On the Toyota UA Case and the Redefinition of Product Liability for Embedded Software

Roberto Bagnara

bugseng
http://bugseng.com

& Applied Formal Methods Laboratory
Department of Mathematics and Computer Science
University of Parma, Italy

12th Workshop on Automotive Software & Systems, Milan, October 30th, 2014

Outline

Acknowledgment

This presentation owes a lot to the following individuals and organizations:

- Michael Barr and Philip Koopman, who testified as software experts in the Oklahoma trial, wrote about it and their findings, and gave interesting presentations about all that
- NASA Engineering and Safety Center (NESC), who investigated the reported Toyota UA cases
- National Highway Traffic Safety Administration (NHTSA), who conducted several investigations on the UA cases
- colleagues of BUGSENG and BUGSENG North America
- Associated Press, The Wall Street Journal, CBS, ...

Several ideas, pieces of information and images were borrowed from the above: thanks!

From Last Year's Talk

Okla. jury: Toyota liable in acceleration crash

(By SEAN MURPHY / Associated Press / October 24, 2013)

OKLAHOMA CITY (AP) Toyota Motor Corp. is liable for a 2007 crash that left one woman dead and another seriously injured after a Camry suddenly accelerated, an Oklahoma jury decided Thursday.

The jury awarded $1.5 million in monetary damages to Jean Bookout, the driver of the car who was injured in the crash, and $1.5 million to the family of Barbara Schwarz, 70, who died.

It also decided Toyota acted with "reckless disregard" for the rights of others, a determination that sets up a second phase of the trial on punitive damages that is scheduled to begin Friday.

Before that Friday...

- ...Toyota reached a settlement to avoid punitive damages
- And shortly afterwards started an "intensive" settlement process (ISP) to try to resolve $\approx$ 300 more personal injury cases
- At that time (December 2013) Toyota:
  - had already spent $2 billion in legal costs
  - had recalled more than 10 million vehicles worldwide
  - had its executives subjected to congressional hearings
  - paid more than $65 million in fines for violating US vehicle safety laws
- In March 2014, Toyota reached a $1.2 billion settlement with the US DOJ, ending a criminal investigation into UA issues
- As of mid October 2014, Toyota has resolved more than half (145) of the consolidated injury lawsuits
- Many more remain: the final bill will be enormous

Extensive Damage To Public Reputation

By CBSNEWS / AP / *May 25, 2010, 7:08 PM*

Toyota "Unintended Acceleration" Has Killed 89

A 2005 Toyota Prius, which was in an accident, is seen at a police station in Harrison, New York, Wednesday, March 10, 2010. The driver of the Toyota Prius told police that the car accelerated on its own, then lurched down a driveway, across a road and into a stone wall. (AP Photo/Seth Wenig) / AP PHOTO/SETH WENIG

Unintended Acceleration

Definition of UA by NHTSA (February 2011)

"Unintended Acceleration" (UA) refers to the occurrence of any degree of acceleration that the vehicle driver did not purposely cause to occur

- Not a problem of Toyota alone of course
- Very recent announcements about UA issues caused by software bugs with some GM and Honda models
- UA is only one of the manifestations of software faults that endanger safety
- The level of awareness for such issues has increased dramatically after the Toyota case

Toyota Case Partial Timeline: 2000–2004

2000 Toyota adopts an electronic throttle control system (ETCS)

2003 The National Highway Traffic Safety Administration (NHTSA) conducts the first of many defect investigations regarding speed control problems

2003 Toyota internally deals with an "unwanted acceleration", not reported to NHTSA until 5 years later

2004 NHTSA opens an investigations into unwanted acceleration concerning Lexus Sedan, and 2002–2003 Camry and Solaris models

2004 Such investigations are closed claiming no defects were found; because of lack of resources, NHTSA turns down two more requests from owners

Toyota Case Partial Timeline: 2005–2007

2005 NHTSA conducts an evaluation of the Camry after reports of "inappropriate and uncontrollable vehicle accelerations"

2006 NHTSA receives hundreds of complaints about accelerator issues with the Camry models of 2002–2006

2006 NHTSA fails to identify the problem and closes the investigation citing "the need to best allocate limited administration resources"

