Automated Model Revision for Automotive Cyber Physical Systems*

Sandeep S. Kulkarni

1 Introduction

In this white paper, we argue that one of the promising applications for enhancing assurance of automotive cyber physical systems lies in the domain of *model revision*. Model revision attempts to achieve the middle ground between model checking and model synthesis. Specifically, the problem of model checking focuses on verifying whether a given model satisfies a given property. Hence, it either declares that the model indeed satisfies the property at hand or identifies a counterexample. Generally, the issue of how to fix the model (or property) is not addressed. On the other hand, model synthesis focuses on constructing a model that satisfies the given property of interest. While model synthesis is desirable as it generates a *correct* model, it is generally very computationally expensive and can result in inefficient models. Model revision attempts to provide the assurance provided by model synthesis while reducing its disadvantages. In particular, model revision focuses on revising an existing model so that it satisfies the property of interest. Thus, the goal of model revision is to begin with an *existing model* and *revise* it so that it satisfies a new property. The constructed model is correct-by-construction thereby providing the advantages of model synthesis. Moreover, because the revised model is based upon the existing model (that could be designed manually and optimized according to application needs), the revised model typically preserves the efficiency properties of the existing model. Moreover, model revision complements model checking by addressing the issue of what happens when the given model fails to satisfy the given property.

Model revision is especially useful when existing programs need to be revised due to bug fixes, revised requirements, etc. Moreover, in this case, it is necessary that no new bugs be introduced, i.e., it is required that the existing specification continues to be satisfied. Model revision is also desirable when we want to revise only a component in the system while leaving other components unchanged. By ensuring that the revised component continues to satisfy the existing specification, we can ensure that other components (e.g., those designed by different teams and for which source may not be available) are not affected. Furthermore, since model revision focuses on preserving existing specification as well as satisfy the new specification, it can potentially be used where the details of the original specification are not completely available.

2 Promising Applications

To illustrate the use of model revision in the context of automotive cyber physical systems, we begin with the recent recall of about 4 million vehicles by Toyota due to sticky accelerator paddles. While the exact technical details of this are not available to us, one suggested *solution* to fix this problem that was considered included that of brake override. We utilize this example to illustrate the role model revision can play in these scenarios.

The engine control module (ECM) is one of the crucial part in a car (cf. Figure 1). It takes inputs from several subsystems including the accelerator (to determine how much pressure the driver has applied on the accelerator), air mass sensor (for air quality information), and fuel injection subsystem. In turn, it controls the throttle position thereby affecting the speed of the car. Thus, a sticky accelerator

*Department of Computer Science and Engineering, Michigan State University, East Lansing MI 48824, sandeep@cse.msu.edu, Phone 1 517 355 2387
September 15, 2010

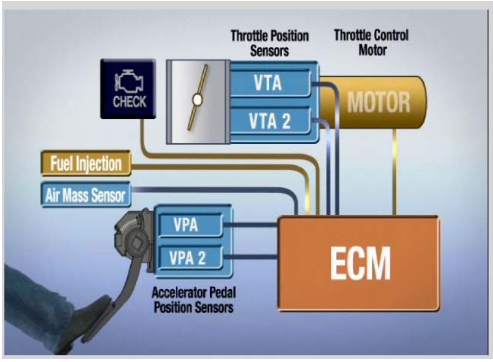

Figure 1: ECM Module (from http://toyota.com)

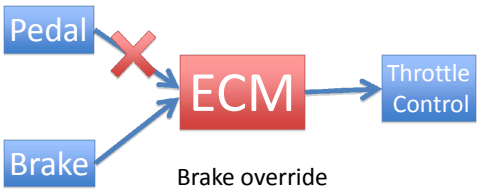

Figure 2: Brake Override

can cause the throttle to be in 'open' position thereby continuously increasing the car speed. One solution for dealing with this fault is to use *fault-prevention*; an illustration of such a solution was to add a shim to increase the tensions on the springs that would prevent the accelerator from sticking. Another solution is *fault-tolerance*; an illustration of such a solution is to revise the ECM to add a brake override (cf. Figure 2) so that if the brake is ever pressed while the accelerator had been pressed already (and some other conditions are satisfied) then the brake would take precedent over the accelerator thereby making the car stop.

As we can see from the above description, the above problem and solution requires us to modify an existing model (namely ECM) to satisfy new constraints (namely, if the brake is pressed then accelerator input should be ignored). Moreover, since this module is critical in safe operation of the car, it is important that the revised module should still satisfy the existing specification. Thus, it is straightforward to observe that model revision can be highly valuable in this context to provide assurance during the revision process.

