Automated Program Maintenance for Automotive CPS*

Sandeep S. Kulkarni Borzoo Bonakdarpour

March 10, 2008

1 Introduction

With advances in automotive, computing and networking technology, it would be possible to provide several services –some envisioned already such as driverless car and some not yet anticipated. Such automotive cyber physical system would integrate computing, communication, and provide capability to monitor (e.g., speed, road condition, etc.) and modify entities in the physical world. One of the important issues in such systems is maintenance. The focus of this paper is on *providing assurance* in software maintenance due to changes in requirements, changes or faults in hardware, bugs, and so on. We identify different requirements and technical challenges in these automotive CPS and identify possible milestones that can be reached in the near future.

2 Requirements of Automotive Cyber-Physical Systems

Promising applications. One example for software maintenance comes from the recent recall of Ford Mustang due to a software problem in side front-airbags. In particular, this recall was caused due to internal testing that demonstrated a possibility of injury to a small passenger. Currently, such software maintenance is very expensive for the manufacturer and cumbersome to customers, as it requires the vehicle to be serviced at a dealer. The future research in automotive CPS has the potential to reduce the cost of such maintenance for both parties. With appropriate security mechanisms in place and the advances in vehicular networking, the required maintenance code could be delivered to the customer more easily, e.g., at a gas station.

Another instance where software maintenance may be necessary also includes potential hardware failures. We expect that in some cases such hardware could be monitored and replacement would occur only if the hardware degrades. However, for such an approach to work, it would be necessary to modify the software to include additional monitoring. Without such software, it may be necessary to change hardware in many automobiles although in most cases, that hardware would not fail in the lifetime of the vehicle. While software maintenance 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.

In addition to the above scenarios, if software maintenance for automotive CPS is made easy and cost-efficient 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

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

safety-critical.

Providing assurance during software maintenance. One requirement in such software maintenance is assurance that it would not violate existing properties of the system including safety properties. Especially when the maintenance is done due to identification of new faults, this requirement is especially important in that the maintenance may permit additional behaviors when faults occur. We have to guarantee that these additional behaviors meet safety properties and user expectations, some of which can be specified in terms timing constraints, status of different signals on the dashboard etc. Another requirement is security, i.e., ensuring that maintenance changes come from an authorized source and are performed at a time when it would not affect safety, e.g., when the engine is turned off. This position paper does not discuss the security aspect in detail but rather focuses on the assurance aspect only.

To provide assurance about the revised software, it is desirable to use automated techniques to ensure that the transformed program is correct by construction. Such automated methods for revising existing programs are desirable for engineers during design, development, and maintenance phases of the software life cycle. In an (automotive) CPS, since this would obviate the need for engineers/programmers to get manually involved in complex structures of the CPS and it would guarantee that the revised code would preserve existing properties in addition to the new desired properties.

Current state of the 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 (models) that we have been revised include those for an altitude switch controller, a cruise controller, agreement in the context of malicious entities, as well as several problems in the context of distributed computing. In fact, our experience suggests that there is a significant potential to find missing specifications by using our approach for program revision. In particular, our approach provides several possible paths for providing recovery in the presence of faults. However, it will not add recovery paths where some safety or timing requirement is violated. Thus, if a safety or timing requirement is missing then it would provide that as an option for adding recovery. Thus, the designer can identify missing specifications by focusing on undesirable recovery paths that may be added. Since it is well-known that such missing specifications can be very expensive in later stages of the software cycle, we expect that identifying them at an early stage can be highly beneficial especially in automotive CPS where they could cause extensive damage.

3 Challenges to interface/manipulate the Physical world

Research challenges. While the requirements identified in the previous section would enhance automotive CPS, there are several technical challenges that have to be met to realize them. In the context of automation, we need to model several constraints in the real world so that the revised program can be run efficiently in the physical world. In particular, it would be necessary to model the interaction of the software system with the underlying hardware, external environment and/or operating system. Specifically, consider a controller that manages a heating element inside an automotive CPS that requires that after the driver changes the temperate setting, the temperature inside a car is corrected within a given time. In such a system, the external factors such as the hardware and environment will determine the maximum rate of increase in temperature. This information needs to be used during automated revision so that the algorithms chosen for changing the airflow correspond to the constraints of the underlying hardware. Likewise, if it is possible to perform several tasks concurrently in the given system, that information needs to be modeled during program revision.

Another technical challenge is feature-interaction. In particular, it is anticipated that some program revisions may conflict with others. Especially when program revision is performed for *value-added* reasons (rather than safety reasons), users may control the features they want to include in their system. Therefore, it is necessary to identify methodology that will allow to track conflicts between different revisions so that users can utilize only those that are desirable. The use of automated program revisions can assist in identifying interaction between such features; in particular, if it turns out that automated revision is possible for feature f1 and feature f2, but it is not possible to automatically revise to have both features then it suggests that there is a conflict between f1 and f2.

Another technical challenge is that of complexity. Automated program revision is an expensive operation in terms of time and space complexity; we need to identify constraints under which the design of these systems is efficient and constraints that make the complexity high. This would allow one to identify those areas where automated program revision is likely to have the most impact.

Yet another challenge in CPS systems for such program revision is the need for multitolerance where the system tolerates multiple classes of faults and potentially provides differential level of services and assurance in their presence. While such multitolerant systems have been designed manually in many contexts, the use of automated techniques in them is still an open question.

4 Milestones

To realize the potential of the automotive CPS, we would need to combine techniques from model-based design, model verification/model checking and model revision. In particular, model based design would simplify the initial design of the CPS, model verification would provide assurance about it and model revision would allow one to revise the model to meet new properties that are added at a latter stage. We believe that model based design and verification would play a significant role in next five years allowing designers to hide the complexities of the CPS. In about 5-10 years, 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.

Borzoo Bonakdarpour received his B.Sc. in Computer Engineering from the University of Esfahan, Iran, in 1999. He received his MS in Computer Science and Engineering from Michigan State University in 2004. He is currently a Ph.D. candidate of Computer Science and Engineering at Michigan State University where he is a member of Software Engineering and Network Systems (SENS) Laboratory. His research interests include automated synthesis, revision, and verification of distributed and real-time systems.