2007 NHTSA launches probe into floor mats in Lexus models

Toyota Case Partial Timeline: 2007–2008

2007 NHTSA verifies a fatal crash link to floor mats: a person is killed when a Camry accelerating out of control hits his car at 120 mph (Camry's driver unable to slow the vehicle for 23 miles); Toyota settles with the family for an undisclosed amount

2007 NHTSA upgrades its investigation from "Preliminary Evaluation" to "Engineering Analysis"

2007 Under pressure from NHTSA, Toyota recalls 55,000 Camry and Lexus models because of suspected floor mats that interfere with the accelerator pedal

2008 NHTSA opens an investigation of 478 incidents where 2004–2008 Tacoma's engines allegedly sped up even when the accelerator pedal was not pushed

Toyota Case Partial Timeline: 2008–2009

2008 After eight months review, NHTSA closes the Tacoma investigation, claiming no defect was found (this is the eighth investigation of Toyota vehicles since 2003; over 2600 complaints have been reported, 271 of which were rejected by NHTSA without even asking Toyota for data)

2009 In April, NHTSA receives a petition to investigate throttle-control problems unrelated to floor-mat issues in the Lexus ES vehicles

Toyota Case Partial Timeline: 2009 (A Key Event)

2009 In August, the fatal crash of a Lexus ES350 in California kills four people

- Occupants were on the phone with 911 before and at the time of the crash: "We're in a Lexus. . . and we're going north on 125 and our accelerator is stuck. . . there's no brakes. . . we're approaching the intersection. . . . Hold on . . . hold on and pray . . . pray."
- The driver was a California Highway Patrol vehicle inspector with 20 years of experience
- NHTSA (too) quickly linked this incident to floor mats

2009 Between October and November Toyota recalls nearly 5 million vehicles related to the floor mat issue

Toyota Case Partial Timeline: 2009–2010

2009 It becomes clear that floor mats can only be blamed for some of the incidents

2010 In January, Toyota tells NHTSA that they may have "an issue" with sticking accelerator pedals, recalls about 3.4 million vehicles for sticking accelerator pedal, and halts production and stops selling eight models under pressure from NHTSA

2010 In February, Toyota admits problems with brake software and recalls 437,000 vehicles worldwide (2010 Prius, 2010 Lexus HS 250h, 2010 Camry Hybrids, Sai)

Toyota Case Partial Timeline: 2010

2010 In February, Toyota announces that Exponent Consulting had completed a 56 page report and sent to Congress a report that a test of Toyota's electronic throttle system behaved as intended

- Exponent's work was highly criticized by many independent experts: not neutral (blatant conflict of interest), technically very weak (inadequate testing)

2010 In February, the Wall Street Journal reveals that only one "reader" exists in the U.S. for Toyota's "black box" event recorder: it is located at Toyota's headquarters in California; moreover, Toyota claims that such unit is still experimental and can give false readings

Toyota Case Partial Timeline: 2010–2012

- 2010 In February the congressional hearings started
- 2010 In March new evidence emerged revealing that both Toyota and NHTSA knew (since August 2002) that reported sudden acceleration incidents were linked to a glitch in the vehicles' electronic system
- 2010 In March NASA Engineering and Safety Center (NESC), upon request from the DOT, agrees to conduct an assessment of Toyota's ETCS-i system
- 2011 In February, the NASA report was published
- 2012 For Toyota, more congressional hearings, fines and settlements

Toyota Case Partial Timeline: 2012–2014

2012 In September, new report "Analysis of Toyota ETCS-i System Hardware and Software" prepared by Exponent for Toyota

2013 Bookout/Schwarz v. Toyota: first trial in the US in which the plaintiffs allege that the UA was caused by a malfunction of the ETCS, as well as the lack of a brake override system that would have allowed the driver to slow or stop the vehicle

