IN THE DISTRICT COURT OF OKLAHOMA COUNTY
STATE OF OKLAHOMA

Jean Bookout; Charles Schwarz, individually and as Personal Representative of the Estate of Barbara Schwarz, deceased; Richard Forrester Brandt, as Personal Representative of the Estate of Barbara Schwarz, deceased, Plaintiffs, vs. Toyota Motor Corporation; Toyota Motor Sales, U.S.A., Inc.; Toyota Motor Engineering and Manufacturing North America, Inc.; Aisan Industry Co., Ltd., Defendants.	)	Case No. CJ-2008-7969

* * * * *

TRANSCRIPT OF MORNING TRIAL PROCEEDINGS

HAD ON THE 14TH DAY OF OCTOBER, 2013

BEFORE THE HONORABLE PATRICIA G. PARRISH,

DISTRICT JUDGE

Reported by: Karen Twyford, RPR

******* THIS TRANSCRIPT HAS NOT BEEN PROOFREAD *******

APPEARANCES

For the Plaintiffs:

Mr. Benjamin E. Baker, Jr., Attorney at Law
Mr. R. Graham Esdale, Jr., Attorney at Law
Mr. J. Cole Portis, Attorney at Law
Mr. Jere Beasley, Attorney at Law
Beasley, Allen, Crow, Methvin, Portis & Miles, P.C.
218 Commerce Street
Montgomery, Alabama 36104

Mr. Larry A. Tawwater, Attorney at Law
The Tawwater Law Firm, PLLC
14001 Quail Springs Parkway
Oklahoma City, Oklahoma 73134

For the Defendants:

Mr. J. Randolph Bibb, Jr., Attorney at Law
Mr. Ryan N. Clark, Attorney at Law
Lewis, King, Krieg & Waldrop, P.C.
424 Church Street, Suite 2500
Nashville, TN 37219

Mr. James A. Jennings, Attorney at Law
Mr. J. Derrick Teague, Attorney at Law
Jennings Cook & Teague
204 N. Robinson, Suite 1000
Oklahoma City, Oklahoma 73102

THE COURT: We're in recess for 15 minutes.

All rise while the jury exits.

(Whereupon, a short recess was had.)

THE COURT: We're back on the record. Members of the jury are present as well as counsel and their clients.

Mr. Baker, you can call your next witness.

MR. BAKER: Your Honor, at this time we call Michael Barr.

THE COURT: Raise your right hand, please.

(Witness sworn.)

MICHAEL BARR,

called as a witness, after having been first duly sworn, testified as follows:

DIRECT EXAMINATION

BY MR. BAKER:

Q Tell us your name, please.
A Certainly. I'm Michael Barr.
Q Where do you live?
A I live in Maryland, near Baltimore.
Q And are you married?
A I am.
Q And do you have any children?
A I have two boys, six and ten.
Q How old are you?
A Forty-two.

Q And could you tell us what you do for a living?
A I'm an embedded software expert.
Q What does that mean?
A That is the question everybody always asks. Well, I will have get to what embedded software is in a minute, but let me tell you a little bit about my background. I have studied electrical engineering; that is what my degrees are in. I have two of them, both from the University of Maryland, a bachelor's degree and a master's degree. Along the way, earning my electrical engineering degree, I also studied software.
Q Let me stop you there. Pull the microphone a little closer. We're having trouble hearing you. As with Dr. Koopman, slow down a little bit.
A Sure thing.
Q You were talking about your software experience.
A Yes. So I actually started programming when I was about 12. I grew up in a house where we had some of the early personal computers like Apple II and before that one from Texas Instruments. So I became interested in programming. And all throughout my education in electrical engineering, which really focuses on the design of circuits and chips, circuit boards and other electrical aspects, I was also studying software programming, so I have been programming for about 30 years.

Q Who do you work for?

A I am co-owner of a company called the Barr Group. I have a partner who runs the business. I'm the chief technical officer of the company, so I oversee our technical activities.

Q What is it that the Barr Group does?

A The Barr Group helps companies that make embedded systems. We will get to what they are, I promise you. We help to them make them more reliable and also more secure. So we help all kinds of different companies and a lot of different industries. We help companies who make -- I myself have worked on receivers for Direct TV. So if you have a satellite dish or a cable box in your house, I have worked on a product like that.

I have also worked on products that are industrial control systems that are used, for example, in a factory to do manufacturing. I have consulted with companies and have been involved with the design of a number of medical devices, both medical devices that are used in treating the patients, and also those like pacemakers that could injure someone if they malfunction.

And the Barr Group has a number of clients in a range of industries like that, so industrial controls, consumer electronics, medical devices, et cetera.

Q And I know we have got a slide up here with your background on it. Have you put us a PowerPoint slide together to help demonstrate some of the testimony you will give us today?

A I have.

Q All right. I know you mentioned a couple of your degrees. Can you go back and tell us when you received those degrees, please.

A Yes. My bachelor's degree was 1994, and my master's degree was 1997.

Q In terms of the Barr Group where you work now, when did you start that company?

A The Barr Group was founded about two years ago, but it came out of another company that was founded in 1999 called Neutrino (phonetic.)

Q For the jury's benefit, can you give us a little bit of your work and background before you started the Barr Group.

A Sure. When I finished my bachelor's degree, I went to work for a company that developed a lot of the telecommunication systems. The company was called Hughes Network Systems, and they made everything from satellite receivers for point-of-sale equipment. Like, a gas station in a remote area would receive and upload its pricing information and sales records through a small satellite terminal, and also base station equipment that is used in cellular base station, when you pick up your cell phone and it connects to the tower, we made equipment for the tower.

I was involved with a project there that was called an air phone. It was one of the first telephones on an airplane, so we were involved in enabling that system, and I worked on one of those products there. After that, I finished my master's degree, and I went to work for a company that had spun out of NASA.

NASA has a green belt location just outside of Washington, DC, and some engineers had left there and formed a company to work on satellite ground station equipment to communicate with satellites. And I worked there for my next job. And pretty much after that, I started consulting and founded the Neutrino company.

Q After your work with the group that came out of NASA, is there anything else that you did before working with the Barr Group?

A That was the foundation of Neutrino in 1999.

Q Looking here at your slide with your background and experience, it mentions that you have three patents. What do those patents involve?

A I'm a named inventor on three patents, I'm not the only inventor on any of them. Those are related to my work with various companies that I have consulted with. So in one instance, the first patent was related to a piece of physical therapy equipment which is like a -- kind of like a piece of gym equipment, but it is more important that it be safe because it is usually helping someone who is injured to recover a muscle injury or something like that doing repeated twisting motions or lifting motions and things of that sort. So one of those is related to -- not all of them come off the factory floor identical because of mechanical difference and that is related to the calibration to make sure they all behave the same way through the software.

Q Now, I know you discussed some of your work in terms of your consulting work, but you mentioned up here that you have done specific consulting and training in embedded software process and architecture for reliability.

Can you explain to us what embedded software process would mean in that context.

A Sure. I think Dr. Koopman spoke at length about process for safety critical system design, and he talked about some of the international standard safety processes like MISRA. And I think he talked about 61508, which is an international standard not specific to automotive.

So software process relates to how the software is specified and built. And there is -- that is the process. The architecture consulting relates to once you decided what you want to build and that you're going to follow a coding standard and do those other things to ensure that the process is in place, architecture relates to the design of the software at a high level.

Before you get down to the individual line of code, how do you structure things, and that is the architecture. It is kind of like the architecture of a building. In the architecture of a building, they're not necessarily concerned with who is in what office and how it is decorated, they're concerned with how many bathrooms there are, how many floors there are, what the supports are.

Q It also mentions here reference to you served as editor and a columnist and a conference chair. Can you tell us about that.

A Yes. For about 3 1/2 years I served as editor in chief of an industry publication with about 60,000 embedded software engineers as readers. Believe it not, there is that many of them. And the magazine focused on good practices for designing embedded software. And our readers were our authors, so I was serving in a selection role selecting the best articles, the best techniques, and making sure that they got published.

Q Within that role, would that have been the time that you published some of the 65 articles and papers that we see here?

A I started writing articles before I did that. In fact, that's how I ended up getting involved in that. My first article was published in 1997. And then I published articles and columns during that time, during those 3 1/2 years, but I continued to do so right up to the present day and other publications as well.

Q And you reference three books, and we have a picture of those three books?

A We do.

Q Can we see that, please. When was this first book, *Programing Embedded Systems* published?

A This first book was published in -- the copyright date is 1999, but it came out late 1998. And that book was actually very popular. It is a book that introduces new engineers and programmers to the aspects of programming that that are specific to designing embedded systems. So it was sold in tens of thousands of copies. It was -- I have up here a picture of the Japanese cover. So around 2000 or 2001, this book was translated into Japanese, Taiwanese, Chinese, and Korean.

Then later in 2006, another author came along and made a second edition of it, and I served more as an editorial role at that time.

Q What is the next book?

A The next book is called *The Embedded Systems Dictionary*. I wrote that book in 2003 with another industry experts who had been a columnist and a contributor to the magazine that I was editor in chief of, and it defined about 3,000 basically engineering terms that people use our in our industry, in the embedded system space, provided concise definitions of them so that we could all -- many of them we did have a common understanding, but there were certainly some where we didn't, so we tried to rectify the language and clarify some things. That was published in 2003.

Q What about the last book?

A The last book was published in 2008, and that was called *The Embedded C-Coding Standard*. There has been a second edition of it in 2012. And you heard about MISRA-C, and I will also talk a little bit about MISRA-C today. This is not a replacement for MISRA-C. There are some embedded programs that are not safety critical, and can use this standard, which is designed specifically to keep bugs out of systems and has some overlap with MISRA-C but is a lighter-weight version, if you will.

It is also complimentary with MISRA-C in that MISRA- C is silent about style. It is more about rules that you should use to make your program safer, and this is both some of those safer rules and also stylistic rules to make your programs more readable and easier to obtain.

Q All right. I will back up just a minute. You talked about your consulting work and the things that you do with Barr Group. As part of your consulting work, have you from

time to time done exactly what you're doing here today, acting as an expert related to software and embedded systems?
A I have.
Q What sort of things have you done in terms of that type of consulting?
A Probably the most common engagement I've been involved with is patent disputes. So I've worked on patents related to smart phones, set-top boxes like the Direct TV receivers. Sometimes there are disputes between those who patent an idea and those who make a product about whether there is an and infringement between the two. And I often get involved in looking at the source code for the product that the accused to see if it infringes the patent or not.
Q You just mentioned the word source code. And I know we will talk about it a lot today. Can you go ahead and tell us what source code means.
A Yes. I have a example of it coming up, but the source code is just simply for now the human readable part of a software program. So there is the human readable part that the programmers write and maintain, and then there is the nonhuman readable binary part or version that the computer understands.
And there are tools called compilers and things of that nature that convert the human readable into the machine

readable part.

Q Does the source code, I guess from my layperson's view, the instructions that have been written by a human that the computer reads so it knows what to do that?

A That is a good general explanation of it. Yes. The source code is what the humans write to tell the computer what to do.

Q Now, you have been retained in this case to look specifically at certain aspects. Can you tell us what you were asked to do in this case.

A Yes. So I have reviewed the source code for the engine control module in the 2005 Camry vehicle that was driven that day. And also in the -- I reviewed the facts of the incident in terms of what happened. And then I have expressed opinions with respect to the software and with respect to the incident as it relates to the software.

Q So you were asked to look at the software and determine whether it worked or not in this vehicle?

A That's correct.

Q And you mentioned looking at several things. In the information you've looked at, have you looked at depositions? The jury has heard about depositions. Have you looked at depositions?

A I have.

Q Have you looked at what I call fact witness depositions from people who saw or witnessed things related to the wreck?

A I have.

Q There are a number of experts, jury already heard from a few of them. Have you looked at expert depositions?

A I have.

Q There have been -- there has been some testimony about Toyota documents. Have you looked Toyota documents that have been produced?

A A lot. A lot of Toyota documents. Yes.

Q There is a bunch of boxes back here. Have you looked at enough documents to fill many, many boxes?

A I've had access to probably more pages of documents, but many of them were produced electronically, so I don't know how big they would be when printed. But I imagine it would be larger than that.

Q Have you used those as part of your analysis to render opinions in this case?

A Yes, I have.

Q Also, as part of your analysis in this case, have you reviewed sworn testimony of people who claim to have also had unintended acceleration events?

A I have.

Q And have you used that to help you analyze the facts in this case?

A Yes, I have.
Q As part of your review in this case -- and let me step back -- this is not the first Toyota UA case that you have been involved with, correct?
A That is correct.
Q Have you written reports related to those other cases?
A Yes.
Q And in a general context, have you also written a report that embodies much of your analysis of Toyota software or source code?
A I have.
Q Does it encompass 13 chapters?
A Yes. It consists of a summary report and 13 chapters of detail.
Q Is this the approximately 800 pages worth of analysis that you have done related to Toyota software?
A That's right.
Q All right. What I would like to do now is move on to your analysis and talk about some of the terms that we will be hearing about, okay?
A So embedded systems is probably something you're wondering about, it is all over my bio and things like that. Embedded systems are simply computers that you don't think of as computers, your microwave oven, this laser pointer,

the Nike fuel band that I wear as a watch and a pedometer. Those are all examples of embedded systems. Like it or not, the world is producing over 10 billion of these a year.

In fact, when you think of a computer and you think of a laptop or a desktop computer, that is about one or two percent of all the processers that are being made. A lot of less expensive processors are going into everything from these kinds of examples to satellites in the sky, your TV. That TV that is there has a computer inside it and software. So those always consist of the electronics, a processor and software.

Q And as these embedded systems, computer embedded software systems that you're trained and have experience in analyzing and writing?

A Yes.

Q Are these systems also included in cars?

A They are. They have been included in cars for quite a while. One of the early motivating reasons for including a computer in the car was related to emissions control. So putting a processor and software at the heart of the car in order to control the spark timing is something that has been done going back several decades now.

Q As we see on the slide, has it evolved to where it encompasses many, many functions that go on within an automobile?

A Absolutely. It was probably 2006 when I saw a BMW ad that said for the series 7 they said we have over 100 processors inside this car. And that included things like in a seat, when you raise and lower electronically the seat, there may be software involved in that with some cars. When you can remotely control the mirrors, there may be software involved with that. Some of the cars have automatic, the mirror will automatically go back.

So, basically, a modern car is a network of computers. We will talk a lot about the engine control module, but there are also air bag computers, and there are also antilock brake computers, and there are a number of other safety systems in a car that are embedded systems.

Q And we will focus through your testimony on the electronic throttle control system?

A That's correct.

Q Let's move then to what you have specifically looked at in terms of Toyota's source code for the electronic throttle control system.

A So I've had access to a secure room located in Maryland that had Toyota's source code and a number of other source code related documents produced in it. And in that room, I had access to the source code for the engines of a number of different Toyota vehicles, including the 2005 Camry, but also other models like the Lexus ES, the Tacoma and some others. And for many model years, from about 2002, when Toyota first introduced the electronic throttle control, until generally 2010 model years.

Q And you mentioned here what you saw was subject to confidentiality agreements?

A Yes.

Q I mean, just any of us could walk in off the street to this facility that used to be in Maryland and take a look at Toyota source code, could we?

A No. There were only 12 experts have ever been allowed in.

Q As I understand, that secure facility has now been moved to California?

A Yes. It was recently moved.

Q And a moment ago, we heard some testimony very briefly where some phrases from the source code were used when we were listening to Mr. Osawa's testimony. Do you recall that?

A I do.

Q And is it those bits of information and how they're described in Toyota source code that are subject to this confidentiality agreement?

A That's correct.

Q Is the operating system for these vehicles you listed here from 2002 to 2010, the Camry, the Lexus ES and the

Tacoma substantially similar?
A Yes. There are, to be clear, there are two different of operating systems that Toyota used in that time frame; one was a version of Itron, (phonetic) and the other was a version of OSEK. And I will come back and talk, more about OSEK which is relevant to the 2005 Camry. With respect to the details that I will talk about, they are substantially similar.
Q In terms of the software that actually runs the electronic throttle control system for the Camry, the Lexus ES, and the Tacoma in the year models that you have up here 2002 to 2010, is that software substantially similar for the analysis that you're doing?
A Yes.
Q And I guess I should have asked this earlier: I know you've testified in court before, but have you ever testified in court about the Toyota software issues that you're going to talk about today?
A No. This is the first time I've talked in court about what I've seen in this code room.
Q The type of software review that you've done in terms of Toyota software code, is that standard type of procedure used to evaluate source code for any type of product?
A That experts see source code is not unusual, but the protections around this source code are certainly unusual in

my experience.
Q All right. And I don't know if you can explain it to us. Give us just a general idea of exactly what it is when you go to review source code. What is it you're doing? Are there books there that have the source code written out?
A Thankfully no. The source code review involves looking at electronic documents on computers. There is basically a room the size of a small hotel room that is disconnected from the Internet, no cell phones allowed inside or would work inside. In that room there is about five computers and some cubicles.
In there, it is possible to believe view on the computer screen Toyota's source code. We couldn't take any paper in, take any paper out, couldn't wear belts, watches. There was a guard. It was worse than airport security was on the way here. Each time in and out, even to go to the bathroom.
Q How much time did you spend doing an analysis of Toyota source code?
A Countless hours. I haven't -- I mean, over a calendar period, it has been approximately 18 months that we had access to the code. I guess now it is maybe closer to 20 since the first production of source code for those vehicles. And so I was supported in there by a number of other engineers, including three from my own team from the

Barr Group.

Q And we heard some discussion about a NASA study related to this Toyota UA issue and the software. Did NASA have access to some of this source code?

A NASA was brought in to look at source code because NHTSA couldn't get to the heart of the problem, it didn't have any software engineers on staff. So NASA was given access to a few model years of Camry source code, as I understand it, at a Toyota facility in California.

They didn't have as much time. They didn't have as many vehicles, and so what we did actually was to build on their work. First, we confirmed that what they were seeing was consistent with what we were seeing, at least for the vehicles that they had, the 2005 Camry was the one they wrote about. And we also dug deeper, and so we pushed on various topic issues researching different aspects of the software design.

