Food for Thought™

An e-newsletter published by
Software Quality Consulting, Inc.

February 2010 , Vol. 7 No. 1
[Text-only Version]

Welcome to ***Food for Thought*™**, an e-newsletter from **Software Quality Consulting**. I've created free subscriptions for my valued business contacts. If you find this newsletter informative, I encourage you to continue reading. Feel free to pass this newsletter along to colleagues by clicking on the **Forward Email** link at the bottom of this email. If you've received this newsletter from a colleague and would like to subscribe, please click this **Enter New Subscription** link. If you don't wish to receive this newsletter, click the **SafeUnSubscribe**™ link at the bottom of this newsletter, and you won't be bothered again.

Your continued feedback on this newsletter is most welcome. Please send your comments and suggestions to **info@swqual.com**.

In this issue

In **This Months' Topic**, I discuss software quality issues in the automotive industry.

Regular features to look for each month are:

- **Monthly Morsels**
  Hints, tips, techniques and reference info related to this month's topic
- **Calendar**
  Conferences, workshops, and meetings of interest to software engineers, QA engineers and anyone interested in software development

This month's topic

Running On Code

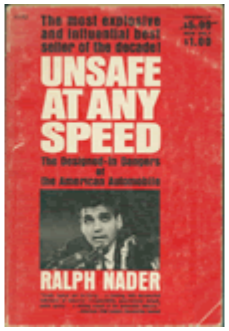
Fly-by-wire Controls Expose a Host of Automotive Software Quality Problems

"For over half a century the automobile has brought death, injury, and the most inestimable sorrow and deprivation to millions of people. With Medea-like intensity, this mass trauma began rising sharply four years ago, reflecting new and unexpected ravages by the motor vehicle. A 1959 Department of Commerce report projected that 51,000 persons would be killed by automobiles in 1975. That figure will probably be reached in 1965, a decade ahead of schedule." [1]

Ralph Nader's infamous attack on the US automotive industry was highly controversial when it was first published in 1965. Back then, most cars lacked basic safety features we take for granted today – things like seat belts, air bags, safety glass, collapsible steering columns, crumple zones, anti-lock brakes, child restraints, strong door locks, etc.

The automotive industry was arrogant back then (remember "planned obsolescence"?) and style was more important than safety. For example, many 1960's era cars had dashboards and instrument panels designed using shiny chrome parts and glossy paint, all which reflected sunlight directly into the drivers' eyes. Dashboards also were not padded and had protrusions, which often caused severe injuries.

For many years, the automotive industry resisted calls from within the industry as well as from outside the industry to design safer cars. It wasn't until Congress passed laws mandating basic safety features that the automotive industry relented.



Since Nader's book was published, mandatory safety features in cars have resulted in dramatic decreases in the fatality rate - highway deaths per 100 million vehicle miles travelled. In fact, for 2008, the fatality rate was at the lowest level since 1961. [7]

So why are so many people concerned about the safety of their cars now? In the 1960's, cars didn't have one more thing we take for granted today – **software**.

In my June 2007 e-newsletter, I reported that 2010 model year cars were expected to have upwards of **100 million lines of code** running on as many as **100 different microcomputers** (called Electronic Control Units or ECUs)all networked together. Today, projections are that cars will soon have 2-3 times that amount of code – **300 million lines of code!** [6]

By comparison, software that controls the **Space Shuttle** is less than a **half-million lines of code.** Even with the vagaries of counting lines of code, the difference between the amount of code in a car and the Space Shuttle is mind-boggling.

Lately, several problems with Toyotas have been in the news, including

- **Rapid acceleration**

  Rapid acceleration issues have been reported in 8 different Toyota models. Initially Toyota blamed floor mats, then sticking gas pedals. What has frustrated many owners is that Toyota and the National Highway Traffic Safety Administration (NHTSA) have known about this problem since at least 2002. [11] And the problem is not limited to Toyota. Other manufacturers such as Ford and Audi have had similar problems.

  John Liu, a professor of electronics and computer engineering at Wayne State University, has consulted on fly-by-wire technology for automakers and recently said:

> "Each electronic throttle control component determines the appropriate position based on signals from three or four sensors. That communication can be disrupted by signals from a nearby Blackberry, a microwave or radio transmission tower." [2]

Most engines today are fly-by-wire – that is, they are sensor-driven throttle systems controlled by software. Engineers believe these systems can be adversely affected by signals from cell phones or microwave towers.

Toyota has so far refused to acknowledge that software could be the real root cause.

- **Anti-lock brakes**

  Toyota just announced a recall of about 400,000 Prius and Lexus hybrid models. Under certain conditions, like an especially bumpy or slippery road, there may be a brief momentary delay in the brake response. The car will still stop but the distance required could increase slightly. This problem is due to a software defect and Toyota has begun installing a software fix on Prius models in Japan.

- **Electronic Power Steering**

  An analysis by *Automotive News* [8] found that the Corolla has been the subject of 83 power-steering complaints to NHTSA since April 2008. Seventy-six of those reports note that the vehicle unexpectedly veers to the left or right at 40 miles an hour and up.