Another instance where model revision will be useful involves problems identified in hardware that require a recall and a replacement of the hardware. However, replacing the hardware on all affected cars is potentially expensive, especially if for most vehicles, the hardware is unlikely to fail during the lifetime of the vehicle. Hence, one can utilize model revision in this context; specifically, we could revise the existing software so that any potential issues in the hardware are flagged immediately. This model revision could assist in reducing the overall cost by only replacing the hardware that shows signs of degradation.

While model revision in the above two cases is for safety reasons, we believe that such maintenance would also be desirable for improved functionality. As an illustration, consider the example of a driverless car that is assisted by sensors on the side of the road. As new sensors are developed, it may be necessary to modify the software in the car to utilize them. Specifically, in these scenarios, we expect that the modification involved by adding the new sensors could be done manually. However, the desired properties while using these sensors could be achieved using automated techniques.

In addition to the above scenarios, if model revision is cost efficient and provides assurance then it opens up several new possibilities (including possible revenue streams) where existing hardware is re-tasked based on user requirements. Examples of such scenarios include improvements to auxiliary systems (e.g., entertainment systems) or upgrading a smart cruise control to a smarter cruise control. These scenarios differ from the previous ones in that they are optional and potentially value-added and not safety-critical.

Current State of Art. Based on our current experience, we believe that such automated program maintenance for automotive CPS is feasible in near future. In particular, in our current work, we have shown the feasibility of automated program revision for adding different properties (timing constraints, fault tolerance, safety properties, etc.) to existing programs. Examples of programs (mod-

els) that we have been revised include those for an **altitude switch controller**, **a cruise controller**, as well as several problems in the context of parallel and distributed computing. We have also shown the feasibility of revising *moderately-sized models* by demonstrating the feasibility of applying revision successfully for state space larger than 10^{100}.

We have also demonstrated that model revision also has the potential to find *missing specifications.* In particular, when applied in the context of fault-tolerance, our algorithms for model revision provide maximum alternatives for recovery as long as they do not violate timing constraints and/or safety properties. Hence, analysis of these models has been found to be useful in identifying specifications that are missing from the original specification. Identifying these missing specifications early is crucial to reducing the cost of software development.

We have also shown the feasibility of applying model revision to SCR specifications (developed by Naval Research Laboratory). This work has enabled one to modify existing SCR specifications to add new fault-tolerance properties. We have also developed preliminary results for revising UML models.

3 Technical Challenges

To realize the potential of the automotive cyber physical systems, we would need to combine techniques from model-based design, model checking and existing algorithms for model revision. In particular, model based design would simplify the initial design of cyber physical systems, model checking would provide assurance about it and model revision would allow one to revise the model to meet new properties that are added at a later stage. However, to apply model revision for automotive cyber physical systems, one needs to overcome new technical challenges. In particular, for this, we need to develop techniques for modeling different characteristics of these systems, identify the effect of different characteristics in terms of the complexity during model revision, develop efficient algorithms for model revision, and utilize them in building tools.

Specifically, since cyber physical systems interact with an environment, they have to meet timing constraints, typically of the form 'time for response is within a lower and/or upper bound of an event in the physical world'. In other words, it is necessary that a given task not only provides the correct output but it is achieved within a given time interval. New algorithms and tools need to be developed to represent and add such property during revision.

Also, it is necessary to develop modeling techniques that will allow us to capture the behavior and constraints of the physical subsystem. While some of this modeling is required even for verification, more detailed modeling is required for synthesis and the way we model the physical world has a significant potential to change the complexity of synthesis. In particular, we need to address how concurrency provided by hardware as well as constraints imposed by it can be utilized during revision.

With these advances as well as efforts to promote use of model-based design and assurance based on formal methods, we expect that it would be possible successfully revise models and generate code from them that can be deployed.

Biography

Sandeep Kulkarni received his B.Tech. in Computer Science and Engineering from Indian Institute of Technology, Mumbai, India in 1993. He received his MS and Ph.D. degrees in Computer and Information Science from Ohio State University, Columbus, Ohio, USA in 1994 and 1999 respectively. He has been working at Michigan State University, East Lansing, US A since August 1999, where he is currently an associate professor. He is a member of the Software Engineering and Network Systems (SENS) Laboratory. He is a recipient of the NSF CAREER award. His research interests include fault-tolerance, distributed systems, group communication, security, self-stabilization, compositional design and automated synthesis. Some of the above challenges are being addressed with support from NSF and AFOSR.