And importantly, NASA had a very tight time line and not necessarily unlimited resources or unlimited time to review the code. This is a Toyota document where they were discussing the NASA project internally. And Mr. Ishii's name -- and apologies for mispronouncing these Japanese names, I'm sure -- Mr. Ishii's name is on this document, and he is talking about how he or someone was talking to him about NASA has a very short time line, only a few months to

reach their conclusion. And that was the NASA process.
Q And I know we will talk about it in depth as we go along, but did NASA have access to as much information as you ultimately had in reaching conclusions about Toyota software?
A No. Even just for the 2005 Toyota Camry we had more documents, we had more source code, we had more things than NASA had.
Q Can you show us an example of what source code looks like. And I know what is on your slide is not Toyota source code, it is just an example, right?
A That's correct. We don't need to clear the room. This is just a simple example of source code in the same programming language that Toyota's main computer source code was written in. And that is the C programing language, the letter C. And it probably looks like nothing, right? But Dr. Koopman talked about how it is a -- like a recipe.
And so this is basically, what I put here, is some sample code in the C language for a recipe for something that most children in first grade or second grade can do, which is to figure out if you give them two numbers which one is larger. So this is a recipe for a computer to take any two numbers, and the recipe name is also the function name, which is larger of.
Now, I chose that name. I could have chose a less

descriptive name, or I could have chosen a more descriptive name. And the ingredients that the recipe relates to are what are called variables, so here A and B. So this is a generalized recipe. You can give it any two numbers. So you might tell a child is 67 bigger than 63? My son can do that. And this computer can do that by passing 67 as A and 63 as B, and then the recipe will compare them.

The first line here says if A is bigger than B, so if 67 is bigger than 63, then return 67. And if the situation was reversed, let's say it was 63 first and 67 second, then this "if" would fail, and we would go to the "else," and then we would return the 67 that came into second -- called parameters when they are passed -- so that is the recipe for comparing two numbers to see which one is larger and returning back the larger one.

So another part of the software can use this recipe at any time. And the last thing that I wanted to talk to you about is these things over here between the slash stars, and those are just simply comments.

Q Are both of those comments?

A They are. I only marked one of them. So the comments are simply more human readable stuff, but that stuff is it never seen by the computer. That stuff is there for the benefit of the programmers to explain what they are trying to do. So one way of explaining what you're trying to do is pick good variable names and good function names. And another way to explain what you're trying to do is to write a lot of comments or a commentary to explain what it is that you're trying to do.

Q All right. In terms of Toyota's source code that you would have reviewed for your analysis, I mean, you have shown us something here in English. Was it in Japanese?

A The source code was written in English. The variable names were in English. The function names were in English, and the things of that sort. The programmers were working in English. However, the comments were predominately in Japanese. We actually had a tool that came from a Japanese company that called Atlas that we could run in the room to translate things.

At first, we would cut and paste a particular comment into this tool, and we could read what it said in English. But then we actually had a small project where we wrote an automated process of converting all the comments at once into English so we could look at the code with the original English source code exactly as it had been and the translated comments next to it. Not everything was translatable automatically like that, but most of it was.

Q And I know you have given us an example here of comments just so we understand what you're talking about. Do you always have comments in lines code?

A Generally there are comments in source code. There need not be in order for the compiler to make a program, but they're generally are and should be so that the humans working with the code can understand it.

Q And you just mentioned a word there, compiler. In terms of reading source code, what is a compiler?

A A compile is a development tool that programmers use. It is another piece of software, one that they use to take the human readable code and turn it into machine readable binary code that can be downloaded in your car, for example.

Q When you say it is turned into binary code, what is binary code mean?

A Sorry. Binary codes is ones and zeros. And the machine knows what to do with them because it knows that it should group them together into groups of 16 or groups of 32 and that certain ones are instructions that it know what to do like add two numbers, compare two numbers, see if something was zero, move to another address, things of that sort. And the compiler generates sequences of these 16 or 32 bit instructions, which are a bunch of binary bits. And the computer knows how to interpret them and what to do to follow the recipe in that situation.

Q Now, you mentioned using your tool to help you translate part of the comments into English. Were you required to use any other types of tools that would help you or assist you to read the source code while you were in the source code room?

A Well, we weren't required to necessarily, but we had access to a number of tools that we did use. We requested that certain tools be placed into the room when the room was open. And those tools included the actual Green Hills compiler that Toyota used, a related set of utilities that would have been used in a software development process, names I don't need to bother you with.

And also, importantly, a simulator which Green Hills provides, along with the compiler, which is able to pretend to be the target processor so that you can run code and step through it one instruction at a time, if you like, or set places where you want to stop and see what is going on. We did take advantage and use that simulator in our analysis of the source code in the code room as well.

Q Would the simulator help you to read or understand the instructions in the code as if it was running in the vehicle?

A Yes. But of course the simulator itself is just running on a desktop computer, so it is not a vehicle. So it cannot simulate all the things that a vehicle can do.

Q Were you able to run certain tests on the software in the source code room?

A Yes.

Q What sort of tests did you run?
A Well, so, for example, we were examining the operating system and understanding how the operating system worked, and we were able to use the simulator to both examine what is happening while the computer ran or what would be happening in the car. And also to analyze certain aspects of its behavior to see if it functioned as it said in the user manual, for example, or as it said in the source code and things of that sort.
Q As you're reviewing the source code, did I hear you say earlier that you couldn't take notes and carry them out of the room?
A No. To-do lists were a bit of a problem. You had to remember that you wanted to get something when you got out of the room and then go look it up, and you had to remember what it was you learned when you went back into the room. It was quite an impediment to the process.
Q While you were in the source code room using some of these tools and reviewing the source code, were you able to identify any coding rule violations?
A Yes. Many.
Q Was there a specific tool that you used to do that, or was that a manual process that you yourself had to go though?
A Well, checking for compliance with coding standards

can be done both by reviewing things as a person sitting there looking at the code, but that is not necessarily efficient. So for some coding rules, at least, there are tools called static analysis tools which look at the source code for you and look for certain types of rule violations, and we had access to several tools of that sort in the code room, and we used them.

Q And you were here last week for Mr. Ishii's deposition?

A Yes. I heard that.

Q He mentioned something about source code modules. Do you understand what he was talking about?

A Yes.

Q Explain that to us briefly.

A Yes. So the source code consisted of for a particular vehicle on the order of a million lines of code. And so by a line of code, I mean like a line in a document. So if you look at the page of a Word document, it might have 50 lines on. If you were to print out a million lines of code, you can imagine it would be pretty large.

The source code is generally, and Toyota's was, divided up into what are called modules. So related recipes, or parts of the recipe are grouped together in files, just like I broke up my report into a summary and 13 chapters. They broke up their software into approximately

4,000 files. Don't quote me on that number, but it is on that order.

Q So while you're there, knowing that they are in modules is your focus to look for the modules that relate to electronic throttle control?

A Well, in one since there are modules that relate to electronic throttle control because they are recipes that are specific to electronic throttle control. But in another sense, it all relates to electronic throttle control because it is all running on the same processor. So one part over here that might not appear to be named as throttle control recipes can actually interfere with and cause problems with the throttle control recipe.

So it is not that we only looked just at the code that said, Here are the throttle recipes. We did, but we also had to look at other parts of the code as well.

Q Through this source code review, were you able to identify bugs within Toyota's software?

A Yes.

Q What sort of tools did you use to identify those bugs?

A Most of the bugs that we -- that I wrote a whole chapter on bugs that we found in their code -- most of those were found inadvertently. They were found when we were reading some module to see how it worked because we were understanding the system, and we found that there was a bug in the code.

The other way that we found bugs was when we ran the static analysis tools, for example, to see if there were rules violations. Sometimes those rule violations or the results from the tool would be -- would turn out to be bugs. So the static analysis tool doesn't say this is a bug, it says there might be a bug here. We investigated those, and some of them were bugs.

Q Did you find all the bugs in the software that you reviewed?

A Absolutely not.

Q Why not?

A Because there is a lot of bugs, and all indications are that there are many more. We haven't specifically gone out looking for bugs. The metrics, like the code complexity and a number of global variables, indicate the presence of large numbers of bugs. And just the overall style of the coded is suggestive that there will be numerous more bugs that we haven't found yet.

Q And we have talked about bugs. Can you for the benefit of all of us tell us what you mean when you say there is a software bug. What does that do to the software?

A Software bug causes the software not to work right. It can be a little thing. If you're editing a Word document on your computer, you might see that suddenly one area of the screen is not drawing right, and you have to refresh or close the application and bring it back. So that little momentary glitch that you might see, or it could be something big like the whole program crashes or the whole computer crashes and you have to start over.

Q You were here earlier and heard Mr. Osawa's testimony?

A I did. Yes.

Q You understand that he was a Denso engineer?

A I did.

Q And Denso provided the monitor CPU within the electronic throttle control system?

A Yes. That's one of the things that they did.

Q Did you hear his testimony where he said they had never found any bugs in their software?

A I did, but I didn't think he was just referring just to the monitor CPU.

Q My question goes back to this: Is there any software that you're aware of that does not have bugs?

A No.

Q And we will talk more about this later, but I want to go ahead and bring it out. The term task death. Can you give us just a general description of that, because we will need it as we go on.

A Sure. I think it is a bit premature. I can give you briefly that a task death is a type of software malfunction.
Q Were you able to test for task death while you were in the source code room?
A Yes.
Q And were you able to cause a task death in the source code room?
A Yes. We able to confirm that tasks could die in the Toyota ETCS and that would cause a software malfunction.
Q Go to the next slide. Tell us why you put this in here.
A Yes. Before we talk about the software anymore, I think it is important that we all sort of have a high-level view of what is going on. And you might know how a car works, you might have thought about it some, but not in a while. Let's start at the beginning. The driver has two ways of controlling a vehicle's speed or making it go faster. One of those is using the accelerator pedal. The more you push down, the faster the car goes. The other is using the cruise control where the computer and the software will take over and keep the speed at a constant.
On the right-hand side, I have drawn fuel, air and spark. And that's because you need those three elements in order to make the engine go, at least in the gas engine. A useful analogy is if you have ever pushed a child on a

swing, or someone on a swing, you know that you are giving them motion, but they also have a certain motion of their own that will continue if you stop.

Same is true with a combustion engine. The combustion engine is causing the piston to go up and down and the crank shaft underneath to rotate and move the pistons up and down together. There is a certain amount of that motion that is like the swing going back and forth that will keep going briefly.

The spark, or the fuel, first of all, is you have to have energy. You, yourself, have to have energy in order to push them. That's where -- the energy comes from the fuel. The spark relates to the timing when you push. If you push at the wrong time, you know you will not get as much umph, you are not going to cause as much of an increase in the power of the swing unless you hit at the right moment; that's what the spark does. The spark ignites the fuel at the right time.

The air that is in chamber that is compressed in the chamber with the fuel, that is coming in through something called the throttle. And that is controlling how hard you push. So the more air that you let in through throttle, the more push you are giving to the swing; therefore you will get a faster engine out. And the spark is just going to follow along and hit it at the right time.

The air is really going to provide the power for the engine.
Q So is our throttle control system and the area that we're concerned about what is controlling the air in the system?
A We are. And I will get there in a minute. So the throttle, for a minute, it is a fancy word, in a car it is a fancy word, but it is really no different than you turning up the hot water in your shower. You get in the shower and your turn the knob. What is happening inside that pipe is there is something blocking the water, and then there is not something blocking the water.
You can make it 100 percent of all the capacity that it has hot, or you can make it zero percent of all the capacity that it has hot. The same is true in the car's engine. When you close the throttle, you're robbing the combustion engine of its fuel, of its power. There is still the gas, of course, but you need fuel and air ideally in a certain ratio in order to cause the explosion.
So the air comes through the throttle. If you think about an older car, where your foot on the accelerator pedal is always adjusting the throttle, your foot is directly in control of how fast the engine is going, and that is what is giving the car power. The change to the electronic throttle control, which with Toyota began in about 2001 in the Prius and 2002 in the Camry, at least in

the United States, that means that earlier car computers had been in charge of the spark and the fuel.

They had been in change of two of the things that make a car go. But the driver had always been directly in control of the air, which is directly related to how much power the engine has. When electronic throttle control comes in, you have software that is now responsible for all three of them at once. So you have a portion of the software, the job of which is to make the spark at the right time, inject the fuel at the right time and the right amount, and open the throttle a certain amount.

And the throttle opens to allow air to actually be sucked in. Not blowing in air, but instead the vacuum that is left behind, after the previous combustion, you have blown up everything in there, every air particle and every gas particle, for the most part is gone. So you have to put in both new fuel and new air. So it actually the vacuum sucking the air out of the throttle, out of the tube, into that chamber that is causing it. So you're just allowing more air to flow in and the combustion is taking it from there.

The software in electronic throttle control is responsible for all three things, which means if the software malfunctions, it has control of the engine and can take you for a ride. What is of particular importance is that there is another part of the software that is looking at the driver controls, looking at the accelerator pedal and cruise control -- it is looking at more than that, but that is a simplification, that is appropriate right now -- so there is a part of the software looking at what the accelerator pedal position is, is it down, is it up, how much down. Then that is translating that into a calculated throttle angle. And then another part of the software is performing the sparking and the throttle control.

Q Is this what is referred to when we heard it here drive by wire?

A Yes. Some people call it drive by wire. It is confusing to me because there used to be a wire and they took the wire out and they call it drive by wire.

Q Do you have an example of what Toyota's computer module looks like that controls these things?

A Yes.

Q So I think you have a laser pointer on that thing that you have?

A Do we have the actual board.

Q I do. Explain to us what we have here.

A So this is a photograph of the ECM. And this ECM, or engine control modules, has two big chips on it. Has a bunch of other chips, capacitors, circuit tracers that you can see, and other things. This biggest one, the square one, is the main CPU. It is a type of a CPU or a model of CPU called a V850. That is kind of the equivalent of calling it a Pentium. V850 is the model number of that processor. Comes from a company, a supplier of Toyota that used to be called NEC. It has since changed its name.

Then there is a second rectangular chip here, and that chip is what has been referred to by various witnesses as the monitor CPU, the ESP-B2 and sometimes the sub-CPU. Importantly, each of those is a processor with its own software. Then, of course, all together they comprise an embedded system.

Q So the software that we're going to talk about is stored within components on this board?

A Almost always when I'm talking about the software, I'm talking about the software on this main CPU, which performs the throttle control, the combustion, monitors the accelerator, and all those things, cruise control. But there is also software, and I will specifically call out when I'm talking about this monitor CPU and its software.

Q This is from a 2008 Camry?

A This particular photo is from 2008 Camry.

Q Is the 2005 generally very similar to this?

A The chips would be moved around a little bit, but in terms of the electronics of what is there, there is a V850 processor, there is an ESP-B2. From a substantial

similarity point of view, they are very similar.
Q Can you tell us what this is.
A That is the very 2008 ECM that this photograph reflects.
Q Would this be the general size of the board that contains these compute components with a 2005 Camry?
A They are about the same. Correct.
Q Let's talk about safety critical systems?
A So a safety critical system is an embedded system, but it can also kill or injure someone. So my Nike fuel band is not going to kill or injure anyone. But a car is an example of an embedded system, at least some of the computers inside it, can cause injury. Now, it wouldn't be a case necessarily of the mirror control, but it would be the case of the engine control.
Q So do you consider the electronic throttle control system to be a safety critical system?
A I do.
Q What sort of things can possibly go wrong with such a system?
A Well, the risks in such a system are manyfold. The first is that these electronics are being driven around, bounced around, splashed around, and in a generally rough environment. A lot of embedded system designers don't have to worry about their products doing anything other than

sitting on a desktop, but a car is a very harsh environment.

So it is a noisy environment, electrically noisy, there is a lot of vibrations. And so one of the things that can go wrong -- and this can happen in any electronics, but it can particularly happen in a car electronics -- is some sort of glitch in the electronics. And that means that momentarily one bit inside a chip flips or an electrical pain takes on the wrong value.

With a digital value, if you have an in-between number between zero and five volts, you might inadvertently get momentarily wrong signal, and that can affect what the software does. So that is one thing that can go wrong, a glitch in the hardware. You heard Dr. Koopman talk about the bit-flips. Another thing that can go wrong is that there could be a software bug and it can be activated at any time. So the software bug is latent, always there, but then you happen to be driving a car that day and the software bug suddenly, because of something the car did or a glitch in the electronics or something else, it suddenly activates, and now you have a malfunction.

And any reasonable -- any program of reasonable size is going to have bugs in it, so you have to, as a designer, expect random hardware faults and also there are software bugs in there.

Q Let me ask you a question about that: In terms of software bugs, just because they're there will they always cause a malfunction?

A Just because they're there doesn't mean they will always cause a malfunction. No.

Q Are some bugs such that there has to be a specific condition met with the product, the car, whatever in order for them to manifest themselves?

A Yes. So just going back to my simple example of the larger recipe, that is a very simple recipe. But suppose it was a more complicated recipe and we gave it two numbers, you know, 8,012 and a million and 16. And for that case, maybe because one of the numbers was over a million or maybe because of the difference between the two numbers or maybe because of a bounce that this car did at that very moment or an electrical glitch or something else, it gives the wrong answer. Instead of saying the larger number is a million, it says the larger number is 8,000. That is an example of a bug that was there. It might have never caused a problem, but in that particular instance, it caused a problem.

Q For example, there has been some testimony or discussion in this case that Ms. Bookout bought this car, driven it for several years, put about 9,000 miles on it, never had a problem. I don't think there is any dispute about that. In a circumstances like that, could the car have bugs but yet never display them?

A   Yes.
Q   In order for the bug to display itself, would the vehicle just have to meet and put itself in certain conditions that would bring that bug to the surface?
A   Yes. But let me just be clear that there is vehicle operating conditions and then there are software operating conditions. So you can think about the vehicle operating conditions is like whether you're accelerating, whether you're decelerating, whether you are pressing the brake, whether you are not pressing the brake, whether you have cruise control on, whether you don't. Those are all different examples of the vehicle being in different states.
          But also the software internally contains many thousands of variables, all of which can have different values at the moment. Think about that spreadsheet full of numbers that Dr. Koopman talked about. That is all going on at the same time. Essentially, all the possible values of those things represent different software states.
          So you have a very large -- measured in billions or trillions, or essentially an infinite space -- of software states. If you get yourself into one of those corners, then the bug can occur. And that might not be because of what you were doing with the car that day, it could simply be that the software got into that place. Then what is happening with the car layers on top of that, because maybe

you were going five miles an hour and versus going 50 miles an hour, then you might have a different outcome.

Q You've mentioned, and Mr. Ishii mentioned, that there is always bugs. As a software developer, somebody that analyzes embedded systems, is it reasonable for a manufacturer to try and put in safety features which try to take up for or anticipate what bugs may do?

A Yes.