  Read what a Corolla owner recently reported:

  > "[I] notice the steering wheel sometimes pulses only when my cell phone is docked to the right of the steering wheel. It's strange, I can sometimes tell if my Blackberry is going

> to ring or get an email. The steering wheel seems to shake or try to steer on its own. This is similar to my other 2009 Toyota Corolla that I resold to the dealer. I wonder if more shielding is needed to reduce any interference." [9]

There have been many automotive **software recalls** in the recent past:

- 2008 - Chrysler recalled 24,535 of its 2006 Jeep Commanders because of a problem in the automatic-transmission software.
- 2008 - Volkswagen recalled about 4,000 of its 2008 Passats and Passat Wagons and about 2,500 Tiguans for a problem in the engine-control-module software that could cause an unexpected increase in engine revolutions per minute when the A/C is turned on.
- 2008 - GM recalled 12,662 of its 2009 Cadillac CTS vehicles for a software problem within the passenger-sensing system that could disable the front passenger air bag when it should be enabled or enable it when it should be disabled.
- 2005 - Toyota recalled 160,000 Prius hybrids due to a software defect where the engine would suddenly turn off at highway speeds.
- 2004 - Mercedes-Benz faced the largest recall in its history for problems with its highly touted "Sensotronic" braking system, which relies on sensors to calculate the optimum brake pressure for each wheel. The German carmaker recalled 680,000 vehicles, saying bubbles in the system's hydraulic tank may cause braking failure.
- 2004 - Jaguar recalled 67,798 cars after discovering a defect in an electronic module that could inadvertently cause cars to slip into reverse gear.

- 2003 - A man was trapped inside his BMW for several hours after the on-board computer crashed. The door locks, power windows, and A/C were inoperable. Responders had to smash the windshield to get him out.
- 2002 - BMW recalled the 745i because the fuel pump would shut off if the gas tank was less than 1/3 full.

How did we get here?

The first production automotive microcomputer was a single-function controller used for electronic spark timing in the 1977 General Motors Oldsmobile Toronado. In 1978, GM offered an optional Trip Computer on the Cadillac Seville. The computer was a modified Motorola 6802 microprocessor chip and displayed speed, fuel, trip, and engine information.

By 1981, GM was using microprocessor-based engine controls executing about 50,000 lines of code across its entire domestic passenger car production. Other car companies quickly followed suit as automotive engineers realized that they could use software to measure and control engine functions in order to meet increasingly strict emissions and safety regulations.

Software soon found its way into audio systems. High-end audio companies like Bose now have more software engineers than audio engineers. And then came GPS navigation systems. Alfred Katzenbach, the director of IT management at Daimler, has reportedly said that

> "…the radio and navigation system in the current S-class Mercedes-Benz requires over 20 million lines of code alone and that the car contains nearly as many ECUs as the new Airbus A380 (excluding the plane's in-flight entertainment system). Software in cars is only going to grow in both amount and complexity." [6]

What can be done to improve automotive software safety?

Automotive software engineers need to adapt techniques successfully used to develop safety-critical software in other industries. For example, some of the techniques used to develop software for the Space Shuttle can be easily adapted to automotive software development:

- **Good requirements are essential to produce reliable software**

  To develop safety-critical software, we need to start with clearly written, unambiguous requirements.

- **Use historical data to predict your defect injection rate**

  Knowing your organization's defect injection rate is critical. You need this data to accurately predict the number of defects injected in every release. By subtracting the number of defects found, you can estimate the number not found.

  We know that the best, most highly skilled software developers inject on average 120 defects/KLOC or one defect for every 8 lines of code they write. [5] We also have anecdotal information suggesting that through peer reviews and testing, we typically find about 95% of the injected defects. The result is on average, released software has a defect density in the range of 5-6 defects per thousand lines of code (KLOC).

  So for a car that has **100 million lines of code** here's what we'd expect:

  - **Defects injected** using 1 defect/8 lines of code = ~12,000,000 defects

  - **Defects removed** assuming 95% found = 11,400,000 defects

    - **Defects remaining** (defects injected - defects removed) = **600,000**

  This means that there can be as many as **600,000 defects remaining** in the software running in our cars!

- **Development and Test need to be viewed as peers - each with an equal stake in the outcome**

  Developers and testers must be able to work cooperatively. Developers should be expected to deliver code that is as defect-free as humanly possible. Testers should be expected to find defects developers don't find.

  You should know about how many defects there are in a given release. And for mission critical software, you're not done until you've found as many of them as humanly possible.