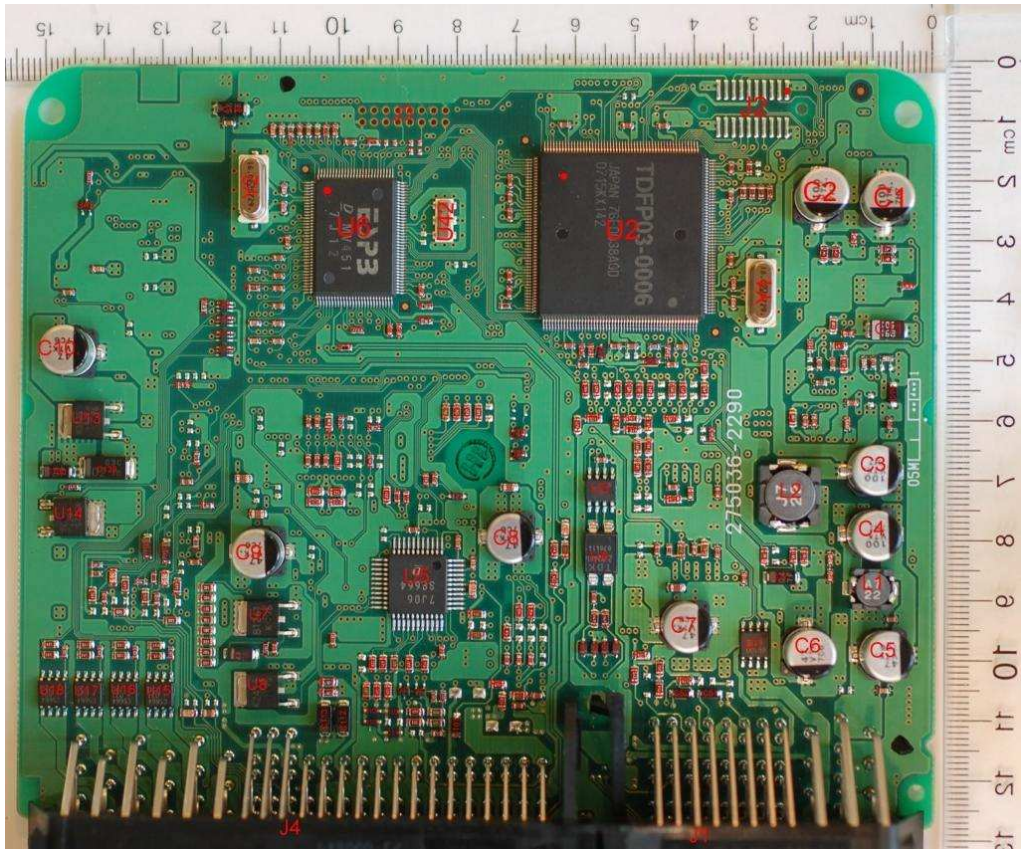
2013 Toyota launches the intensive settlement process

2014 Toyota reached a $1.2 billion settlement with the US DOJ: this is a fine for concealing safety defects covering floor mats and sticky pedals only