Q And have you mentioned that here?

A Yes. So the third thing that can happen is that if you're a software developer and you think, Oh, well, I'm worried about the possibility that someone will set the throttle angle to 150 percent -- and I don't know what that means, but that sounds bad, I don't want it more than 100 percent. So you might think about that, so you put in a detections that says if it is ever more than 100 percent then do something safe. That can range from, depending on the situation, keeping it at 100 or saying, Well, I don't know why it ever would have been more than 100, there must have been some serious problem and resetting the computer.

But just because a company and its engineers think up 100 possible things that can go wrong, or a thousand possible things that can go wrong and implement a set of failsafes that they think will defend against them, there is two problems with that. The first is the failure of imagination possibility, which is it didn't get on their list. They forgot that it was possible that tasks could die, for example.

Another possibility is that failsafe itself has a bug in it, a hole in it, a gap. They think they have mirrored all the critical variables, made a second copy of them, but they haven't. Or they think they have a Watchdog supervisor that detects task death, but it doesn't or doesn't always. So they can have gaps in their safety architecture.

So a third thing that can go wrong is that one of those gaps is exposed in the safety architecture. And sometimes it takes all three of those happening at once in order for your car to malfunction or to malfunction in a dangerous way that you report. For example, it might begin with a hardware bit foot, and that might cause a bug and that might escape detection because they didn't think of that possibility.

Q Are coding standards like we've talked about and heard from Dr. Koopman, for example MISRA, are those structures that manufacturers can use or rules that manufacturers can use to help reduce unforeseen gaps in their safety architecture?

A Yes. Well, no. Not specifically in their gaps in their safety architecture. They can help to keep bugs out.

Q And if you don't have bugs, then it helps to create -- you don't need as big a safety architecture?
A I wouldn't say that's true either.
Q Okay. What would you say?
A I would say that following a coding standard like MISRA-C can help to reduce the number of bugs in your software. Doing what Dr. Koopman talked about, which is having a software process like MISRA software standard, the Fat Standard, or the ISO Standards, that is a way to make sure that there are no single points of failure in your system. And so even if you have a bug that you don't know is there, you always have a way that it will be safely handled.
Q So in terms of creating a safe architecture, a safe system, can it be something that is an afterthought?
A No. You have to design in safety. Safety has to be there from the beginning. I think Dr. Koopman said it really well. He talked about the Therac-25, which was a famous case that embedded software engineers studied where a medical device that was used in treating patients, actually was killing them by giving them too much radiation.
And he talked about how Dr. Leveson at MIT who studied the subject she found that simply the developers would find a bug and fix it and think they had solved the problem, and then the next patient was given too much

radiation and they would find a bug and fix it. You cannot got down the path of find a bug and fix it. You have to design safety in.

And that's also important because sometimes embedded systems can't be updated, can't be upgraded. For example, in this Toyota electronic throttle control, there are two processors. The main processor has the potential to be updated, have the software updated, when you're in the dealer. It is capable, anyway, the chip of doing that.

But the second processor, the monitor CPU is burned in a factory, a million chips all alike, and those chips can't ever be changed. So if there is a flaw, you can't go in and fix that flaw, so you have to have a good design from the beginning, you know, separate fault containment regions, no single points of failure, and you should follow a software process, safety process, in order to achieve that.

Q Let's look at our next slide. I think Dr. Koopman showed us this one as well.

A Right. So the slide says two things. First of all, it says that NASA agrees that Toyota's electronic throttle control is a safety critical system. They add some other terms of art that I don't think we need to get into, hard realtime. Then this figure that Dr. Koopman had shown may make a little more sense now, so I will just briefly explain it.

On the right, we have the combustion controls, we have a throttle valve that is controlled through a motor. The motor is doing the job of turning the knob on that hot water. We also have the fuel injection, that is the squirting of the fuel into the cylinder, and then we have ignition coil, which is charged up and then at the appropriate time creates a spark.

The ECM in pink is the circuit board that has the two processors on it. And there is some explanation of kinds of thing that it does, but it does a lot more than this. You can see that it is monitoring the accelerator pedal, it is making sure you car doesn't stall by setting the idle speed, which can be different depending on whether you have the heat and air conditioning on, things of that sort.

The cruise control, the transmission shifting and various over functions are taking place in there if you have an automatic transmission. Then this is showing the inputs to that. So, for example, the accelerator pedal sensors and other vehicle sensors that are used in that process.

Q All right. So is the significance of this slide that NASA has reached the conclusion that this throttle control system is a safety critical system?

A I think that is an important point. Yea.

Q Now, based on all the things that you have done and the analysis that you have done in this case, have you reached some conclusions that you will talk to us about?

A Yes.

Q Is that the next slide?

A Yes, it is.

Q All right. Let's start with the first one at the top. And tell us about your conclusion.

A So the first main conclusion is that the 2005 Camry electronic throttle control, the software os of unreasonable quality. It contains bugs, but that's not the only reason it is of unreasonable quality. And it's otherwise defective for a number of reasons. This includes bugs that when put together with the defects can cause unintended acceleration.

Q As we go forward are you going to explain to us how those problems that you found will cause an unintended acceleration?

A Yes.

Q Then you mentioned the code quality metrics. What do you mean about that?

A So the code complexity and the McCabe Code Complexity is one of the measures of that. And the code complexity for Toyota's code is very high. There are a large number of functions that are overly complex. By the standard industry metrics some of them are untestable, meaning that it is so complicated a recipe that there is no way to develop a reliable test suite or test methodology to test all the possible things that can happen in it.

Some of them are even so complex that they are what is called unmaintainable, which means that if you go in to fix a bug or to make a change, you're likely to create a new bug in the process. Just because your car has the latest version of the firmware -- that is what we call embedded software -- doesn't mean it is safer necessarily than the older one.

So the metrics that I see in the source code that I will talk more in specific with you about, they predict that there are many more bugs.

Q Are you also going to tell us about a conclusion that we see on the board related to the failsafes?

A Yes. And that conclusion is that the failsafes are inadequate. The failsafes that they have contain defects or gaps. But on the whole, the safety architecture is a house of cards. It is possible for a large percentage of the failsafes to be disabled at the same time that the throttle control is lost.

Q And you make that statement, but in practical terms what does that mean?

A That means that the random hardware fault that can occur from time to time, the software bug that is latent, lurking, witting to happen can on the right day and the right conditions can get through or knock down the failsafes that are in place.

Q All right. And the your last comment here.

A So ultimately my conclusion is that this Toyota electronic throttle control system is a cause of UA software malfunction in this electronic throttle module, can cause unintended acceleration.

Q And I know we will get to it later, but ultimately you have a conclusion that it also was the cause of the wreck in this case?

A I do.

Q All right. And we mentioned it here, we mentioned it several times, unintended acceleration. Do you have a specific definition for that?

A Yes. I have simply adopted the definition that was used by NHTSA and NASA, which I think is a reasonable definition, which is if the vehicle is experiencing any amount of acceleration that the driver didn't want or purposely caused. And that comes in different flavors, of course. It could be that the car suddenly accelerated away, but it can also be that the car continued to go at the same speed even though you let off the accelerator. So I've cited that definition here from the NHTSA report that was published in 2011.

Q All right. Now, Mr. Arora, who is sitting right back

here, is Toyota's software expert. And you reviewed his work, correct?
A Yes, I have.
Q Does he also use NHTSA's definition for unintended acceleration?
A No, he doesn't.
Q All right. Let's go to the next slide and talk a little bit about NASA.
A Before we go on, I just want to say that I also sometimes will refer to it as loss of throttle control. So if you lose the ability as a driver to control what is happening with that throttle valve, that is another way that I sometimes say unintended acceleration. You might see that on the slides, you might hear me say that.
Q All right. Let's look at the next slide. Before we get into the details of the conclusions that you have here from the NASA report, NASA had a report, evaluated some vehicles, software and came up with conclusions, correct?
A Correct.
Q Have you essentially taken what they have done and built upon it?
A Yes.
Q Tell us what is significant about the portions here in this slide that you're showing us.
A I was actually familiar with the NASA report and had

looked at it before I was ever engaged with these cases. One of the things that jumped out at me as an embedded software engineer reading the work of other embedded software engineers at NASA was that their ultimate conclusion was not from their analysis that a software bug or malfunctioning could not cause UA.

They simply concluded that in the time they had they couldn't find the bug that caused UA, or a bug that caused UA. And, in fact, they sought a very narrow definition of UA. They thought -- they saw it, and they state this in the report -- only a bug that would open the throttle more than 25 degrees, not leave any, what are called diagnostic trouble codes behind as evidence later, and some other criteria. I'm not sure why they scoped it in that particular way.

Q And we will talk about diagnostic trouble codes later, right?

A That's correct.

Q All right. This slide here, does it show some of NASA's scenarios that they postulated where a UA can occur?

A Yes.

Q Take us through it, please.

A So NASA summarized, in particular on a table on page 78 of their main report, a bunch of scenarios that they considered could cause UA. And they had ruled out a number of them, but there are two rows left that they couldn't rule out. And that is what these paragraphs are about.

The first row that they couldn't rule out is that the accelerator pedal has two sensors, redundant sensors. And the first one they couldn't rule our is if they both failed together, or were electrically entangled, became electrically entangled, then as a result there was no way for the system to detect that.

So they worried, one, that that could cause UA. Then the second one they were worried about is what we will have talking about which is a systematic software malfunction in the main processor that is not detected by the monitor system, the monitor CPU. I think that is the main quote.

Q Okay. So one of the proposed scenarios that NASA thought might could happen is that which you believe happened in this case?

A Yes.

Q All right. What else about this slide is important?

A Well, ultimately, you can see at the end there also NASA states clearly that just because they didn't find the bug, the proof, doesn't vindicate the system or say that the system is safe. NASA didn't say in their report that the system was safe.

Q All right. And are you going to describe -- I think in your next slide -- several of the defects that you found in Toyota's electronic throttle control system?

A Yes.

Q All right. Start at the top and describe them for us.

A So we're going to be talking about these things in more detail. I want to kind of give you a preview of where we're going, if you will. So NASA falsely understood or misunderstood that all critical variables, or all critical values in that spreadsheet had a second copy, and that's not true.

Q Is that called mirroring?

A That is mirroring. It can be called mirroring or echoing depending on precisely how you do it. But, generally, we can use the term mirroring.

Q Will we discuss that in more detail later?

A We are. Just to be clear, what we found is that NASA had a misunderstanding here. There were actually critical values that were not mirrored.

Q All right. What is next?

A The other thing is that Dr. Koopman talked about how bit-flips can occur in the real world. There can be a one that becomes a zero or a zero that becomes a one, and this can happen inside integrated circuits or chips. And NASA was under the false belief that there was a protection mechanism in there. Dr. Koopman gave an example of a parody bit, an extra bit of information, additional bits of information that were like a partial copy that indicated something was wrong.

And that is also known as EDAC in NASA's report, E-D-A-C. It stands for error detective and correction codes. And so NASA didn't know that that wasn't there. It wasn't there in the 2005 Camry. And so if the bit-flip occurred, there would be no hardware mechanism to find it. And if it occurred in a critical value that was not mirrored, there would be no software protections against it.

So the conclusion here is that there are critical variables in which bits could flip. Or there could be a software bug if you correct them.

Q NASA, as part of their evaluation, looked specifically at the 2005 Camry, correct?

A They did.

Q And are you telling us that they were under the belief that the 2005 Camry had EDAC?

A Yes.

Q Does that make a difference in the analysis?

A Yes.

Q Does the 2005 Camry have EDAC?

A No, it does not.

Q How do you know that?

A We received additional information that NASA didn't have. We received information, a spreadsheet, that summarized -- it is one of the documents that I'm most familiar with this -- which is a spreadsheet that showed which vehicles, like Camry, which model years, like 2005, had hardware memory protection and which ones didn't.

There was a sort of EDAC, not as much as NASA was talking about or NASA would employ in space, but there was one in the 2008 Camry, but there was not in the 2005 Camry. So later they put it in, but they didn't have it in the vehicle that NASA studied.

Q And you're going to talk next about memory corruption?

A Yes. So hardware bit-flip can occur. And NASA states that as well, and they were concerned about that, which is why they relied on the EDAC being there and the mirroring. But there were also bugs in Toyota's code that will have allow memory corruption to occur from a latent or just hanging around software bug from time to time.

Q A hidden bug?

A A hidden bug. That's right. One of those relates to stack overflow. NASA didn't realize that a stack overflow was a possibility, but our analysis shows that it is. And I will talk more about that. And also there are also software bugs. Now, NASA found bugs and said they found issues in

Toyota's code, but they didn't find the one or a one that opened the throttle 25 degrees and various other things.

We found a set of bugs that specifically can cause memory corruption. So they're lurking there. And if they happen, then as a result of that, then the some critical variable could be -- could have a new value, for example, the throttle commend could become instead of opening 20 percent opening 50 percent letting in a lot more air and giving the engine a lot more power.

Q And you will discuss that in a lot of detail later right?

A Yes.

Q So is it, at least right now, memory corruption is a way that UA can occur?

A That's correct.

Q All right. And we're going to get into detail on these defects. But the thing that I wanted to ask you about, are these defects that you will discuss consistent with the opinions and testimony that Dr. Koopman gave us last week?

A Yes.

Q He talked to us about the process and rules and that sort of thing on how to create a safe system. Does your analysis for this case go deeper than what Dr. Koopman's did?

A Yes. So Dr. Koopman was not able to see the source code, and so Dr. Koopman's analysis focuses on the science that underpins designing safe systems, that standards that are available to carmakers for making their car safe, whereas I support that by examining the source code and finding that those things weren't done.
Q So where he couldn't tell us whether those problems that he saw caused Ms. Bookout's unintended acceleration, you're able to go into that detail analysis because of your review of the source code?
A That's correct.
Q All right. Let's go to the next slide?
A So the ultimate conclusion from the presence of these defects is that the software could malfunction. And the most dangerous such malfunction would be if the car had a portion of its software that was working, and that part was running the combustion feeding air and fuel and spark to the engine at the same time that the part that the driver was interacting with through the accelerator pedal or the cruise control switches was not listening to the driver because it crashed or hung, like one application might crash on your desktop while another one is still running.
Q And are the defects that you're describing here that can cause an unintended acceleration, can that occur when the cruise control is on?

A Yes.
Q Can it occur when the cruise control is off?
A Yes.
Q And it is the same software defects that would relate to both?
A Yes.
Q Let's go to the next slide. You're talking about the software malfunctions here?
A Yes. This just uses an analogy and makes the point that, of course, software malfunctions. And we see it all the time in our daily lives whether it your laptop or your desktop, sometimes you have to reboot things, restart applications, et cetera.

It is a fact of life that software developers are well aware of, or should be well aware of that software malfunctions can occur. I don't know if you ever had the experience where is one app on your Smart phone is not working and the others are. And we all know, we are trained to reboot it. Just reboot it. Oh, you didn't get my phone call? Well, maybe your phone is not taking calls right now because of a software bug. That can happen in an iphone or an android. Even though your might be able to make outgoing calls, if one part of the software is not working, the rest is. So you reboot it and suddenly everything is fine.

The 2005 Camry has apps. They don't call them apps, they call them tasks. And so there are [redacted] tasks inside the engine. As an example, there is one task whose job it is to keep track of how fast the car is going. That is important, obviously, if you will have a cruise control feature because a cruise control needs to know not only what speed you would like it to be but what speed it really is.

Q Let me stop you right there.

MR. BAKER: Your Honor, my next question is going to involve some source code. So at Toyota's request, I think we need to clear the folks out of the courtroom again.

THE COURT: Is this going to be periodically, or is this the only time?

MR. BAKER: I hope this is the only time.

THE COURT: If not, I will just exclude everybody from this point on. You think this may be the only time?

MR. BAKER: I will transition into our nicknames for it so we don't have to do it anymore.

MR. BIBB: I think there is one other area that I noticed, but it is a long way from here in this slide show.

THE COURT: Again, if you do not have source code access, please exit the courtroom.

(Whereupon, the courtroom complies.)

THE COURT: You may proceed, Mr. Baker.

MR. BAKER: Thank you, your Honor.

Q (By Mr. Baker) You're talking about [redacted] tasks that run this system, correct?

A Correct.

Q All right. Earlier we heard some testimony from Mr. Osawa, and he mentioned a couple terms that I believe are tasks names, and I want to ask you about those. He mentioned [redacted]. Is that a name for one of these [redacted] tasks?

A It is.

Q He also mentioned [redacted]. Is that also the name of a task?

A It is.

Q All right. And in terms of [redacted], do any of those characters have specific meaning to you or a programmer who is looking at this?

A Yes. In Toyota's design, there were [redacted] tasks. And some of those tasks did things on a time basis. There were three of them, in fact. One of them that did something every [redacted] millisecond, one of them that did a lot of stuff every [redacted] milliseconds; and that's this one, [redacted], and another one that did a lot of stuff, again, every [redacted] milliseconds. And those are known as the [redacted] millisecond [redacted], [redacted], and [redacted] millisecond [redacted], [redacted] tasks.

Those are the only tasks that were named quite like that. Most of the other tasks related to moving the combustion process at a certain speed that varied depending on the engine speed, so it wasn't time based. And also there were some asynchronous things that happened separate from the engine speed, separate from the time, amount of time.

Q And these two terms that we have specifically referenced here, are those source code terms to which Toyota has claimed are confidential and they don't want the public to hear those characters?

A Yes. If you were to look at my report there, you would see every time I said [redacted] it is blacked out. Every time I said [redacted] it is blacked out. And other similar things are blacked out, and the same is true with the deposition transcripts from my testimony.

Q And so for these [redacted] tasks that you referenced here, each one has a name like this similar?

A Well, as I said, there is only the three that have time-based names.

Q In terms of our case here, are we going to talk a lot about [redacted]?

A We are.

Q In order to avoid having to clear the courtroom every time we talk about it, do you generally talk about in your work as task X?

A I do. I call it task X, letter X.

