic advisor will aid in identifying dependencies and interactions that are not apparent to the system designer.

After synthesizing the data gathered during design and development, the outputs from the advisor can be used to diagnose problems at runtime. As opposed to current diagnostic modules that operate strictly at a component level, at runtime the diagnostic advisor will operate at a system-level. Moreover, the diagnostic advisor will be self-correcting over time, by detecting emergent behavior after runtime, and feeding it back into the process (i.e., it will leverage the scale of operating data to continuously improve the runtime diagnostic process according to the law of large numbers).

2.1 Challenges

There are challenges related to each of the three aspects of the diagnostic advisor: *holistic*, *data-driven* and *diagnosis*. For instance, the diagnostic advisor is influenced by every phase of the lifecycle, so it is important to extract data from every phase. However, it is even more important to extract the *right* data. Moreover, this data comes at the expense of human (designer) overhead as well as system and communication overhead. This overhead will need to be minimized, as well as balanced with the benefits provided by the diagnostic advisor. Appropriate analytic techniques will then need to be developed, applied and evaluated in order to provide actionable diagnostic output.

Additional challenges include the following:

- How do you identify the important pieces of information and discard irrelevant data?
- How do you collect real-world data following runtime instantiation and deployment?
- How do you encode the various pieces of information in a standard machine-understandable format?
- How do you synthesize diagnostic information between lifecycle stages?
- How do you handle incomplete or partial information, especially information that is not known until runtime?
- What is the cost / performance ratio to every stage during the lifecycle, and how is it aggregated across the entire end-to-end lifecycle.

3 Summary

Despite extensive design processes, emergent behavior will still appear at runtime in dependable automotive systems. A holistic diagnostic-advisor can address this problem in two ways. First, by synthesizing data across design phases, the advisor can identify dependencies and interactions that would have otherwise been undetected. Second, by coherently characterizing the expected behavior of the system, the diagnostic advisor will provide a more robust means of detecting and diagnosing emergent behavior as it occurs. Information learned from each occurrence will be fed back into the system in order to prevent or predict future similar behavior. As the diagnostic advisor operates, the whole will become greater than the sum of its parts.

References

[1] H.-G. Frischkorn, "Automotive software - the silent revolution." Automotive Software Workshop, San Diego, CA, Jan 2004.

[2] D. Wilson, "Ray of hope for auto industry," *Electronic Business*, Nov 2006. Online. Available at http://www.edn.com/article/CA6385672.html.