2014 Mass settlements. . . developing story. . .

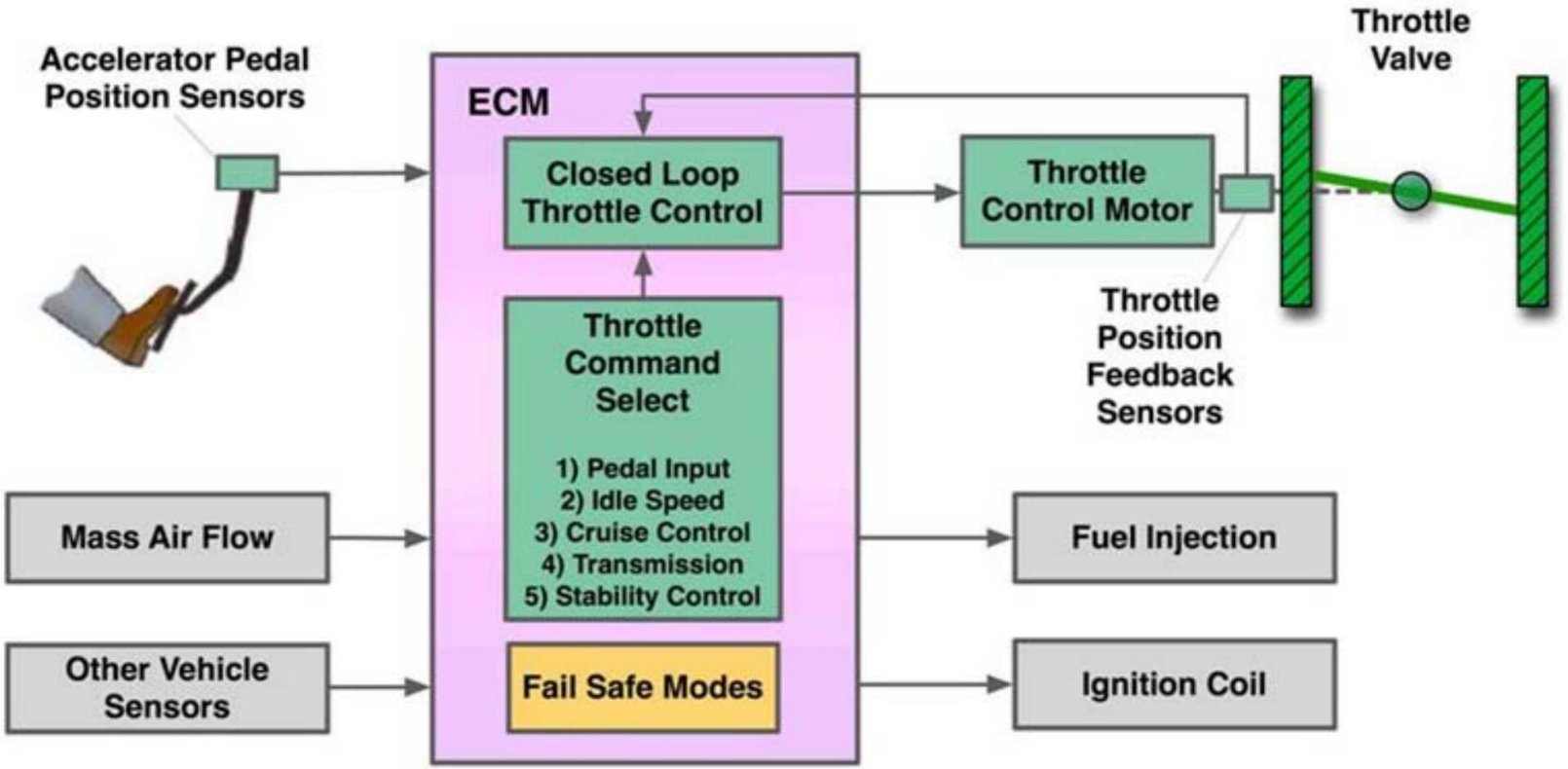
The Problem Is (More Likely than Not) Embedded

Central to the Oklahoma trial was the Engine Control Module's (ECM) firmware



The 2005 Camry ECM board

The ETCS-i System



- It mainly controls the throttle valve
  - Fuel injection and ignition are adjusted, taking into account several parameters, so as to ensure proper combustion
- It was first investigated by a NASA team in 2010–2011

The ETCS-i System Is Safety Critical!!!

Pumping brakes with throttle valve fully open causes loss of brake power assist (force required: 7–20 kg $\Longrightarrow$ 80 kg)

- As a result, braking, the most instinctive driver's response, may not allow the driver to stop the car
- In several cases, drivers burned the brakes without being able to stop the car

Drivers cannot be expected to take other countermeasures

- Shifting to neutral: driver might not think about it, or might think it could harm the engine, or it might not know how to shift to neutral when the vehicle is in modes other than drive and reverse
- Turning the vehicle off: driver might not think about it, or might (correctly!) think this would cause loss of power steering, or might be afraid the steering wheel will be locked

The NASA Investigation (2010–2011)

NASA worked under significant time pressure

- The study started mid April 2010 and ran through mid August 2010

They were given partial and misleading information

- Analysis is limited to the main CPU (V850) but the monitor chip (ESP-B2) also has software
- They were told that the static RAM is protected by EDAC, but it is not
- They were told mirroring of critical variables (to protect them from corruption) is always done, but this is not always done
- They were told that less than half of the available stack memory is used, but stack overflow can occur

The NASA Investigation (2010–2011)

NASA was unable to prove that the ETCS-i was responsible for the UA

- But did not exclude the possibility of an electronic-based cause for the UA
- Moreover, they found plenty of questionable things from the technical standpoint

The expert witnesses of the Oklahoma trial had more time to conduct their investigation and, building on the work of NASA, were more successful

The Oklahoma Trial Investigation (2013)

Embedded software system's expert witnesses:

Michael Barr studied the ETCS-i software in depth and found the likely cause for the UA

Philip Koopman studied the highly-confidential Toyota design documents for the ETCS-i and the reports produced by NESC, by Michael Barr and others who had had access to the code

They found many extremely serious problems, both with the system's design and with the software

In their reports there is a lot of crystal-clear evidence of inadequate engineering practice

Why Didn't Vehicle Testing Find the Problem?