Q So whenever we say task X, you're referring to this

specific task?
A That's correct.
Q For that specific task, can you tell us what particular functions that task has to perform?
A I can't, because it is a very extensive list. I actually also refer to this tasks as the kitchen-sink task, because it does so much in the system. But importantly, for our purposes, it does throttle control; that is it selects the next throttle percentage, whether it should be 100 percent, 50 percent, 20 percent. And it does that based on looking at the accelerator pedal position, whether the cruise it on.
It executes also the cruise control code, so it is responsible both for turning on cruise control, maintaining speed of cruise control, and turning off cruise control. It also is responsible for many of the failsafes on the main CPU. We will talk more about that as well.
Q We also mentioned DTC. What do those stand for?
A DTC stands for diagnostic trouble codes. And most of those also are either in the [redacted] millisecond task, task X, or they are -- they require its help in order to be recorded. These are codes that are recorded in your -- if you have ever taken your car to the dealer because the check-engine light was on and they read the computer and they told you that you have a back oxygen sensor or

something like that, that is an example of a diagnostic trouble code. Many of them indicate there is a problem with a sensor or a that there is a problems with some other engine component.

Q And in your Camry, is it this task X that has the job to either set or help set diagnostic trouble codes in the car computer, at least associated with what we will be talking about?

A Yes. I won't say all of them, but most of them, the vast majority of them, will not be recorded unless that task X is doing all its job.

Q You have gone through all these things, you told us this task has control over or performs. Is it unusual for a single task to have so many tasks within it?

A Yes. It is not a good software architecture.

Q Why is that?

A In particular, combining the part of the system that does the calculation of the throttle angle with the failsafes and trouble codes is a well-known bad design. There is a pattern that people usually follow where you have a controller and you have a monitor. And so even within the software, it should have been architected so that the control of the throttle was separate from the failsafes related to the throttle and sensors that inputs them.

Q Let me ask about that then. The jury heard testimony

about a brake override system. Are you familiar with that?
A Yes.
Q Wherein the accelerator is in certain condition, if you press the brake it will automatically cut the throttle. Are you familiar with that?
A I am. There is not one in the 2005 Camry, to be clear.
Q Right. Do you have an understanding of the system that Toyota has since used?
A Yes. I reviewed the one that they put into the 2010 Camry.
Q Where is the function for that brake override? Where is the task located, as you understand it?
A Yes. So the brake override that is supposed to save the day when there is an unintended acceleration is in task X, of course, because it is the kitchen sink.
Q All right. And we will later in more detail about task death where a task just stops running, correct?
A Yes.
Q And I think your focus is going to be in on the death of task X?
A That's correct. I don't think I will need to name any of the other tasks in order to talk about the rest.
Q Just to followup your example on brake override systems, if Toyota's system were used, and task X died and

caused a UA, would brake override work?
A No.
Q Why not?
A Because you have software watching the software. So if the software malfunctions and the same program or same app that is crashed, is supposed to save the day, it can't save the day because it is not working.
Q How would you fix that?
A Well, the right way to design a brake override, in my opinion, is to have it on an external chip. It is not just my opinion, it is also in a standard called EGAS (phonetic) for automotive makers. And in that design, you have a separate chip that looks at whether the driver is braking and whether the throttle is open. Does it make sense that you're braking but you are having to fight the throttle because it is open 50 percent or 100 percent?

It would be relatively simple, and I will have explain later how Toyota could have done this back in 2002 without any extra cost to the vehicle, that if you were braking and the throttle was stuck that there must be something wrong with the main CPU and it can reset. A car traveling at 60 miles an hour, a Toyota 2005 Camry traveling at 60 miles an hour, can reset its computer in about 11 feet.

So it's okay to reset the computer in order to solve the problem. And that would, just like resetting your iphone, solve the problem. And Toyota had the means and could have done that, but they didn't do that in the 2005 Camry. Even in the 2010 Camry, when they were responding to the NHTSA problems and investigations, what they did was software watching software. They didn't put a separate chip or have a proper brake throttle override.

Q Have you covered everything on this slide that you want to talk about? I have a question if we have.

A There is one thing that I want to talk about. I wrote there all of these tasks are meant to be running always. So I talked about task death a little bit, the idea that one app crashes, right?

So what if you're driving down the road and you only now have [redacted] of these tasks working but your car seems to be operating normal? Is that a good thing? No. Let's say that there are [redacted] tasks, each had assigned to it one programmer at Toyota or Denso. It is as if though one of them, you're not benefitting from the work of that [redacted] engineer that day while you're driving down the road until you restart your car. It may cause a malfunction that is dangerous. It may cause a minor malfunction that you don't even notice. Then when you restart the car, it goes back to being a car.

Q Let's talk a little bit about the operating system we discussed earlier. Tell us -- I know you have your graphics here. The task that you just mentioned, would those be the tasks that we see at the top here?

A Yes. I've illustrated those. I just call them task 1 to N. Of course, N would be [redacted] in this case for this particular vehicle. The point of the slide is two things: First of all, to tell you that Toyota had an operating system in its cars, in its engines. And the other thing is for me to explain what an operating system is. You're obviously not running Windows in your engine. If you were, it wouldn't be able to reboot in 11 feet at 60 miles an hour.

So you are running a much smaller simple operating system. In this case, in this vehicle, it is called OSEK, O-S-E-K. And that operating system has a couple of jobs. One of those jobs is to provide helper recipes that all of the tasks need. The other job, which is critically important to the system, is it picks and chooses which task gets to sue the processor at any given moment.

There is only one processor, one main CPU. You have [redacted] apps running on it. So the operating system performs a bit of magic where it time slices and selects, Oh, task 3 for a while, task 4 for a while, task X for a while, task 24 for a while, task 2 for a while. And that selection process is really the main job really of the operating system.

And I wrote up here that inside the operating system it keeps what are called critical data structures. And what I mean by that is since the operating system's job is to keep track of all of this, it is like a taxi cab dispatcher sending out calls; it needs to keep track of its charges, its tasks. So I think it is useful to think about the operating system as being a person with a set of three-by-five cards.

On each three-by-five is written the task name or number, task one, and some notes about it like, Hasn't run yet, or hasn't run in a while, needs to run. Or task X currently using the processor. So -- and the operating system does its job. I have actually written an operating system and written about it in my first book and studied operating systems.

Inside it it is basically doing that data-keeping function, and so it is doing something like, Well, this is the three-by-five card I have on a pedestal of the task that is currently running. And this is a group of them that I sorted by importance that need to use the processor when it gets a chance. And then these over here, they don't need to run for a while because it hasn't yet been eight milliseconds since the last time it started.

So the operating system is shuffling these data structures around, these three-by-five cards. And if in the process the cards get mixed up, or some of the notes on the cards become corrupted, then bad things can happen to the apps that are running on top, the tasks that are running on top.

Q And as we go through this process, are you going to describe for us defects in the operating system?

A Yes.

Q Is the operating system an important part of the design of Toyota's ETCS, throttle control system?

A It is an extremely important part. It is like the columns that hold up a building in an architecture. So the choice of what kind of operating system to use, and the choice of how that operating system is structured is critically important to the integrity of the system. Yes.

Q We talked earlier about you have reviewed 2002 to 2010 vehicles that included Camrys, the Lexus ES and the Tacoma. Within that time frame, are there certain of those vehicles that all use the same operating system that Ms. Bookout's vehicle used?

A Yes. Many of them used the OSEK operating system. Many of them used the same exact version of the OSEK operating system which means exactly the same source code and ultimately the same machine code. And then others that used OSEK used a version number of one or two versions off

that are substantially similar from that point of view.
Q Here in your report that you did, did you do a chapter on operating systems?
A Yes. The first chapter is on the operating systems.
Q Did you provide a chart which shows which vehicles will contain the same operating system as Ms. Bookout's Camry?
A Yes.
Q Lets's go to the next slide.
A Don't worry. I don't expect you to understand everything that is on here.
Q You and I have looked at it before and I still don't understand.
A And you don't need to. This is just a representation of what we found with respect to those data structures, those three-by-five cards. So this is just a depiction of what we found inside the operating system when we looked at it to see how it kept track of which tasks needed to use the CPU and which ones and which ones were eager to do so, and which ones were using it.
And it has this three-tier structure that is actually the same between the two different ones called Itron and one called OSEK operating systems that Toyota has used in these electronic throttle vehicles. But you can think of these as three-by-five cards about a task, and this

would -- I was going to say you can think of it as the notes on a three-by-five card, but my analogy would break there.

So this is actually a sort of scoreboard, if you like, that keeps track of what importance the various things are that need to be done.

Q Is there defects in this operating system that you believe relate to unintended acceleration?

A Yes.

Q Take a look at the next slide.

A So it turns out that Toyota didn't look at this operating system. And inside this operating system when we looked, we found that these critical data structures aren't protected in any way.

Not only is there not a hardware protection against hardware random faults, but there is also no protection against either hardware faults or software faults, software bugs, causing corruption of this data inside the operating system. So you can actually see that this particular bit here that I flipped on the drawing from a one, which it used to be, to a zero, that will actually have the effect, a bit-flip there, will have the effect of killing one of the tasks.

And now that task -- depends on how the corruption happens, actually -- but one thing that can happen is that task will never run again until you reboot the car, which

generally speaking is it is taking the key and turning it off and turning it back on. If you have a push-button start, you actually have to get out of the car with your key on a remote before it will actually reset the processor.
Q Is that top line talking about a bit-flip where Dr. Koopman was talking about bit-flips but he was talking about from outside rays doing that?
A That is one way it can occur. Another way it can occur is by a software bug. And the software bug could be inside the operating system or outside the operating system. And it could affect more than one bit at a time. A hardware bit-flips that Dr. Koopman talked about and that NASA talks about are often called single event effects or single event upsets. And very often they effect just one bit.
But a software bug, of course, can corrupt a whole area of the memory or one bit or a collection of bits. And any corruption that occurs in here has the potential to kill one or more tasks, either temporarily or permanently.
Q You mentioned early EDAC. Does EDAC come into play if it existed with some of the things that your are describing here?
A It does and it doesn't. If there was EDAC, remember is like the parody bits, those hardware memory protections, if there was that, then we wouldn't have to worry -- might not have to, depending on how it is designed -- worry about

those single event electronic effects, EMI or Alpha particle strikes like Dr. Koopman talked about.

However, if there was EDAC, this could still be corrupted by a software bug. So EDAC alone is not the answer here.

Q And these things you're telling us can happen, how do you know that?

A I know that because we simulated it in the code room using the Green Hill simulator that Toyota used. And we also simulated it in the vehicle, in multiple vehicles, Camrys.

MR. BAKER: Your Honor, I know we're a little bit early, but we are about to transition into something that will take longer.

THE COURT: We will take our lunch break now. Ladies and gentlemen, it is 11:45. We are in recess for an hour and 15 minutes or until 1:00.

I would remind you: During the recess, do not discuss the case, and do not begin to form any opinions about the case.

All rise while the jury exits.

(Whereupon, the jury exits the courtroom.)

THE COURT: We're on the record. We're outside the presence of the jury. We're discussing the proposed deposition of Mr. Takimoto. The defendants have objected,

IN THE DISTRICT COURT OF OKLAHOMA COUNTY
STATE OF OKLAHOMA

Jean Bookout; Charles Schwarz, individually and as Personal Representative of the Estate of Barbara Schwarz, deceased; Richard Forrester Brandt, as Personal Representative of the Estate of Barbara Schwarz, deceased,	)	
Plaintiffs,	)	
vs	)	CJ-2008-7969
Toyota Motor Corporation; Toyota Motor Sales, U.S.A., Inc.; Toyota Motor Engineering and Manufacturing North America, Inc,; Aisan Industry Co., Ltd.,	)	
Defendants.	)	

* * * * * *

TRANSCRIPT OF PROCEEDINGS

HAD ON THE 14TH DAY OF OCTOBER, 2013

AFTERNOON SESSION

BEFORE THE HONORABLE PATRICIA G. PARRISH

DISTRICT JUDGE

Reported by: Kim Lewin, CSR

THE COURT: We are back on the record. The members of the jury are present, as well as counsel and their clients. Mr. Barr is still on the stand.

I was thinking, I didn't remember if I swore you in earlier, but I did. I remind you, sir, you are still under oath. And Mr. Baker, you may continue your direct exam.

MR. BAKER: Thank you, your Honor. Could you lower the light for us?

THE COURT: Yes.

MR. BAKER: Slide 19.

Q. (BY MR. BAKER) All right, Mr. Barr. We left off at slide 19, and I think we were about to transition.

You had mentioned, I believe, that you had done some software testing in the Code Room in Maryland, correct?

A. That's correct.

Q. And I think one of last things you said you mentioned you had also been involved with some vehicle testing?

A. Yes. I wasn't directly involved with the vehicle testing. I wasn't there when the vehicles were tested, but what we had simulated in the Source Code Room was the tasks could die and so the operating system by these corruptions inside the critical data structures. And some testing was done by a gentleman named Mr. Louden,

using 2008 and 2005 Camry vehicles.
Q. All right. And I think the jury's heard a little bit about that before. Were you involved in helping him do that process?
A. Yes. I was involved in assisting from the Code Room.
Q. All right. What was the purpose of doing the -- and I suppose they were software tests?
A. Yes.
Q. What was the purpose of running the software tests on the 2008 and 2005 Camry, generally speaking?
A. Well, the Source Code Review had indicated both that task could die by the memory corruption, and that also that one of side effects of that would be that this -- for example, that task died, that many of fail safes would be disabled. And so the purpose of vehicle testing -- in the room, of course, we didn't the real hardware. We could simulate the operating system, we could simulate the task to a certain extent running on the process server but it wasn't on the circuit board and it wasn't in the car.
And so that testing was to perform the same testing and demonstration to determine what the fail safes would do, if anything, in response to this task death.
Q. So Mr. Louden ran multiple tests on the '08 and '05

Camry?
A. That's correct.
Q. And all looking at how the software task made out?
A. That's correct.
Q. Was that reported in some fashion?
A. Yes. The testing that he performed, he used data logging equipment to record, you know, things like the accelerator peddle position, both sometimes outside the car, what it looked like, electrically.
And also inside the computer there was a tool that we had from Toyota called a tech stream. He was able to monitor certain memory locations inside the computer log. Ran to see, for example, whether the computer thought the brake was pressed, in comparison to whether the brake was actually pressed and things like that.
Q. Was the data that he collected from these tests compiled into some documentation that people like you could take and read and use?
A. Yes, in Mr. Louden's expert reports.
Q. All right. And have you reviewed the data and reports of failure relating to the test that was done on the '08 and '05 Camry?
A. I have.
Q. Have you considered that information as part of your analysis in this case?

A. Yes.
Q. In terms of talking about, from this slide, memory corruption and task death, have you pulled the piece of the data from some of testing that helps explain what you are talking about?
A. Yes.
Q. Is that the next slide?
A. Yes, it is.
Q. Let's look at that.
All right. The title here is Example of Unintended Acceleration. The first thing I wanted you to do is tell us what it is we're looking at.
A. Okay. So we're looking at a bunch of different pieces of data all plotted together in one graph. And just to generally orient you, elapsed time that is being measured across the bottom in seconds. So this particular piece of the graph begins at time 80 seconds on his clock and ends a little bit after 150 seconds, maybe 155 there.
And then on the vertical axis we see the speed of the vehicle. He was measuring that in kilometers per hour. And so we're seeing that in kilometers per hour. I've made some notes here in miles per hour to make it a little easier to understand.
Q. Is a plot of some of data that Mr. Louden collected

from some of his testing?
A.   Yes.
Q.   Can you walk us through it and explain to us what we're seeing here?
A.   Sure. I've tried to make clear what the different colors of the data mean. So for example, the speed of vehicle is this blue line. The throttle angle is measured here on this red line. And then there is, whether the brake is on and off is a binary signal, on or off. And so it looks like it goes way up to the top of the graph. It just really means the brake was not on, the brake was tapped and the brake is on solid.
Q.   Just so I'm clear, where we see these intermittent green lines, is that somebody tapping the brake?
A.   That's correct.
Q.   And when we see up here at the top, it's a long piece. That means the brake is applied at hilt?
A.   That's correct.
Q.   Okay. What were you simulating in this?
A.   So you can see there is a vertical line here at time, just before 100, maybe 98 seconds. And that is the marker for the point in time it tests when this task-X was killed and the mechanism of killing it was to flip one bit inside the operating system. So those working inside the Code Room indicated particular bit to flip to

Mr. Louden.
Q. All right. Let me back up and ask you additional questions. In this testing that was done on the vehicle, was the test required to go in and simulate some occurrence in order to have task-x data?
A. I'm sorry. I don't understand your question.
Q. Did it have -- did the person that run the test have to make the task die?
A Yes. So using the same tech screen, laptop basically as Toyota test equipment hooked up to the car's computer, he was able to simulate the bit flip. Of course we can't -- you know, as scientists we want to test something, we need to be able to make it happen, we need to make it happen in no time. We can't just wait around for that particular bit to flip, which may take a long time.
So he was able, using that same computer, to, you know, enter a command and cause that bit to flip. And then that would have the effect of killing that task in the vehicle. And then the rest of data is the data collection of cars's behavior around then.
Q. Does he drive this car on the road?
A. No. He's doing it on what's called a dynamometer. In Maryland, anyway, when you get your car's emissions tested you put your car on a dynamometer, where the front

wheels -- the drive wheels are turning and the car's not going anywhere. He had a similar arrangement.

Q. All right. And so, this vertical line, I'm estimating, is somewhere close to 100 seconds into that test, he's able to, using a computer, to kill task-x?

A. That's correct.

Q. When you say kill task-x, what does that mean in terms of the car's operation?

A. Well, the graph is showing that at that time you have [redacted] of the [redacted] tasks alive, but you don't have this task-x running. And we're seeing what happens to the vehicle, which is a loss of throttle control subsequent to that.

Q. And in a previous slide when you were talking about memory corruption, killing task-x and causing a UA, is that an example of that?

A. That's correct.

Q. Tell us what happen after the task was killed.

A. After the task was -- so the setup here with this particular test was that the car had been run in the time, obviously, before 80 seconds and using the accelerator pedal, Mr. Louden had gotten the vehicle up to this 68 miles an hour and he had set the cruise control. So now he had the car driving at cruise control at 68 miles an hour.

And then he canceled the cruise control and a little bit later here at this inflection point, the bottom of blue line, he hit the resume button on the cruise. So it'd try to go back to the speed of the vehicle that was previously set, which was about 68 miles an hour.

So if it starts at -- I didn't calculate there in miles per hour, but you can see the inflection point at the bottom in the blue, it starts below 68 miles an hour. And then of course, the car begins to accelerate because the car is operating normally.

What happens is that the task death caused in this particular test. Because that task was not there when the vehicle actually reached the set point of 68 miles an hour, it should have closed the throttle more and slowed the vehicle -- or not slowed the vehicle, but kept the vehicle going at 68 miles an hour. Instead, the throttle remained open and the vehicle continued to accelerate.