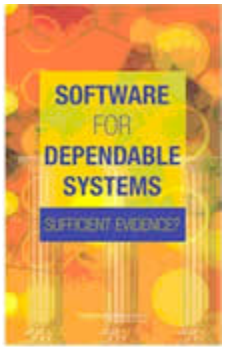
- **Blame the process and not people for failures and trust process to be self-correcting**

  People will always make mistakes. We need effective processes that help us find most of them and then help identify what aspects of the process need to be changed to ensure that more problems are detectable…

Researchers [4] have proposed approaches for improving software safety based on:

- **Make Explicit Safety Claims**

  "No system can be 'dependable' in all aspects and under all conditions. So to be useful, a claim of dependability must be explicit. It must articulate precisely the properties the system is expected to exhibit and the assumptions about the system's environment upon which the claim is contingent. The claim should also indicate explicitly the level of dependability claimed, preferably in



quantitative terms." [4]

- **Provide Evidence**

  "For a system to be regarded as dependable, concrete evidence must be presented that substantiates the dependability claim. Because testing alone is insufficient to establish properties, the [dependability] case will typically combine evidence from testing with evidence from analysis." [4]

- **Expertise**

  "Expertise - in software development, in the domain under consideration, and in the broader systems context, among other things - is necessary to achieve dependable systems." [4]

The researchers then identified the following **recommendations**: [4]

- Make the most of effective software development technologies and formal methods.
- Follow proven principles of software development - take a systems perspective and exploit simplicity.
- Make a dependability case for a given system and context: evidence, explicitness, and expertise.
- Demand more transparency, so that customers and users can make informed judgments about dependability.
- Make use of but do not rely solely on process and testing.
- Base certification on inspection and analysis of the dependability claim and the evidence offered in its support.

And this just in...

- Honda has just announced a recall of 640,000 cars to fix faulty power windows.

- Citroën and Peugeot are considering recalls since they use the same Toyota gas pedal, which is the subject of Toyota recalls, in some of their models.

It’s going to get a lot worse before it gets better. The current Toyota problems are just the proverbial tip of the iceberg…

The Bottom Line....

Have you had this experience?

> “Last year I bought a new car and was staggered to discover a 500-page manual explaining its operations, along with a 200-page companion manual for the GPS and radio systems. One of the new features touted was the much larger glove compartment, a size probably dictated by that of the required manuals.” [6]

If it takes over 700 pages to explain how to operate the features of your car, maybe it’s just too complex. We need to reduce complexity in order to improve safety. If we don’t change how we develop and test automotive software, then pretty soon Ralph can publish a sequel: **Software Unsafe at Any Speed.**

‘Til next time...

Monthly Morsels

Every month in this space, you’ll find additional information related to this month’s topic.

References

1. Nader, R. *Unsafe at Any Speed: The Designed-In Dangers of the American Automobile*, Grossman Publishers, New York LC # 65-16856, 1965.

2. Gardner, Greg, “Toyota’s Problem in Other Vehicles”, Detroit Free Press, posted Feb 1, 2010.

3. Robert L. Mitchell, “Toyota’s Lesson: Software can be Unsafe at

any Speed", *ComputerWorld* Blog, posted Feb 5, 2010.

4. Jackson, D., *et. al.*, *Software for Dependable Systems - Sufficient Evidence?*, National Research Council, National Academies Press, 2007.

5. Humphrey, W., "The Quality Attitude", *news@sei newsletter*, Number 3, 2004.

6. Robert Charrett, "This Car Runs on Code", *Discovery News*, posted Feb 5, 2010.

7. US DOT NHSTA Press Release, "Overall Traffic Fatalities Reach Record Low", July 2, 2009.

8. Neil Roland, "NHTSA fielding complaints about 2009-10 Toyota Corolla steering", *Automotive News*, posted Feb 9 2010.

9. Reilly Brennan, "Are Toyota Steering Problems Next?", Aol autos, posted Feb 9, 2010.

10. Julia Scheeres, "Teched-Out Cars Bug Drivers", posted on Wired.com on June 29, 2004.

11. Harry Stoffer, "Again NHTSA Probes Sudden Acceleration - March 22, 2004", *Automotive News*, March 22, 2004.

Calendar

Every month you'll find news here about local and national events that are of interest to the software community…

- **Software Quality Calendar**

  There are many organizations that sponsor monthly meetings, workshops, and conferences of interest to software professionals. **Find out what's happening…**

- **Workshops Offered by Software Quality Consulting**

  Software Quality Consulting offers workshops in many topics related to software process improvement. **Get more info...**

About SQC

Software Quality Consulting provides consulting, training, and auditing services tailored to meet the specific needs of clients. We help clients fine-tune their software development processes and improve the quality of their software products. The overall goal is to help clients achieve Predictable Software Development™ – so that organizations can consistently deliver quality software with promised features in the promised timeframe.

To learn more about how we can help your organization, **visit our web site** or **send us an email**.

I hope this newsletter has been informative and helpful. Your comments and feedback are most welcome. **Send me your feedback...**

Thanks,

Steve

Steve Rakitin

info@swqual.com

Software Quality Consulting Inc.

Steven R. Rakitin
President

- Consulting
- Training
- Auditing

Phone: 508.529.4282
Fax: 508.529.7799

www.swqual.com
info@swqual.com

Food for Thought, Predictable Software Development, Act Like a Customer, and ALAC are trademarks of Software Quality Consulting, Inc.
Copyright 2010. Software Quality Consulting, Inc. All rights reserved.

Graphic design by **Sarah Cole Design**.