Vehicle testing is of crucial importance

But it is impossible to test everything at the vehicle level

- Huge number of possible operating conditions
- Failures may depend on precise timing sequences
- Huge number of (possibly intermittent) faults
- Random memory corruption

In 2005–2010 Toyota reported approximately 35 million miles of vehicle level testing and 11 million hours module level software testing

Divided by the number of vehicles this means roughly 1–2 hours per vehicle

- There is no way this can find all failures
- Remaining testing will be, de facto, conducted by end users

Testing Is Not Enough

This is well known!

E. W. Dijkstra, *Notes On Structured Programming*, 1970

"Program testing can be used to show the presence of bugs, but never to show their absence!"

R. W. Butler & G.B. Finelli, *The Infeasibility of Quantifying the Reliability of Life-Critical Real-Time Software*, 1993

"[...] life-testing of ultrareliable software is infeasible (i.e., to quantify 10^{-8}/hour failure rate requires more than 10^{8} hours of testing) [...]"

Testing Is Not Enough: A Rigorous Process Is Needed

Development of a safety-critical system requires a rigorous process

Based on the concept of Safety Integrity Level

Today it would be ISO 26262 (2011)

At the time the ETCS-i was developed, it would be as defined in the MISRA Development Guidelines for Vehicle Based Software (1994)

ETCS-i is SIL 3 (not the highest level, but people may die)

MISRA Guidelines for SIL 3

Specification and design Formal specification for those functions at this level.

Languages and compilers A restricted subset of a standardized structured language. Validated or tested compilers (if available).

Configuration management: products Relationships between all software products. All tools.

Configuration management: processes Automated change and build control. Automated confirmation process.

Testing White box module testing — defined coverage. Stress testing against deadlock. Syntactic static analysis.

Verification and validation Automated static analysis. Proof (argument) of safety properties. Analysis for lack of deadlock. Justify test coverage. Show tests have been suitable.

Access for assessment Techniques, processes, tools. Witness testing. Adequate training. Code.

ETCS-i Main CPU Software

Table A.7-1. Basic Code Size Metrics Camry05 Software

C code	#Files	SLOC	NCSL	Comments	NCSL/ File	SLOC/ NCSL	Comments/NCSL
sources	1,761	463,473	256,647	241,683	145.7	1.8	0.9
headers	1,067	100,423	39,564	67,064	37.1	2.5	1.6

- For the development of ETCS-i, Toyota did not claim to have followed the MISRA guidelines
- In particular, Toyota did not claim enforcement of the MISRA C guidelines
- If there was an auditable software process plan, NASA did not disclose it
- If there was a written safety argument, NASA did not disclose it
- Toyota and Exponent (on behalf of Toyota) claimed that ETCS-i fail-safes would mitigate UA

The Need for Fail-Safes

Hardware bit flips due to radiation

- 1 HW fault every 10,000 to 100,000 hours per chip
- optimistically, 2% of them are dangerous: 0.2 times per million hours
- assuming cars are driven 1 hour/day
- only taking into account the Camry vehicles built in one year ($\approx 430,000$), this results into one dangerous fault every 11.6 days per chip

Software defects

- They can corrupt memory, any memory
- They compound badly with (unanticipated) HW faults

Moral: hardware and software faults will happen, all the time

[Credit: Philip Koopman]

Inadequacy of ETCS-i Fail-Safes

Investigations proved that ETCS-i fail-safes are "defective and inadequate"

- defective fault containment
- inadequate redundancy
- existence of single points of failure

For instance, there are two redundant copies of the accelerator position signals

- but they both go to the monitor chip for A/D conversion
- so, if this fails, the main chip can obtain invalid accelerator position data regardless of the signal redundancy

Coding Rule Violations Are Bug Predictors

Toyota claimed to use an internal coding style with a 50% overlap with MISRA-C:1998

In reality, out of the 127 rules of MISRA-C:1998, only 11 are in the Toyota coding guidelines

As NASA found out, Toyota did not always follow its own coding rules (e.g., many `switch` statements without a `default` clause)