And you can see that this total length time with the throttle open, letting in air, and the car accelerating to past two and past the cruise set point, is approximately 30 seconds. So from time, about 100, until a time, about 130.

Now, Mr. Louden, as I understand it, at this point got nervous at 90 miles an hour because the vehicle was on the dynamometer. And so at that time he pressed on the brake solidly and continuously this whole time.

There's a couple of effects you should be aware of because it was on the dynamometer. First of all, is that on a dynamometer, there is a lot of momentum in the dynamometer itself. So when he started braking there and a fail-safe, called a brake echo, kicked in, at that time the vehicle did not decelerate as fast as it would have on the road.

But what we see here is that there was an unintended accelerate or a loss of throttle control that spanned from time 98 until about time 129 when he pressed on the brake solidly at that time.

Q. You mentioned at -- was it at this point that the fail-safe kicked in with the brake applied?

A. Yes. The -- at -- it would be within that, between that 129 and 130-second gap.

Q. All right. So we see in some of these green lines, he just taps the brake and the fail-safe did not come on?

A. Yes. That's correct.

Q. All right. And now, this is also from the 2008 Camry?

A. Right. So this was the first testing that was performed was on a 2008 Camry.

Q. You mentioned earlier that you had looked at other cases or been involved in other cases, correct?

A. Yes.
Q. One of them was called Van Alfen?
A. That's correct.
Q. And I think the jury heard about that one. Another one was called St. John. Were you involved in that one?
A. Yes.
Q. In St. John, it involved a 2005 Camry?
A. That's correct. Same model as this case.
Q. In both cases, were you doing the same analysis that you're doing here?
A. In terms of the overall analysis?
Q. In terms of looking at UA?
A. Yes.
Q. And evaluating the software code?
A. That's correct.
Q. All right. Was Van alfen the first case in which you had an opportunity to perform this type of analysis?
A. Yes, it was.
Q. And in that case, did you write a report for the Court that outlined your opinions in that case?
A. Yes.
Q. And were St. John and Van alfen pending in what the judge has already told the jury, was an MDL or big federal litigation in California?
A. Yes, that's correct.

Q. All right. And were both cases being supervised by one judge?
A. Yes.
Q. Judge James Selma?
A. I don't know his first name. Judge Selma.
Q. All right. Very well. After you wrote your report in Van alfen, did you come to realize that you had made an error relating to the brake echo?
A. Yes.
Q. All right. Tell me about that.
A. Well, at the time that I wrote my report in July of 2012, in the Van alfen case, I did not understand that a portion of this behavior that occurred right here was a fail-safe in the second CPU, in the minor CPU. And that was, in part, because Mr. Louden did not realize that the throttle had been cut at 129 there. He saw the engine stall at 132.
And additionally, it related to some source code that I had been provided just in the final weeks of my report writing. And that -- I made an error in my analysis of that code the first time.
Q. And once you realized there was an error, did you go back and look at it?
A. Yes. As soon as I became aware of that, which was in late September of 2012, within 10 days or so, I issued

a supplemental report, reviewed the additional code and filed it in the same case.

Q. So you ultimately corrected your error?

A. That's correct.

Q. And the source code that you were looking at when this error occurred, was that the source code from what we've called the monitor CPU?

A. Yes. The ESPB-2 monitor CPU.

Q. And in the time frame there where you were looking at it, had there been a delay in producing that software code from Toyota?

A. Yes.

Q. Was there also a problem with getting the proper tools, and I may be using the wrong word, to read it?

A. You're not using the proper terms. The source code for that ESPB-2 chip, despite being asked for much earlier, had not been produced until about three weeks before my report deadline. That was about -- that was in late June. So that was about -- almost six months after the rest of the source code for the main CPU had been produced.

And so I had -- while I was preparing, of course, my full report, which is about the same size, to analyze this new code that had come in, within about three weeks, and write a report on that.

And additionally, Toyota only provided the source code and they did not provide the tool that went with it, called the assembler. This code is written in assembly language, which is a harder to read human source code language, more machine-like. And they were using one that needed a special compiler called an assembler and they didn't produce that or it's user manual. And so I erred in my analysis on the basis of not having that manual or that tool.

Q. All right. And it wasn't an error in reading the code. You just hadn't read that part of the code yet, is that right?

A That's correct. The error related to something called a preprocessor directive which stemmed from not having the -- I made a logically reasonable decision and I consulted with my colleagues on making that decision. But without that actual tool we didn't have a definitive answer.

Q. And did Judge Selma ultimately allow you to supplement your report?

A. Yes, he did.

Q. And did he ultimately conclude that part of the reason that you reached that error was due to a delay in production of software by Toyota?

A That's correct.

Q. Is there anything else about this particular slide you wanted to tell us?
A. Yeah. I just wanted to -- this is one example from the vehicle testing. And I just want a make a few points about and it foreshadows some of other things we're going to talk about.
First of all, is that this testing, although it was done on a dynamometer, is representative of what would happen in the vehicle on the road if you resumed cruise control and task-x was dead at the time. It would exceed the speed of your planned -- you know, set speed. And it would not, in this particular scenario, begin to correct anything until the driver acted.
So the driver would have to realize that the car had gone above the 68, maybe much above the 68. And then when he stepped on the brake an action was taken in that particular scenario.
This testing confirmed that -- so this was related to cruise control. But we've also confirmed that during this time the accelerator peddle is not responsive. So there is two ways you can tell the car how fast you want to go, one is the cruise control buttons and one is the accelerator pedal. And neither of them works during this dead task-x time.
The other thing is that this ended, this particular

test ended when the driver stepped on the brake. However, we have confirmed in other vehicle testing that I'll talk about later, that if the incident begins with the peddle, brake peddle pressed at all, even lightly, then the unintended acceleration will continue, potentially, forever unless the driver tries the risky thing of letting go of the brake while the car is driving away with him.

Q. So in other words, if you're driving down the road and you put your foot on the brake to slow down, for whatever reason, during that time period task-x is where it actually dies, the vehicle starts to accelerate. You've got to actually back off the brake and try and catch it?

A. That's correct. Which is both counter intuitive because your car is zooming away and you have to let go of the brake. And it's also dangerous because as you let off the pressure of the brake, at least you were applying some mechanical pressure, but as you let off the car speeds up. And so that may increase the risk in the short term, at least, before this fail-safe would take effect.

Q. And your foot on the brake, as you described, and your vehicle begins to accelerate while you're coming back off the brake, does that actually give you the

impression that the vehicle was accelerating?
MR. BIBB: Objection. Leading.
THE COURT: Sustained.
MR. BAKER: I'll move on, Your Honor.
Q (BY MR. BAKER) Have we covered this slide?
A. Yes. I think so.
Q. All right. We talked about memory corruption. Is this talking about it in any particular way?
A. Yeah. So we've talked about the memory corruption that can happen and we've talk about some of the effects that that can have.
What this talks about is different ways that the corruption itself could happen, different types of software bugs, probably more detail than you wanted to know, but I wrote a whole chapter called software bugs in Toyota's code, and this slide summarizes the types of bugs that were found in Toyota's code that could cause bits to flip to memory tp become corrupt.
Q. Could you describe each one of the for us as you have listed here?
A. Yes. The first type of software defect is a buffer overflow. This is where you have a region of the memory, let's say 100 bytes of space that's reserved for a particular buffer storage area.
If the software contains a bug that writes past the

100 bytes, say, 101 bytes, that will obviously override whatever is the next thing. I think Dr. Koopman gave an example where, you had a notebook and you got to the back of the notebook and you accidentally wrote on top of the other pages. It's that kind of thing where you have another variable or another thing being stored and your code accidentally overwrites it and now it can take on a new value. So that is a buffer overflow.

Q. All right. What is -- and do you find that to be a defect in the '05 Camry?

A. Yes. The 2005 Camry code contains at least one buffer overflow.

Q. Now, what's an invalid pointer?

A. An invalid pointer D reference is if the -- quite technical. But if you have in one cell of your spreadsheet the information about the location of another cell in the spreadsheet. Let's say, on your spreadsheet it says cell E-5, a pointer is like that. In the source code it says, I'm not what you're looking for, but here's the address that you're looking for. That's a pointer.

If you, instead of going to cell E-5 you accidentally to cell A-1 because you've used the wrong pointer, then you will write over somebody else's memory, some other part of the source code.

Here, the defect in the 2005 Camry is that there are many places where pointers are de-referenced without checking them to be valid. And that is something that's important to do in the safety critical system, is to check that they are valid.

So if, for example, one of those pointers became corrupt, then it would cause a chain reaction of additional damage to the memory.

Q. What is a race condition?

A. A race condition is a subtle timing bug. Toyota uses the term task interference, and on that basis NASA also refers to task interference. You heard about the 10,000 global variables. You can imagine that one of those global variables is the balance in your checking account. Suppose there is two of you who each have a checkbook or checks from that account. If you are near the bottom of your bank balance and both of you write checks, you're going to end up with an overdraft condition in the bank, but also it's not clear -- there's a race it's not clear which one of you is going to get to clear the check and which one of you is going to get to balance the check.

There's something similar that can happen in software which is that you have two or more tasks, like two tasks as having two checkbooks and they're both referring to the same both global variable or same cell in that spreadsheet, and if they're both writing at close in time, they can actually step on each other's toes. And it could be that one of them gets its answer there, or the other one gets its answer there, or together they corrupt and damage it to create a third value. You don't get a check balance, in this case you get a corrupt memory.

Q. And you found that defect to exist if the '05 Camry?

A. Yes.

Q. Nesting schedule or -- nested schedule or unlock, what does that mean?

A. A nested schedule or unlock is a very bad thing. The use of -- it's complicated to explain again. I promise this is a last slide about these things. But in nesting schedule unlock is, one of the ways you prevent corrupting these data locations that are used by multiple tasks is you tell the operating system, hey, while I'm updating my checkbook balance, don't let the other -- it's like calling your friend and saying, don't write a check, I'm about to write a big check for the rent. That's a version of that in the software where you tell the operating system, while I'm doing this, don't let any other tasks switch and take over. Let me finish my job and then when I'm done then you can give the processor to someone else. You ask the processor please don't let any other task to run until I'm done, and then briefly, maybe just a few instructions later you tell the operating system, okay, I'm done updating that variable, it's okay for other tasks to run. It's called a scheduler or locking.

It is a bad practice that is in Toyota's code to lock the scheduler, tell the operating system to lock, and then a short time later lock it again. And it's particularly dangerous with the operating system that Toyota's using because when the first of those to finish unlocks, it's like someone going to a deadbolt on your front door and you lock, someone else comes along, locks it again, no change, right? But the first one of you unlocks it actually chances the security state.

The same thing inside the operating system, if you have nested call to the operating system to lock, the first unlocker is going to create race condition. It's going to create an opportunity, a time window through which race conditions can occur. It won't happen every time. If it happened every time, it would get in the vehicle testing in Toyota's factory. It happens rarely because it's a subtle time-related bug. It depends on sort of the stars aligning in a bad way. And those kinds of bugs are exactly the kinds of bugs that I'm using to looking for and finding in imbedded software. And we

found those types of bugs in Toyota's code also.
Q.   Can any one of these cause memory corruption?
A.   Yes, any one of these besides themselves can cause memory corruption.
Q.   The unsafe casting?
A   So unsafe casting is where numerical values can become inadvertently rounded and take on new numeral values. Give you an example of this. One of the bugs related to this is in Toyota's code. It is possible for the software on the main CPU, although the actual car is supposed to have its engine move -- you know about RPMs and tachometer, the car is supposed to go from zero RPMs, revolutions per minute, to in this case a maximum of 6400 RPM, that's the red line. We're above the red line where it will stop. And along the way there is all these different values of RPM. Well, those are okay in the software, but due to a casting bug it is Toyota's code, is is possible for that value to be become negative and there's something like 100 parts of the code, that look at the end engine speed and they could become very confused if the value went negative. And it could also become very large like 13,000 RPM which could confuse the software in a different way.
Q.   And last one stack overflow?
A.   So a stack overflow is a very dangerous problem

where, it's like a buffer overflow but it's a very special buffer. It's a buffer that all the tasks use to keep traffic of their internal decision making and keep notes for themselves about what they were doing when the processor was taken away from them the last time.

And that stack is of a set size. In Toyota's it's about four kilobytes, very small region, and if that buffer overflows then you end up overwriting whatever's beyond it in memory.

Q. All of these defects that you found in the '05 Camry?

A That's correct.

Q. And all of them can corrupt memory?

A. That's correct.

Q. Why is that memory corruption is so significant?

A. Well memory corruption is so significant because it's a memory corruption that can cause a task death and task death can cause in a general sense unpredictable results, but in a specific sense, as with task X, cause loss of throttle control and also a disablement of a number of the fail safes.

Q. We talked about earlier some of your books. And one of them was a dictionary. And was that an effort to define certain terms that are used within the software industry?

A   Yes.
Q.   Is one of the terms that you defined in your book spaghetti code?
A.   It is.
Q.   Let's go to the next slide please. Tell me generally what spaghetti code means in your industry?
A.   Well, in a nutshell it means that the code is very difficult to read and maintain. You heard Mr. Ishii say that NASA had trouble reading Toyota's source code. That wasn't to do with them not following NISRA, it's because it was badly written and badly structured source code. And that's spaghetti. Code spaghetti code is -- I picked this picture of a very complicated electrical wiring intersection because I think it aptly demonstrates what spaghetti code is like.
     Now I have to look at source code and I'm looking at this variable name and this function name and things of that sort, but imagine your job was to go fix the phone line that's out at Apartment 12 in that configuration. That's what spaghetti code is like. And when you go and find it, you may disrupt or you might tangle two wires together and cause the phone service to break in another department. And that's what toyota's engineers are dealing with their source code and that's what they're referring to when they call it spaghetti like.

Q. When you say source code that is developed at one time and then you continue to add onto as time goes on, rather than starting anew, can you end up with a spaghetti code?
A. Yes.
Q. Do you have any idea if that's the case in terms of Toyota?
A. Yes.
Q. What did they do?
A. The way I understand the progression with Toyota is mostly through what I see in the source code evolving from year to year, and also what I read, for example, from Mr. Ishii's deposition. I've read more of it than what I saw here. But he talked about the time frame. He was there the whole time.
Initially they didn't have electronic throttle control, they didn't have an operating system, they didn't use the C programming language. They switched from assembly language to C. They added an operating system. They added electronic throttle control. And they were all the while increasing the amount of complexity and intertwinedness of all this source code.
Q. You were here for Mr. **Something deposition this morning?
A. Yes.

Q. And did you hear the discussion about one of the documents between NHTSA and the Toyota employees about updating the power train software?
A. I did.
Q. Have you actually reviewed that document?
A. I have reviewed that document, that's right.
Q. And in that discussion did Toyota employees refer to their software as spaghetti like?
A. Yes.
Q. And did you create a slide about that?
A. I did. So these are all quotes from that document.
Q. And here it discusses activities to improve the status like -- the spaghetti like status of engine control application were started, is that correct?
A. That's correct.
Q. Is this the type of software that's used to help control the electronic throttle control system based on your review of the document?
A. Yes.
Q. Is there anything else you want to point out in terms of this document?
A. Well, the document refers to first of all that the power train engine code is -- which is another name for the UCM engine control module. That's where the power comes from. It also refers to other problems with

Toyota's process, such as that there are some of the C source modules don't have specifications and have specifications -- specifications or design documents that say how it's supposed to work. In some cases the design documents don't exist, and in other cases the design documents say something different than the code, so which is right?

Q. All right. Let's go to the next slide. Are there several type of spaghetti code?

A. There is two basic types of spaghetti in source code. One is what I'll call data flow spaghetti, that really refers to having all the different, you know, those couple of thousand modules, files of source code all being interconnected with each other, which is a bad architecture, through global variables.

For example, so when NASA says that -- and I can confirm that Toyota's source code has over is 11,000 global variables, they are saying that it is greatly intertwined in such a way that spaghetti -- the data -- if you want to follow a particular path, you know, where does the accelerator signal go, you have to trace through multiple files and multiple tasks to see where that data goes. And they're all linked together with these global variables. Some of which are 25, 30 characters long and some don't have vowels and some -- two of them are

identical, except one has a P and one has a D, or a P and a B.
Q. And just remind us, what is a global variable?
A A global variable is one of those ingredients in the recipe, but it's being used by multiple recipes. So an example of that would be the global variable that tells the combustion part of the software how wide open the throttle should be, should it be 10 percent open or should it be 100 percent open. That's a global variable.
Another global variable is the one that I referred to earlier that keeps track of how fast the engine is going. Is it 2,000 rpm and 3,000 rpm. And when those are being referred to from multiple places, not only is it spaghetti, but also increases the probability of chance of race conditions and task interference.
Q. Is there in your industry a standard for how many global variables you should use?
A. Well, it's not an absolute science with that. Certainly, you should not be using 10,000. Certainly you should not be using 1,000. The academic standard, as Dr. Koopman said is zero. In practice a small number of global variables may exist in some well structured programs, but generally a very small number.
Q. And what is the next type of spaghetti code?
A. There is also control flow spaghetti. So here the

spaghetti that you have within a recipe, it's greatly internally obligated. That's like picking up a recipe book and you can't follow it. You can't figure out, you know, what am I baking at this point, what step am I on. That happens sometimes in source code when one function -- remember, we looked at function earlier, a larger of one function, instead of fitting on one PowerPoint slide, takes 20 pages of printout just to look at that one function, and inside there's all these different cases and ifs and tests and looking at this variable, looking at that other variable. It's like a very complicated recipe that you're not sure what you're going to get when you get to the other side.

Q. You got down here at the bottom you talked about testability and then you talk about scoring of greater than 50. The greater than 50, what are you referring to?

A. So, I wrote a report chapter called Toyota's code complexity in which we produced a large number of tables using some static analysis tools to tell us how complex is each function that's in the source code. So the tools give a score and it's based on the number of different ways you could possibly go through that function. The number of different sub recipes you might imagine. So the number of different possible recipes you can make from that one.

And this actually is something that is useful to software developers generally. If you are, like Dr. Koopman talked about going to a company and assessing the quality of their product. If you run a tool like this and it spits out code complexity numbers for each function that will direct you to the ones with the highest score are the ones most likely to contain bugs. And so if you're hunting a bug, one of the things you can do is go and clean up those parts of the code.

And my organizations that I've consulted with maintain a practice where they will not release a product if it has a code complexity of any function bigger than, a typical number is 30. Toyota's code actually has 67 functions that score over 50, which has been assessed as an untestable score. What that basically means is that this one little recipe within this bigger complex electronic throttle control system, just to test that one little recipe in the factory when you make the car, you would have to test at least 50 different vehicle states and software states. You would have to test all 50 and you would have to have a detail documented plan that said, here's what I'm going to do to test path one. Here's what I'm going to do to test path two. Test path three, et cetera.

And there are actually design techniques and processes called code coverage analysis. Where you try to make sure that the test you run on your product are actually going through every one of those lines of code and every one of those possible halves. I see no evidence that Toyota did that. And particularly not for these untestible functions.

Now, within those 67, there are 12 in the 2005 Camry that have over 100, which is assessed at a level of unmaintainable, which basically means, if you read the papers, that above 100 it becomes so difficult to go in and fix a bag, that every time you fix bag, you make a new bug. So you've got this very buggy code, it's hard to test, and you go in and make a change and you break something.

And one those 12 unmaintainable functions is the approximately 1,300 line functions that performs the calculation of mathematics to decide how open to make that throttle. And that's an area that NASA was very interested in, and in fact tried to simulate and could not simulate to its satisfaction and found that Toyota not only did it not have a test plan to test all 146 paths through there, but did not also have a simulation of it like NASA wanted to run.

Q. Throttle angle function, is that the function that determines how open the throttle's going to be while

you're running the car?
A. That's right. That's the function that takes as its input the accelerator contribution, the cruise control contribution, the idle speed contribution and all the other subtle ways that the throttle need to be trimmed are all taken into account in that. To produce one, ultimately one angle, like 50 percent or 30 percent.
Q. Does that function have to work with task X in order to run the car?
A. Yes, that function is executed by task X. It's among the kitchen sink of things that it does.
Q. Let's go to the next slide. You mentioned stack analysis earlier, is this a more detailed explanation of that problem?
A. Yes. We're going to talk about stack.
Q. Let's start at the top?
A. Okay. So we did an analysis of Toyota's stack. And the first thing I should probably explain is what a stack is. So I mentioned that that if these [redacted] tasks and they're switching back and forth taking turns with the processor, the stack is both how when we are running they pass information from one recipe to another. If one recipe calls larger of -- to compute the larger two numbers, it passes information through the stack and gets the results back through the stack.

But then also it holds that information on the stack in memory in that area temporarily while the processor runs a different task and then switches back. And so this stack is a very important data structure that is used by all the tasks. And the operating system allows them to use it.

And the programmers have to pick the size of it. It has a finite size. It's just block of memory, contiguous block of memory.

So actually in Toyota's design for the 2005 Camry there are two stacks. One is a stack on the right that is specific to task X; the other is a stack on the left that is for all the other tasks. And also there is also something called interrupt service routines. Kind of like tasks, they complicate my explanation, so I'm mostly ignoring them, but they are abbreviated ISR for interrupt service routine. And you see those reflected there as well.

And so what you have on the left is a depiction that every moment in time in the car's operation the stack has a fixed bottom address, and some processor and some designs it's zero in memory, and then is has a fixed top address or end address. And in this case I've depicted it as growing up to 4K, 4096 bytes, and then it also as a current address or a stack pointer, which points to where the system is on that.

And so we performed an analysis called a worst case analysis, which is a process whereby we assess if all the tasks are using the stack simultaneously, which can occur from time to time, will they explode beyond the stack potentially and overwrite what's passed it.

NASA was interested in this subject and Toyota provided them an answer, which was that the stack was only utilized at a worst case of 41 percent, 1,688 bytes.

What Toyota didn't know apparently, and NASA understood, is that -- NASA misunderstood therefore, is that the actual worst case is 94 percent. And that's not including something called recursion. NASA's spent a great deal of time talking about Toyota's use of recursion, and which could because the stack to overflow.

And in fact, we don't know how much memory could be consumed by the recursive function -- recursive function is a recipe that culls itself. Like in order to compute the larger of 67 an 65 let's cull ourselves on the larger of 66 and 65. That's not how that function works, but if you can imagine if it culled itself, it could do it many times. And there are some recursive functions in Toyota's source code, which is not appropriate in a safety critical system.

And the NASA report reflects that inappropriateness,

but NASA did not realize that the recursion was on top of 94 percent. They thought it was on top of 41 percent. Making matters worse, if the stack overflows in the 2005 Camry, the next thing in memory is the critical data structures that are not protected inside the operating system. To if you have a rare stack overflow, the first thing that is going to get damages are those 3 by 5 cards that tell the operating system what to do.
Q. So if in using this the tasks end up running past the allowable memory, it then moves into what memory is being used by the operating system?
A. That just keeping on scribbling.
Q. Does it overwrite things that are going on with the operating system?
A. Right. Those are the critical data structures like the three tears of keeping track of what's going on with each task and which task to run next.
Q. Does that cause memory corruption?
A. Yes. Obviously, the stack overflow itself causes memory corruption. The corrupted data is this unprotected operating system data and a side effect of that can be task death.
Q. Is there any other information we need to know about this slide?
A. Yes. Specifically recursion violates a MISRA C

rule. So had Toyota followed MISRA-C, which is an automotive industry subset of the C language that's safer and specific for the auto industry.

In 1998 that standard had a Rule No. 70 called -- I don't remember the exact language. But function should cull themselves. And the rules basically are the same in 2004 but they changed the numbering system, so in the 2004 standard this rule, same rule is No. 16.2. So this is a violation of the MISRA C rule.

Q. Does the violation of this rule related to unintended accelerations?

A Yes.

Q. In what way?

A. The stack can overflow due to this recursion in the 2005 Camry.

Q. And create memory corruption?

A. And that would create memory corruption, that's right.

Q. What was NASA's view about this recursion?

A So NASA's view, NASA was concerned about stack -- possible stack overflow. They had a couple of pages devoted to it, about five pages. I pulled some quotes here. Recursion could exhaust the stack space leading to memory corruption and run time failures that may be difficult to test -- detect in testing. The question then is how to verify that the indirect recursion present in the ETCS-I does in fact terminate and does not cause a stack overflow.

And then the third one, the CVO in the ETCS-I does not have protective memory and therefore a stack overflow condition that cannot be detected precisely. Overflow would cause some form of memory corruption. And I should just stop there.

When is NASA referring to protected memory here, they're not referring to EDAC, they are referring to something called run time stack monitoring, which is a technique that software developers use to make sure that -- it's like a flood marker on a river. When the river rises and gets to the flood mark, you know there is going to be trouble and you start activating.

The same thing is a technique that is well-known and used for a long time by imbedded software developers, which is you make an area of the stack that you watch and you see if it gets corrupted. a common thing to do is write all ones to it, or some binary pattern, and you have a part of the software that is monitoring to see of the high watermark has been breached. And if it is, you know that you might get into dangerous uncertain territory, and so you can do a safe shutdown or reset the system to get past that.

Q. Is it the memory corruption that's talking here about that can cause an UA?
A. That's correct. NASA didn't know that the memory just past the stack was the operating system, as far as I know.
Q. Are we through with this? No.
A. So NASA also says it's not clear what impact incursion has with respect to the larger UA problem. There are other sites of recursion that we haven't and analyzed.
Q. So they just didn't look at it?
A. They looked at some, they took Toyota's word on some, and they didn't analyze the rest. And NASA didn't ever know that there was so little safety margin. So Toyota's answer to NASA about incursion included that they had -- Toyota said they had added an extra margin of safety more than double the 41 percent. So Toyota's answer to NASA is, don't worry about it, we've added a margin of safety of more than double. But the truth is that margin is not there. And toyota itself didn't even realize this.
Q. Let's go to the next slide. And we talked a little about the some of the things you found. What are some of the stack mistakes?
A. So the first big mistake that Toyota made here, is

that -- and this is why it's not 41 percent, it's 94 percent. Is because Toyota didn't do a thorough analysis. When they did their own internal analysis of the stack to come up with the number 1,688 bytes, they missed a bunch of stuff.

The one that accounts for the most extra bytes is the operating system itself. The operating system, every time it's switching from one task to another, it stores data on the stack, so you can't just add up the worse task themselves, because when they are running or all is have stuff on the stack, you also have all the operating system changeovers between them, as well as interrupt service routines. And Toyota missing that is the biggest factor in why it was 94 percent, not 41 percent. But they also missed about 350 functions. They had some mistakes in their attempt to automating the rest that we found as well.

So actually the 94 percent is the most we found. It's possible that the stack could go beyond that as well.

Q. Let's go to the next one.

A. On top of that Toyota used dangerous recursion. So I showed some quotes from the NASA report. Here's another quote form the NASA report. It says, "Absence of recursion is standard in safety critical imbedded systems." And I would agree with that. It is not appropriate to recursion. And MISRA and NASA and I and Dr. Koopman all agree on that.

Q. But Toyota has it?

A. Toyota has it in the 2005 Camry, that's correct.

And finally, Toyota didn't perform run time stack monitoring. This, by the way, is in the cheaper 2005 Corolla that was supplied to Toyota by an American supplier named Delphi, which is different than Denso, the Japanese supplier. So Denso is supplying 2005 Camrys and it doesn't do any run time stack check monitoring, but Delphi is supplying 2005 Corollas because at the time of partnership of the Corolla being manufactured with GM in California. Delphi supplies that and Delphi one, although it has many defects as well, the stack overflow is not a possibility in that particular design, as I understand if.

Q. Okay. Next line?

A. Toyota also failed to comply with a number of standards, including the standard for its own operating system. So it used an operating system that it got from its chip vendor NEC. They supplied the processor and they also supplied an operating system called RX OSEK 350. The processor is the V850, this was an operating for that processor called RX OSEK 850. OSEK is a reference to an international standard API, which is a programming interface. It's kind of a software term that means how you control the operating system. What the function names are and things like that.

At any rate OSEK came out of the automotive industry in Europe and this was -- a market was created where multiple operating system suppliers could provide compatible operating systems. So that from the auto maker's point of view, they could switch from one to another and they would still be using a version of OSEK. And the idea being that those operating systems would then compete on quality, compete on the price, et cetera. And in order to make sure that the car maker's code would work on any one of these, there were a set of compliance tests set up to make sure it was truly an OSEK.

And only operating systems, when you read the documentation, that have been tested are supposed to have OSEK, they are supposed to say OSEK is a trademark and that sort of thing, so they are supposed to be tested. We found that the one that Toyota used was not in compliance at all. And actually, at that time, by 2002 there was a compliant OSEK available on the market for that very processor, but Toyota for reasons unknown to me, chose to go with one that was not certified as compliant.

Q. This particular operating system RX OSEK 850, is that also included in some of the other vehicles you looked at, like the Lexus ES, certain model years Toyota's V6 Camry?
A. That's correct.
Q. Let's go to the next slide.
A. Toyota also failed to comply with standards, and here we heard from Dr. Koopman about a higher level concern about safety process. That's not what I'm referring to here. Here I'm talking about, for example, the MISRA C guidelines.
Q. That is the smaller book, right?
A That's the smaller book that is very specific on the C programming language. So the big book says you should use a documented subset of a language that is safer. And the little book is those -- that subset, those instructions.
And by 2004 when they updated this, they wrote in the book that this was being widely adopted in multiple industries, they didn't expect it to be used outside of automotive, but they are very happy it was. And also that in 2004 when we were updating it, or prior to that, they had worked with the Japanese equivalent of what here we call the Society of Automotive Engineers, which has standards and has conferences for automotive engineers,

obviously. That is the Japanese Society of Automotive Engineers and the Japanese Automobile Manufacturers Association. They participated in the drafting of th second version of this. And indeed, one of Toyota's own employees was thanked in the contribution.

Q. That was put out in 2004?

A Well, the original standard was in 1998.

Q. And are you talking about, does that relate to the original one, or the one that came out in 2004?

A. Well, the 1998 one was the first version that MISRA, Motor Industry Reliability Association of the United Kingdom published, and then these comments from the 2004 addition of that.

Q. And in the review of what Toyota had done did NASA fine any violation of these codes

A. Yeah, NASA found a number of violations of MISRA rules.

Q. Did you find violations?

A. Yes. NASA looked at about 35 of the rules. There's in total, I forget the exact number. It's basically the same set of rules in 1998 and 2004. But as I recall, it's over 100 rules total. NASA looked at 35 of them and they found over 7,000 violations, and they reported that on page 29.

I checked the full set. There were a couple that were difficult to test, but basically the full set and found more than 80,000 violations in the 2005 Camry.

Q. There was also a discussion about compliance with MISRA rules that we heard from Mr. Ishii, I think he said something like maybe 50 percent of compliance of used MISRA rules. In your code review did you find that to be true?

A. No.

Q. Was did you find?

A I actually wrote on whole report on Toyota's coding standard in one of my chapters, and what I found studying their coding standard was that actually -- the MISRA rules are over 100 rules and the Toyota rules -- I have an appendix that lists them all -- I think it's about the same number, about 100, maybe 119, but only 11 of Toyota's coding standard rules overlap with the MISRA C rules. And interestingly, five of those rules are violated in Toyota's code.

So when they say 50 percent overlap between the two, our rules and MISRA rules, no.

Q. Do you know the percentage on how they actually match up?

A. Just different ways of calculating the percentage. I couldn't make any come anywhere near 50 percent. They moistly shake out around 10 percent.

Q. Did you also review some work done by a Toyota employee names Mr. Kawana related to his development of how to look for bugs in software related to rule violation?
A. Yes, I did.
Q. Tell us about that.
A. So there is a paper by Mr. Kawana that was presented in Detroit in 2002 and also a presentation that was made in San Diego in 2004. They both contained this bug chart, so I pulled that slide from the presentation in San Diego. And this is showing that in Mr. Kawana's view, and these slides are also bearing the Toyota logo, it's reasonable to estimate the number of bugs using the number of violations. And the standard he looks at -- to for what's a violation is MISRA C. And this is the same Mr. Kawana who I see thanked in the MISRA 2004 documents, so he was clearly participating in the update of MISRA in some fashion, and around the same time he has presented this at an automotive industry conference that suggests, at least to me, not knowing otherwise, that Toyota is complying -- that Toyota's viewing MISRA C as a appropriate -- the number of violations in MISRA C an appropriate way to estimate the number of bugs still in the code. It's called bug population estimation. People do the same thing with counting fish in a pond. You can

do things like count some and mark them and throw them back. There's different ways of doing estimation techniques of how many fish are in the pond. Here's a technique that industry can use to estimate how many bugs there are out there. But this is based on the 2002 paper on past Toyota projects.
Q. This is Mr. Kawana's bug chart?
A. That is Mr. Kawana's bug chart.
Q. And on this bug chart they've got 30 rule violations. Does that indicate that you do have bugs?
A. Yes. In his calculations, there's 30 rules violations, there will be one major bug and ten minor bugs.
Q. There's also been testimony that -- and you heard part of it from Mr. Ishii that Toyota had its own internal coding standards?
A. I did.
Q. Have you reviewed some of those standards?
A. Yes.
Q. In your review of the source code, were you able to determine I some of those were violated?
A. Yes.
Q. Let's take a look at that?
A. So Toyota maintained an internal set of coding rules. They may have had multiple coding rules, but this

coding rule was specifically for 32 bit processors, which is what's in the V8 50 main CPU, written in the C language for the power train. So it's referring to the ECM that I analyzed code for.

And what I found is that, first of all, Mr. Ishii's statement that 50 percent of them overlap with MISRA is way off. I also found that at least about a third of Toyota's own coding rules are violated. So they weren't enforcing their own rules.

Q. Would that have been the source code for the 2005 Camry?

A. It was the source code for the 2005 Camry. And that's all documented in my chapter on the Toyota's coding standard.

Q. All right. What's next?

A. So, the whole point of having a coding standard, whether you choose to adopt MISRA or write your own is to follow it. What good is a rule that is not followed? And so it's actually the enforcement part of having the rule that's important.

What I see is Toyota had a standard specifically for this system, they had various suppliers, including Denso contributing to this system, and themselves, but nobody was enforcing this standard at all. And that to me, based on my experience consulting in industry indicates a lack of rigor or engineering discipline within Toyota.

Q. What's next?

A. This is actually part of a larger pattern that I've seen through the documents that I reviewed, through the source code that I've reviewed, et cetera, which is that Toyota didn't do things that I would have expected them to do, and doesn't have documents and paper to prove that they did those things. I would expected them to produce, if their -- if my software was challenged, is there a bug in your code, I would expect to produce, here's the database of all the bugs that passed, found and fixed, who fixed it. That's how these bug databases work. How long it was known about before it was fixed, which ones we haven't found yet. You know, some of those might turn up later. They don't have that. There's testimony about that as well, that they don't have that.

They also do inadequate peer code reviews. So you heard Mr. Ishii say we look at some of the code some of the time when we're interested in it, but they don't look at all the code all the time. And peer code reviews is something that's a known, good, cheap way to find bugs. I wrote the code or change it, you look at it. You look over my shoulder. Just like an editor would do on a document. That's all code review is. It can be formal and it should be formal in a safety critical system, so there should be a paperwork trail that says on this date we met, reviewed this module, we found these three bugs or potential bugs, and we expect those to be fixed. And this paper trail will make sure that they get fixed. And that's how it should work.

Q. Based on a lack of systematic processes you described, have you reached an opinion on whether this software is defective?

A. Yes.

Q. What's your opinion?

A. In my opinion is that this code is a unreasonable quality and defective.

Q. You mention down here there is no bug tracking system?

A. That's what I talked about a database of all the bugs that have been found and fixed. It doesn't necessarily have to be a database, it could be a spreadsheet, but there should be some system in a company that's making safety critical vehicles that says, yeah, that odd behavior that was observed down in the lab yesterday, or on the track yesterday, let's assign some engineer to look into it, see what happened. Find the bug. Or if there's not a bug, explain it.

Q. Does Toyota agree there's bugs in the software?

A Yes. So I think this was in Mr. Ishii's testimony yesterday. When it comes to software there are going to be bugs.

Jumping to the end, so the issue is not whether or not there is a bug, but rather is the bug an important material bug. And indeed, there are not only bugs but there are also important material bugs in Toyota's code.

Q. Based on what you heard from Mr. Ishii has Toyota ever checked to see if a bug would stick the throttle open?

A. Mr. Ishii said he's never looked for one and he's not aware of one.

Q. Did NASA have concerns about software causing UA's in Toyota's throttle?

A. Yes.

Q. And did they look at it?

A. So this chart shows a bit of the methodology that was used by NASA. So, this is what's called a fishbone diagram. And so the idea is that, is there a way -- this is asking a question -- is there a way that unintended acceleration can be caused by a software error. And then they are enumerating through branching the possible ways that could happen.