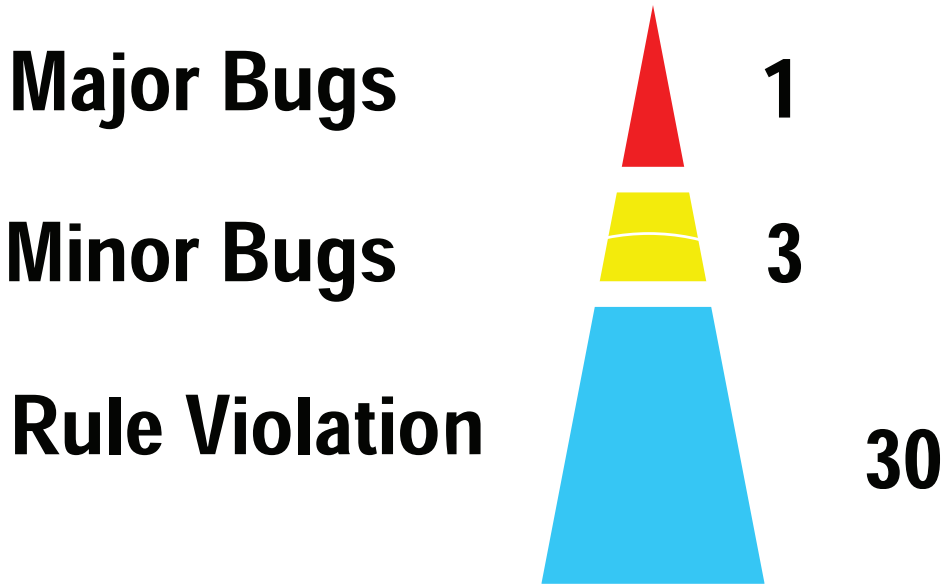
Of the 35 MISRA-C:1998 rules that NASA tools could check (in 2010, but still this is shocking), 14 were violated, for a total of 7,134 violations

The team of Michael Barr found 80,000 violations of MISRA-C:2004

Coding Rule Violations Are Bug Predictors

See, e.g., the works of Les Hatton...

...and of Toyota people as well!



Findings in Toyota infotainment software [Kawana et al., 2004]

NASA Static Analysis Results for ETCS-i

They include (modulo bugs in the tools):

- 2272 global variables declared with different types
- 33 unsafe type casts
- 99 conditions that contain side effects
- 22 definitely uninitialized variables
- 2 arrays of 16 bytes initialized with 17 bytes

Many sources of undefined behavior!

Spaghetti Code: The Control-Flow Flavor

McCabe/Harrison cyclomatic complexity is generally recognized as a good measure of control-flow complexity

- It directly measures the complexity of the control-flow graph
- It indirectly measures code testability and changeability
  - > 50 untestable
  - > 75 fixing one bug very likely will introduce another bug

In Toyota ETCS-i code:

- 67 functions have cyclomatic complexity over 50
- 1 of them (throttle angle function) has cyclomatic complexity 146

Spaghetti Code: The Data-Flow Flavor

Global variables undermine modularity

- Changes in one portion of the code will affect other portions of the code that interact (in often unclear ways) via the globals
- So the program can only be reasoned upon globally
- Same as if the program uses gotos

For the Toyota ETCS-i code, NASA found that "In the Camry software a majority of all data objects (82%) is declared with unlimited scope and accessible to all executing tasks."

$\approx 10,000$ global variables

$6,971$ could be local static

$1,086$ could be file static

Concurrency and Watchdog Timers Issues

Several concurrency issues were identified, including:

- shared global variables not declared `volatile`
- shared global variables not always accessed with interrupts masked

Watchdog timers are supposed to detect task deaths and trigger the appropriate recovery measure, but in ETCS-i:

- RTOS error codes (such as task death) are ignored
- watchdog is tickled by a hardware timer service routine which defeats its whole purpose
- watchdog cannot detect the death of major tasks

Recursion

This is forbidden by all coding standards related to safety

- It makes determining the worst-case stack usage much more difficult
- Stack overflow is a very serious error

In Toyota ETCS-i code

- Recursion is used (so their use of Green Hills Software `gstack` tool to determine the worst-case stack usage was outside the specification of the tool: they allocated 4096 bytes instead of `gstack`'s reported maximum possible stack size of 1688 bytes)
- Unfortunately, stack space is 94% full without any recursion (despite what they told NASA)
- No mitigation for stack overflow: stack overflow directly results into overwriting OSEK RTOS areas

Recursion, Stack Overflow, Task Death, UA

According to the Oklahoma trial investigation, this is what, more likely than not, happened:

- recursion resulted in a stack overflow
- such an overflow overwrote a critical OSEK RTOS area
- this overwriting resulted into the death of "Task X", which includes throttle angle computation and most fail-safes
- death of Task X was not detected by the watchdog
- wide open throttle and unintended acceleration

How Is the Toyota UA Case Special?