And so, for example, there could be a bug in the throttle algorithm, and that would be an example of a coding defect or error in the recipe. That if happened and it related to US, could cause UA, and then NASA broke out other things, other things that could happen. For example, they talk about task interference or race conditions, and they talk about not having protections against faults like bit flips. And then the trace back, well, what would cause that bit flips, data corruption, communication faults, timing faults, et cetera.

And the idea is that if one of these root conditions can occur and is not blocked by something upstream, then it's a possible cause of UA.

Q. This document we're looking at, this diagram, is this one you created?

A. No, that's -- it's from NASA Appendix B pages 36 to 39, is that part of the analysis. I included multiple pages because there they describe their thinking and rationale on each of those sub bullets.

Q. So NASA was looking for the exact same thing you were looking for?

A. That's correct.

Q. Go ahead?

A. And these are examples of things we found. So putting it in NASA's terminology, and NASA's chart, the defects I've described fit into these coding defects, task interference, insufficient fault protection, data corruption paths.

Q. And in terms of the memory corruption we've been talking about, does it fall into these categories?
A. Yes. So specifically memory corruption over here, combined with insufficient protection against memory corruption, can lead to a UA.
Q. All right. There will be some discussion by Toyota in this case about layers of safety and safety items -- fail safes they put in their system to catch what we all term as UA, is that right?
A. That's correct.
Q. Have you examined some of those areas to explain where there may be a the gaps you talked about earlier?
A. Right. So the important thing from a safety point of view is not, we have 12 fail safes, or we have four fail safe layers, it's are there any gaps in them.
And so these are the layers as I see them and understand them from Toyota's documents and reports. And for each of them, each of these layers I wrote a specific chapter where I analyzed that part of the system, documented what I found, documented if there were any defects in the fail-safe or layer, and also if there were any holes that could allow something to get through these layers.
Q. So now we're going to look at each one of these layers and have you explain the defects?

A. Yep.

MR. BAKER: Your Honor, I don't know when you want to do an afternoon break?

THE COURT: Let's go till three.

Q. (BY. MR. BAKER) Let's go to the next slide?

A. So I've sort of put these in order. So layer one is first.

Q. Mirroring of critical variables. Tell me what mirroring means?

A. So mirroring is like having two cells that have the same value sort of in your spreadsheet. Technically, if you just have exactly the same value, I would refer that as echoing, with -- you have an echoed copy. Mirroring is slightly stronger than that, and Toyota generally uses mirroring, which is, mirroring is you also flip all the bits. So you have two copies of the thing, but if they were next to each other and they both clobbered to zero, they wouldn't match, because one of them being zero should make the other one be all ones. So it's an extra layer of protection.

And so the best protection for mirroring is keep them apart in memory, do something like flip all the bits in one versus the other. So that when you write to it, you write both. And when you read from it, you read both. And if they don't match when you read it, then you know that something has gone wrong and you can't trust that value.

Now, depending on how important that value is, it could be that you just use a default value and continue on. Or it could be a very important value like the throttle command, 10 degrees or 100 degrees -- or 100 percent, and in that case then you might do something different than just use a default value.

Q. So this is a technique that Toyota engineers have used?

A. Yes.

Q. Did they use it correctly?

A. Toyota used mirroring to protect thousands of variables. And they did it generally correctly. I'm not going to speak for all thousands of them. But they did it generally correctly with respect to those. The defect is, they missed some of the critical variables.

Q. Tell me about those variables?

A. So one example we've already talked about is the internal data structures within the operating system. They missed it because they never looked at the operating system. They got this operating system in binary from their chip supplier and they never looked inside it to see what was in there.

Now, if you're designing a FDA regulated medical

product, there are guidelines and you are instructed if you're building this insulin pump or pacemaker and you decide to use an operating system or other third party software, you need to audit that as well. Toyota didn't do that here. And that is one of the reasons I believe that they missed mirroring within the operating system.
Q. What about the target throttle angle global variables?
A. Yeah, there is a number of other variable that aren't mirrored, but the one that is really interesting from this point of view, from our discussion is that the target throttle angle, the one that says 10 degrees or 10 percent or 100 percent, 10 percent or 100 percent power, so not mirrored.
Q. So there's not -- there is nothing that's got that data stored like -- it wouldn't be mirrored?
A. There's no second copy of it. Not echoed, either.
Q. So if the first copy is corrupted, it's corrupted?
A It's the only copy.
Q. And why it that important?
A Well, it's important because if you -- if a software corrupts and changes that throttle command, the rest of the software just sees a number in a particular cell in a spreadsheet. It doesn't distinguish or know that it's not a command from the driver or a correct calculation of

what the driver wants and what the engine wants, not to stall, all those things. So if it suddenly changes from 10 percent to 20 percent, is that coming from the driver pressing on the pedal or is that coming from the software changing it?
Q. Have you got an example?
A. Yeah. Let me walk you through the process here. so the way their design works is that you have the accelerator pedal, which is being read by task X, and then it writes the calculated value, that very complex, code complexity of 146 unmaintainable function, it chooses a value. I put here as an example 20 percent of throttle. And then it writes it into a memory location. a 16 bit or two byte memory location.
Q. An unmirrored bit?
A. That's correct. It's an unmirrored 16 bit location.
Q. All right.
A. And then the next thing that happens is another part of the software comes along and reads it and it says, oh, it says 20 degrees, 20 percent. And so its job is to open the throttle to 20 percent. And that's actually kind of complicated because you're trying to move something mechanical and the software to trying to do it, so you're pushing on electrons, and the electrons are pushing on the motor and the motor is opening to the

right amount.
Q.   So how can you have a UA from memory --
A.   So, for example, if task X died and stopped writing to that location, and the unmirrored throttle command was set to a larger opening, the other part of the software is just going to pick up the new value and open the throttle.
Q.   Whether that is a correct value from the accelerator?
A.   Whether that's a correct value from the accelerator or not.
Q.   Go to the next line?
A.   So this says in words what I just said, which is that the death at task X causes the loss of throttle control, accelerator pedal doesn't work, cruise control doesn't work.
Q.   What else?
A.   This motor control task, and it's not just one task it's more complicated than that, I'm just simplifying it here for my explanation, but that motor control task keeps driving the motor -- and by motor here, I mean the motor that moves the throttle, it's the part that turns the knob on the water valve. And so, either if task X is dead, you can get a stuck throttle, which is the last calculated command, or the last computed one over here,

or it can change it to a corrupt value through an additional memory corruption.

Q. So if you have a memory corruption of the throttle angle variable that you just showed in your last slide and then have a task death, what can happen with the number that is sent to the computer to turn the throttle to?

A. Well, then it can become any number between zero percent and 100 percent.

Q. Is there any cap on the actual amount?

A. Well, the throttle physically, technically it opens between [redacted] degrees and [redacted] degrees. Whereas 90 degrees basically would represent no blockage of air flow. And so [redacted] degrees is slightly less than 100 percent. You can never really get 100 percent. And even when you close the throttle, you're usually not blocking all the air flow or else the engine would stall. So you're somewhere between about six degrees and maybe sometimes lower when you're idling, and about 95 percent of what you can get.

Q. Does the task death of X in that scenario involving the throttle angle variable have to occur first or after the memory corruption of the throttle angle variable?

A. If they are close in time, the two memory corruptions are close in time, it could be an either/or. If task X was dead for a while though and then the second memory corruption happened some time later, then it could also happen that way.

So if the two corruptions happen close in time, which is likely when you have memory corruption, often it's not just a single -- when it's a software bug or even hardware bit flip, it can be ricochet and bounce around like a bullet inside, and so it can cause multiple memory locations to be damaged. And so that can begin small and grow over time. And so, if they both happen right about the same time, it could be that the throttle command is corrupted first and then the task dies. But there's more time opportunity the other way.

Q. Can the throttle angle variable be corrupted through a hardware malfunction and a software malfunction?

A. It could be -- by itself, it could be corrupted by either one, that's correct.

Q. What's next on this slide please?

A. So this is just memory corruption can propagate from one to another. You can think of it as shotgun pellets bouncing around inside the memory, flipping some bits or changing whole bytes --

Q. And in this scenario, can the throttle angle go to any number?

A. Yes.

Q. All right. And have you done a diagram to kind of

explain this?
A. Right. So I put the previous graph together and I said, okay, on the left side we still have task X but it's no longer monitoring the accelerator or the driver controls, because it's dead and its death has not been detected. And then now I drew a vertical bar or a line showing that it's no longer ever writing to this global variable that's not mirrored. And so a memory corruption there changes it from, say, 20 percent of throttle to 50 percent of throttle.
Q. Are you just using that as an example for your chart here?
A. Purely illustrative.
Q. What happens next?
A. And then the motor control task not knowing that task X is dead, interprets this command as 50 percent as coming from the accelerator through task X, or from the cruise control through task X, or something else through task X.
And so now, it's just going to drive the throttle to 50 percent open, and you're going to get more engine power.
Q. In this example, do we have a task death?
A. Yes.
Q. Do we have a memory corruption?

A. Yes.
Q. We have the computer setting the throttle at some angle, 50 percent here in your example?
A Correct.
Q. Is that 50 percent in this example set by a malfunction in the software?
A. Yes.
Q. Is it unrelated to where the driver in your example is moving the pedal?
A. That's correct. So there's a disconnect now between that vertical line between the accelerator and what the throttle is doing over there in the engine.
Q. Well, we just talked about failsafes. What happened to the failsafes?
A Well, the failsafes are the monitoring -- that are left, are monitoring this portion over here and saying the throttle's open halfway in voltage, electrical terms, and the command is for it to be open halfway. Those failsafes don't know that task X is dead because they haven't detected it, and task X has taken some of the failsafes down with it that would have known about the driver's intent.
Q. Are some of those failsafes or the activation of those failsafes task X?
A. Yes, most of the failsafes on the main CPU are in

task X.
Q. So when it dies, what happens to the failsafes?
A. When it dies, they don't run and so the failsafes don't run.
Q. All right. And we talked earlier about a situation where something like this would happen and then somebody would step on the brake?
A. Correct.
Q. What would happen then?
A. So if somebody steps on the brake here in this scenario?
Q. Yes, sir.
A. If they weren't on the brake initially, and they step on the brake after this begins, then there is a failsafe in the monitor CPU that will inadvertently detect a symptom of the task X death. That failsafe is called the brake echo check. We'll talk more about it in a couple of slides. But the brake echo check will detect the driver pressing the pedal if they press the pedal and hold it at least about [redacted] of a second, and then it will cause the throttle to close, and [redacted] seconds later it will cause the engine to stall.
So if you have speed on the highway, the engine will stall.
Q. If a person has their foot on the brake when this

scenario in this example occurs, what would happen then?
A. In that event, in order for that brake echo that is inadvertently detecting this task X death to do anything, the driver would have to remove their foot entirely from the brake pedal. So while the car is speeding away from them, and as they are letting up mechanical pressure and maybe pumping or maybe -- I don't know, it's counterintuitive to let off the brake when that happens, but the car is going to speed up first because you are mechanically letting go of the brake pressure that you have, and then, because each time you pump you have something called vacuum loss, which causes the air that is flowing through the engine, because the valve so open for the throttle, that air is getting sucked into the combustion process and not going into the power brakes. So you actually lose brake effectiveness while this is happening if you start it on the brake. And it will go on until, can go on forever.
Q. If we have this example and starts with the driver has their foot on the brake and they never let off the brake, they are trying to get it to stop, how long would this last?
A Mr. Arora, Toyota's expert, says it depends on how much fuel you have.
Q. All right. Have we covered this slide?

A. Yes, I think so.
Q. Let's go to the second layer of safety that we talked about the DTCs and other failsafe modes?
A. So NASA in its report talked about the failsafes that Toyota described to it. And they were five failsafes on the main CPU that NASA discussed and these are called the limp home modes, the idle mode fuel cut and engine off. And just briefly, the limp home modes, some of you may experienced this in a car before, that if your car's engine is malfunctioning, it will allow you enough power to drive, to limp to the dealer or repair facility, but not enough power to go out on the highway. And that is a safety mode.
And Toyota has three different ones. And it depends -- there's two gas pedal sensors, accelerator pedal sensors, if it mistrusts one of them, it might allow the throttle to be open 10 degrees or 1- percent, if mistrusts both of them, then it will only allow the throttle to be open a smaller angle. So there's three different angles. As I recall, they range from [redacted] degrees or [redacted] degrees to [redacted], or [redacted] degrees.
There's also something called idle mode fuel cut, which is that when your car is idling the rpm will never go above 2599. Just like when you're driving on a road, no matter how much gas you give it, the rpm will never go

above 6400. When you're just sitting there at a stop sign it will never go above 2500. Now, 2500 rpm, especially depending on the gear you're in can be a lot of power in a car, but that is a limit that is built into the software that NASA describes as a failsafe mode.
Q. Where are these failsafes located?
A All of them either are located entirely within or depend upon task X. So when task X is not running none of these are relevant to the discussion of UA.
Q. And part of your heading has got DTC, the diagnostic trouble code. What is significant about them in terms of task X?
A. So the DTCs, as I've mentioned, is something that is stored in the computer that says something went wrong. And so when this happens, there is not going to be any DTCs stored, but I don't want to rule out all of them because there is another task that does a few. But generally speaking most of the DTCs are going to be disabled during this scenario.
So if you were to reboot the car and read the computer you may find no codes as though nothing was wrong, and now because you've rebooted it, all the [redacted] tasks are alive and the car is running normally again.
Q. The diagnostic trouble codes that can be set when something is wrong with the car, if they are set and

stored, are they stored forever?
A. No, they are not.
Q. If the vehicle loses battery power, what would happen to the codes that had been set?
A. The DTC codes are stored in an area of memory called battery backed ram. Most of the time when you reboot a computer, the ram, working memory, is emptied out or become nonsense. But battery backed ram, because it's getting a trickle of current all the time can maintain its contents. But it only maintain them while the battery is applied. So if you parked the car after the incident and the battery drained, then you would lose all the information. Or if during the accident there was a disruption of power supply, then you would lose those codes that might have been set.
Q. So for example --
A. And that's true regardless of task X death or anything else. That's just how the system works.
Q. That is how Toyota's system works?
A. That's right.
Q. So if Ms. Bookout's car before it was inspected by anybody lost battery power, would any DTCs if they were set, still be in the car?
A. If the battery had been disconnected there would not be DTCs to recover because they would have disappeared

from memory.

Q. All right. Let's go to the next slide. The third layer your title watchdog supervisor. Can you explain this one to us?

A. This one is going to take some explanation. So if you ever had a computer crash like your iPhone or your Android or whatever, and you were there to reboot it. It's not working and you reboot it. But some computers are in situations where there is nobody there to poke the button. So for example, when NASA sends a mission to mars, Mars Pathfinder is a good example in 1997. The first color images come back from the surface of Mars. The sent it there, they include in the design something called a watchdog. So the idea is that the hardware will wake up or reset the system if there is a software crash. And this actually turned out to save the day in the Mars Pathfinder mission because when that ship arrived on the surface it was able to beam back photographs and other things, and the following weeks when NASA engineers were doing their science on a surface, they had actually a number of watchdog resets. If the watchdog had not been there to save the day, then they wouldn't have gotten the computer to phone home again so they could fix it.

In your car, the watchdog is there to -- if something goes wrong with the software, it should be there to reboot the system very quickly so that you can get back to a safe running car. And Toyota does have something, they have a watchdog timer chip and they have something they call the supervisor. I call it the watchdog supervisor in my report. And that the job of that software, that part of the software is to periodically check in with this watchdog timer hardware, WDT, and if the software doesn't check in, then the hardware resets automatically the processor.

Q. Is that what it's supposed to do?

A. That's what it's supposed to do.

Q. In the example you just gave, if we have a task that dies, say task X, and it doesn't report in to the watchdog, what's supposed to happen?

A. Well, ordinarily when you have one of these watchdog supervisors, the software to kick the dog, kick the timer, you're supposed to monitor all the software for its health. And that's been well-known for a long time. And certainly, when I was editor in chief of the magazine, that was well0-known and we published articles about how to do good watchdog timer design. That would have been in the 2001 to 2003 time frame.

When there are multiple task because you have an operating system, it's necessary to check that they are all working. You can't just say, well, I, the supervisor in here, I"m happy, don't reset us. You have to check on all of them. That is how it should work. Unfortunately, that's not how toyota's design works.

Q. What is the problem with theirs?

A. The Toyota's design actually they have an abysmal design, not just unreasonable in my view, but I use the word abysmal. This was actually the first chapter of my report I wrote because I couldn't believe what I was seeing.

Toyota has a watchdog supervisor design that is incapable of ever detecting the death of a major task. That's its whole job. It doesn't do it. It's not designed to do it.

It also, the thing it does in Toyota's design is lookout for CPU overload, and it doesn't even do that right. CPU overload is when there's too much work in a burst, a period of time to do all the tasks. If that happens for too long, the car can become dangerous because tasks not getting to use the CPU is like temporarily tasks dying.

And in Toyota's watchdog you can have any overload going up to one and a half seconds, which at 60 miles an hour I calculated is about the length of a foot ball field, you have any vehicle malfunction for up to a foot ball field in length that's explained only because this watchdog design it bad, and because the processor is overloaded momentarily. And that should have been also a job of that watchdog supervisor. And that is one they tried to implement and they don't do it well.

They also made a classic blunder, one that's taught by professor like at Dr. Koopman to first year students in his imbedded systems class, which is, you don't dedicate a hardware timer on the main CPU to periodically kick the hardware on the watchdog, because that will keep functioning even though vast portions of the software and the tasks are not rubbing because these interrupts are a higher priority than the tasks.

And so, that is a design that you -- and I have spoken about that at many conferences, not doing it that way. And they do that.

They also, in order to not detect a death of tasks, the operating system is sometimes telling them, hey, the task isn't working right. And they have lines of code in there to throw that information away. They are ignoring error codes from the operating system telling them there's a problem with this task. And that, by ignoring those errors codes, is a violation of another MISRA rule, No. 86 in the 1998 version.

Q. So if a task death occurs and that information is ignored, it would violate this MISRA rule?

A. That's correct.
Q. And could that have an impact on causing a UA?
A. Yes.
Q. Are there ways to do it differently?
A. There are. Reasonable alternatives to this were well known long before this car was designed. In fact, in the 2005 model year Prius, they have -- in a Prius you have two engines. You have a combustion engine and you have a battery engine. The Prius combustion engine looks a lot like the Camry combustion engine code, but they had a fresh new design for the hybrid battery computer. And guess what? It has a good watchdog. It's a better design in there. It monitors the health of every task, and it monitors both for executing it too frequently, and for not executing frequently enough.