Never before (as far as I know) has the software for some automotive equipment been so carefully scrutinized...
...and the findings of such scrutiny so decisive in the trial
Never before (as far as I know) has a jury been lectured so precisely about things such as:

- random hardware faults
- bugs in software
- insufficient safety measures
- sane embedded system programming principles
- the MISRA guidelines
- the importance of coding rules
- the role of software metrics (including violation counts) as bug predictors

Note that in the US car makers were not, and are not required by law to follow the MISRA guidelines

Mind the Lawyers!

On YouTube you can watch lawyers teaching other lawyers how to find "spaghetti in the code"

http://www.youtube.com/watch?v=LE7Xxs3g4yU

More Cases Like Toyota UA To Come

Safety group says Chevy Impala air-bag defect linked to 143 deaths, including Texas crash

Jeff Green and Jeff Plungis
Bloomberg News
Published: 07 April 2014 09:17 AM
Updated: 07 April 2014 02:32 PM

General Motors Co., in the midst of recalling 2.6 million small cars for an ignition-switch flaw that can deactivate air bags, also may have an air-bag defect connected to deadly accidents in its Chevrolet Impala, a safety group said.

The Center for Auto Safety, in a letter to U.S. regulators today, cited a government petition by a former GM researcher who said he found a software fault that can misread a passenger's weight and render frontal air bags inoperative. The consultant, Donald Friedman, is asking the U.S. National Highway Traffic Safety Administration to open a defect investigation into 2003-2010 model-year Impalas.

There have been at least 143 fatalities in frontal crashes when an Impala's air bag didn't deploy, Friedman said, citing data collected from NHTSA's fatal-crash database. In 98 of those cases, occupants who died were wearing seat belts.

"This is a design defect in every GM vehicle with the flawed algorithm" in the software, said Clarence Ditlow, executive director of the Washington-based Center for Auto Safety, which has been tracking recalls and defects since it was founded in 1970.

The Attitude

Things that are said way too often:

- "Yes, we do the MISRA-C rules: the compiler has some warnings for them."
- "What?! 3,000 violations? El Cheapo gives none!"
- "Come on, undefined behavior in C cannot be what you pretend it to be!"
- "I don't care if C standard says it is undefined behavior: I've verified from assembly that my compiler does what I expect."
- "No, I don't care whether these are true or false positives: just silence them all."
- "Training, what training? Our engineers are very good!"
- "This part of code has already been used in a former project: it is OK as it is!"

Snake Oil Being Swallowed, Hook, Line and Sinker

Table 10. 2002 V6 Camry primary throttle modules code verification summary

Code Verification	
PolySpace Verifier	Enabled
Number of Result Sets	x 1
Reds	0
Grays	101
Oranges	147
Green	3654
Proven	96.2%
Pass/Fail	

Table 10 shows the following:

- 96.26 % of the code in the analyzed modules does not contain any run-time errors at the language level.

From "Analysis of Toyota ETCS-i System Hardware and Software", Exponent, September 2012, page 195

The Policies

1. A non verifiable piece of software is a broken piece software (*no matter who swears it is correct*)
2. A verification tool that hides what it cannot prove is a broken tool (*look mom, my code is 100% compliant!*)
3. (Beware: out of the speaker field of competence!)
   Several car makers do not have the software embedded in the devices supplied to them
   - OK, they do not own it, this makes sense
   - But not being able to examine it is a completely different story

Conclusion

- The October 2013, Oklahoma trial marked a turning point in the definition of product liability for embedded software, in particular in the automotive sector
- Never before has the source of embedded software been scrutinized so carefully, with findings crucial in the jury decision
- Things will never be the same as before
  - If not for other reasons, because lawyers now know what to do and how to do it
- The automotive industry should increase the efforts in the improvement of software and systems processes
  - Sticking to the true state-of-the-art in process, design, development, tools
  - The real thing: not just a *pro forma*, minimal-requirements adoption of some best practices

The End


Questions?
roberto.bagnara@bugseng.com
bagnara@cs.unipr.it
bugseng