The primary purpose of this part of the software should have been to detect task death. Toyota didn't do that. In my view, based on all the evidence I've seen, because the CPU was overloaded at times, and the watchdog was weakened to allow that.
Q. So based on your information from the Prius, did Toyota know how to do it right?
A. Absolutely.
Q. Let's go to the next slide. Layer four, this is our last layer of safety that you're going to talk about from

Toyota's perspective.
A. Let me just back up. You asked me did Toyota know about it. And i don't know for a fact whether the engineers would have at Denso or Toyota.
Q. Fair enough. Thank you. Is this our last layer of safety that was in your original slide?
A. Yes, this is the fourth layer.
Q. The ESPB-2 monitor CPU. I think they've heard a lot about this, but that's the smaller chip that you showed them in the picture of the overall board, correct?
A. That's correct.
Q. Tell us about this.
A. So there are some failsafes in the monitor CPU for various purposes, and I examined those. And on this slide I'm summarizing the relevant ones with respect to what happens when there's task death and UA.
One set of them is what's called system guards. And there is these three different system guards, one on the main processor, one on the monitor processor, and one that straddles the two of them.
And in theory they are specifically designed to look out for UA. But in practice, when task X is dead, they are either dead or they don't have any knowledge of the driver's intent. And so they are not operating at that time.

Q. And the brake echo check, you mentioned this a couple of time earlier, correct?
A. Yes, so the brake echo check has turned out to be an interesting aspect of the monitor CPU, because it does sometimes detect the death of task X after there has been a UA in our testing. So in the testing where unintended acceleration by task death was observed, sometimes when the brake switch was transitioned, either the driver first pressed on the brake or the driver released the brake because they had been on it, this brake echo check detects that symptom of task X death, however this is not an appropriately designed failsafe because, first of all, it waits for the driver to have act first.

So, and also if the driver's action when the car is misbehaving, is to say it's going slower than I want, let me step on the gas pedal, this does nothing. So the driver has to act first and the driver has to change the state of the brake pedal, which in some cases could mean doing something very counterintuitive, which is taking the foot off the brake during an emergency event.

Clearly, that is not by design of Toyota's engineers, despite what we heard from Toyota's expert Mr. Arora.

In addition, it takes the wrong action. When this brake echo check that inadvertently detects task death does act after the driver, after the UA, it doesn't reset the ECM to restore the system to normal function. It stalls the car wherever you are. It first cuts the throttle, which slows the car, and then [redacted] later it stalls the car completely, which could also contribute to harm.

Q. You understand there's been no evidence in this that Ms. Bookout's vehicle stalled prior to the crash?

A. I do.

Q. And we've got one more line?

A. Just simply from my analysis of the source code, there are several reasons. I put them in my report my this brake echo check is also nonreliable.

Q. And why is that?

A. There is some reasons why it's not -- it's not designed to be 100 percent reliable. There are several reasons, I'd have to look at my report to refresh my memory.

Q. Do we have another line up here?

A. And finally, nothing in the monitor CPU detects all main CPU malfunctions. There is not, for example, a watchdog supervisor like function that looks out for task death, or looks out for UA. These are the relevant ones.

Q. How do you know that?

A. Because I've viewed the source code, because in the

testing of nothing else is active.
Q. Okay. This particular part, this monitor CPU, have you seen any evidence that Toyota actually did a design check or design review on the software or source code in the monitor CPU?
A. I have not.
Q. Do you have a slide on that?
A. So Toyota didn't look at this monitor CPU. The end final failsafe, the second CPU, they didn't look at it. As -- this was, I think from Mr. Ishii's deposition on Friday, when it comes to the source code for the monitor CPU, we, Toyota don't receive them, there would not be a design review done on that software. And the attorney asked, that's the one with the monitoring software for the electronic throttle control system, correct? And Mr. Ishii said yes.
Q. And you were here to hear that testimony when it was played?
A. I was. And I've read it before.
Q. Was the next slide please?
A I just want to repeat that, because I think that is an important point.
Q. Why do you think it's important?
A. Well, Toyota has made public statements that couldn't possibly be a software cause for UA. I've

reviewed documents where toyota's own investigative teams to end UA complaints don't include anyone for software on the team. They look floor mats, they look at pedals, they look at confused drivers, but they've never really sought the source code to actually look and see like, hey, this second chip, does it really do what we think it does.
Q.   And is it this chip, the monitor chip, you've seen the source code?
A.   I have. And NASA actually has not. NASA was not provided with it. I think we heard Mr. Ishii say maybe they didn't ask for it.
Q.   And the source code for this chip that was produced late?
A.   Yes, this is the source code that was produced about three weeks before my report was due in Van Alfen. And this, by the way, a exactly the same chip and software from 2005 to 2009 in the Camry, and some other models as well, but that is irrelevant to this discussion.
Q.   Is it the same in the --
A.   I don't recall as I sot here.
Q.   Why do you say the monitor CPU is the last a line?
A.   Because there's nothing else beyond the monitor CPU. So if the main CPU is malfunctioning and its own failsafes are either disabled or not doing anything, the

monitor CPU knows that the driver is pressing on the brake, the monitor CPU knows the percentage open of the throttle, the monitor CPU knows how long those things those have been happening at the same time. So, for example, if the driver has been braking for half a second and the throttle is still at 50 percent, surely that suggests there is some sort of problem going on in the vehicle. Potentially, the main CPU is malfunctioned.

And this chip had in it everything it needed when it was designed about 2002 to have paid attention to those two things. It had all the electrical signals coming in, all electrical signals going out, it had adequate memory, it had adequate CPU time to do this. Small check. And it could have -- if it was a software malfunction, a reset of the ECM would cure it. Now, if it was something like an entrapped pedal, resetting again is obviously not going to fix that, and so maybe a second action should be something different.

But as a first step, as a first action, they could have included software like this. And this is extremely important. Toyota designed a vehicle that has a braking system where the power brakes are connected mechanically through air flow to the throttle. When the throttle's wide open, the air is largely flowing into the combustion process, because's is a vacuum there sucking it in. And it's -- and the brake can become depleted so you don't have assistance from the brake. You're losing pressure when you pump.

And Toyota must have understood that. There is a mechanical linkage between the throttle and the brake. And maybe in a mechanical throttle system it was always the case that the driver let off and closed the throttle, so that wasn't a problem. But when they put software in charge, they should have taken notice of this and cared tremendously of the fact that the software was responsible for all three elements of combustion. And they could have acted back in that time in 2002 approximately when they were designing ESP-B2 chip, they could have acted to stop any UA, no matter how many bugs were in the CPU.

Q. If they already had that chip would it have cost them anything to make that software change?

A. I mean, it would have cost some engineering time to do this and testing time. But in terms of a per unit cost per car, it's the same chip, same amount of memory, same processor, a couple hundred line of assembly code.

Q. All right. We've gone through several things. Let's talk briefly about the software process of Toyota. Have you evaluated that?

A. I have, yes, sir.

Q. What did you determine based on that?
A. There is a number of defects, and some apparent explanations for those defects. So one defect is that there are single points of failure and the -- what they call the failure modes and effects analysis that Dr. Koopman talked about and I think he showed on one of his slides some examples of Toyota's documents, where they think of things that might go wrong and then they decide if and how they are going to mitigate them.
They missed stuff when they did that. And that it's my opinion that's because they didn't a formal safety process like the MIRSA, the big book. They don't follow a recipe for making a safe system.
They also have the defect that they didn't do peer reviews on the operating system code or the monitor CPU codes. And here, ultimately, it comes down to resources. Toyota did not put people and time behind checking up on the suppliers who were supplying this critical software. The operating system at the heart of this main CPU and this and second CPU that's doing the monitoring.
Q. What about the watchdog?
A. Well, the watchdog, I haven't seen any evidence that they peer reviewed it. But that design has stayed almost identical through the model years that I've seen on the main combustion engine.

Q.   Did the watchdog supervise the task death?
A.   Not reliably, not most tasks.
Q.   What else?
A.   The -- another defect in their process is that they didn't follow their own coding standard. Now, in my coding standard chapter, I assess my opinion of their coding standard. I've studied coding standards, I've written a coding standard book, I'm familiar MISRA, and I assessed that many of the rules that they have are simply, like this how should name your variables, they did not have very many rules that would have kept bugs out. And in fact, some of their rules actually would have increased, related to race conditions, would have increased the likelihood of bugs in their code, particularly over time.
     And they didn't even follow this lousy coding standard that they had. They didn't put people, again, to make sure that their suppliers -- and not all this code was written at Denso. The code on the main CPU was partly coded from Toyota, partly coded from Denso. And when they is a different supplier like Delphi that GM supplier, they give the Toyota part of the code to Delphi and then Del Phi adds the Del Phi part of the code, so it's a mix of Toyota code and supplier code. And they didn't enforce the coding rules, apparently on either

one.

Q. What's next?

A. Nedt is that the watchdog supervisor doesn't detect most task deaths. As I explained, it's my view that the reason for this is that the CPU was overloaded from time to time. In other words, it cost them less to water down the watchdog than to upgrade the CPU to a fast enough CPU.

Dr. Koopman talked about something, called rate monotonic analysis. It's in my report too. That's something that Toyota's engineers should have done to make sure that all of those tasks would always complete on time and there would never be CPU overload. But they didn't. And there are specific places in the code where they say, oh, that test didn't finish yet? Okay. We'll wait for it next time, maybe it will run next time, because the CPU is overloaded.

And there are also indications that different model years of different cars they are moving around functionality, like the automatic transmission, is in another processor on the same board, or on another board. And because early on they are trying to do all this stuff with older processor technology, and then in the 2005 Camry design they combined them together into one.

And they keep switching these things around, which

is an indication to me that they can't do it all in that one processor. And that, the poor watchdog design, a number of other things that I've documents in my report.
Q. And then lastly talk EDAC.
A. Right. So those extra hardware protections bits, the EDAC that NASA calls it, the parody that Dr. Koopman talked about, those cost money. And it's actually somewhat straightforward to calculated, because if you have eight bits you want to protect, to do it right you need five more bits. And so you're taking something that was eight and making it 13. And a lot of the cost in that is related to the size of the chip, and that's tied directly to the number of bits. So you're increasing the area of the chip making a bigger processor in order to do that. And Toyota chose not to do that in the 2005 Camry. They had by the 2008 Camry added not the five bit version but a cheaper version, I believe it was a three bit version.
Q. In terms of EDAC, is Toyota tell NASA that the; 05 had it?
A. Not only did NASA write in its report that they had it, but I've seen the email where NASA asked if they had it and Toyota responded that they did.
MR. BIBB: Objection, Your Honor, hearsay.
THE COURT: Overruled.

THE WITNESS: That's in my report, by the way. What was I talking about?

Q. (BY. MR. BAKER) You've seen an email where Toyota actually told NASA they had EDAC on the '05?

A. Right. So it's clear that NASA didn't just make this up out of thin air, Toyota told it to them in an email.

Q. Let me ask you this about EDAC. Does EDAC help prevent memory corruption?

A. Yes, it does. And NASA was concerned about if there what bit flip due do EMI or some other hardware effect, could that cause a UA. And NASA relies on the fact that there's no EDAC when reaching its decision that that can't happen.

Q. Because they believe --

A. Because they believe EDAC is in it. And furthermore, Toyota redacted or suggested redactions that were made in the NASA report almost everywhere the word EDAC appears it's redacted. So someone at Toyota knew that NASA thought that enough to redact from the public that false information.

MR. BIBB: Objection, Your Honor, now he's interpreting, I move to strike that last piece of testimony.

THE COURT: I'm not going to strike it but move on.

Q. (BY. MR. BAKER) Let's go to the next line?

A. Just the point really is, if they were confident that they didn't need EDAC, why left NASA believe it if they had some other explanation.

MR. BIBB: Objection.

THE COURT: Sustained. I'll strike that last answer.

THE WITNESS: I'm sorry. I misunderstood.

THE COURT: Is this the last slide on software?

MR. BAKER: We can break if you need.

THE COURT: Why don't we do that. It is now 3:00, we're going to take a 15 minute afternoon break. I remind you during the break, do not discuss the case, do not form any opinions and get lots of caffeine.

(THE FOLLOWING PROCEEDINGS WERE HAD AT THE BENCH OUTSIDE OF THE HEARING OF THE JURY.)

THE COURT: Were back on the record outside of the presence of the jury.

Go ahead, Mr. Bibb.

MR. BIBB: As I understand, the plaintiff intends now to offer most of the other incidents that are identified in Mr. Barr's report, am I correct Mr. Baker?

MR. BAKER: Yes, sir.

MR. BIBB: Our objections to that would be to the extent there are incidents that occurred after the date of the incident in this case, which is September -- back September 2007 or after the date of this vehicle was sold after August of 2005, that those can only be offered for purposes of trying to show defect in this vehicle. And plaintiff has a high burden of showing substantial similarity with those and it is the plaintiff's burden, so I think we're going to have to do more than just ask Mr. Barr to describe them to jury. We're going to have to have some sort of hearing on each one of them as to whether they are, in fact, substantially similar. And I understand the Court is interested in the type of software, but again you've got to look at the type of incident. There are short duration incidents, long duration incidents and I think that you're going to have make more of a showing than plaintiff intends to talk about.

MR. CLARK: A particular problem is the problem that we got into on Friday with regard to those vehicles that have six cylinder engines, because I think the Court's already seen the PowerPoint is full of limitations, you know, limitations to the L4. There has been some sort of discussion of some differences between the four cylinder and six cylinder. For instance, EDAC is present in the later six cylinder engines, something we just ended with. And it's certainly our position that Mr. Barr saying that the four cylinder and six cylinder are substantially similar to my purposes, which I think is the gist of his testimony, is not a sufficient foundation. The evidence is undisputed that there are significant hardware and software differences between the two engines. In fact, the older six cylinder Camrys have an extra CPU in them.

THE COURT: Mr. Baker.

MR. BAKER: Your Honor gave us guidelines that you anticipated you would follow in looking at these defects, and also refute the position taken by the defendants as to reasons they can be both. The Court at that time said whether it's pulling into a parking lot or merging onto traffic is not necessarily a big factor that you were going to consider, that you were more concerned about is that software defect issue that was looked at by Mr. Barr substantially similar. I've already set a lot of the predicate already. I specifically had him describe 2002 to 2010 Camry's, the L4 and E6s where he said the software was substantially similar, that they also had the same operating system, which I'll reiterate. The ones in his report are all Camrys and so I would only ask him about one's he specifically reviewed and relied on in part of his analysis in this case.

MR. CLARK: Your Honor, on this slide we've got some bullet points --

THE COURT: Which page?

MR. CLARK: 55. I'm sorry. Mr. Bibb was talking about having to have a mini hearing on these and that's exactly right. There's at least one of these vehicles in his report that does in fact have an all weather floor mat present in the vehicle and it's in his report anyway. Obviously we are going to have examine him about that. And you know, this is sort of getting into the 403 issues and a waste of jury time and the cumulativeness and the confusion of the issues that I think we've already briefed and already argued, we would reiterate here, because whether or not a particular incident that postdates Mrs. Bookout's crash was caused by a floor mat is wholly irrelevant to what this jury has to decide.

THE COURT: Which one of these did you say -- did you find specifically there was a floor mat issue?

MR. CLARK: Ms. Preese-Morrison testified that she had a plastic floor mat that she bought at Walmart that was on top of her --

THE COURT: I just read her deposition during the lunch hour and she was very clear -- at least her testimony, she was very clear that she had the officer look at it, so.

MR. CLARK: That's right, but that is not what Mr. Barr's sides says. Mr. Barr's slide says no all weather floor mat.

THE COURT: And you certainly attack him or critique him on that on cross examination. Is there another that you think -- because I see a lot of these he says no floor mat.

MR. CLARK: We can go through one by one.

THE COURT: I don't care to do that. Was that the one you were specifically referencing?

MR. CLARK: That was the one I was thinking of. I think Gomez was in his Van Alfen report and he took that out. That's another one.

THE COURT: Let me tell you. I had made notes on what he was saying about these and he said for the 2002-10 Camry models that the operating systems are substantially similar as were the software systems substantially similar. And that he talked about a whole chapter one that discusses the similarity of it. I had another notation on his slide 43 where he specifically says that this particular software is the same in everything from 2005 to 8. And there are only two of these or three of these perhaps that I tabbed that were actually nine, but I think were included in the first --

in his statement having to do with this chart on page five. But Mr. Baker, has address, and again, I don't know other than hearing it in argument that I've heard anybody say that the V4 or V6 that the engine size changes anything.

MR. BAKER: I asked him specifically that question and he said the software was substantially the same.

THE COURT: Regardless of the engine size?

MR. BAKER: In terms of this defect.

THE COURT: Right. And then I did notice in terms of going through, and again, I haven't read each one of these, but I did notice that there is additional stuff in here about people die and say they are going to die or they're severely injured, or going off a sheer cliff.

MR. BAKER: I'm not going to -- I'm just going to ask factually about what happened in the UA event, not who died or who got hurt. I will instruct the witness not talk about that.

MR. BIBB: As I understand, Your Honor, the facts and circumstances of these accidents vastly different between the circumstances in this case, you can still admit them. The first one, Hill, was attempting to enter a parking space where the vehicle suddenly accelerated. A very low speed, very short duration, very confined area. The factual differences between many of these incidents and the crash in this case which was a high-speed exiting of a freeway.

THE COURT: But I don't think there has been any evidence, correct me if I'm wrong, that has said that if -- because Toyota's position has always position has been, this just didn't happen. But from the plaintiff has there been any evidence that task death would perhaps only occur when it's a long term as opposed to a short term? I mean, it happens and then it lasts whatever length of time it might last until there is an accident or it stops?

MR. BAKER: That's right. And he specifically used these events as part of his root cause analysis to come to the conclusion.

MR. CLARK: Something that is important, Your Honor, is that Mr. Barr's testified, that if the incident starts with a foot off the pedal, or a foot on the accelerator pedal, and then the driver brakes, then the brake echo function is going to close the throttle and eventually stall the vehicle after [redacted]. That was his testimony. That was the only testimony that we've heard. So if you take one like -- let's see. Hazel is a good example, 77 and 85, apparently didn't begin with a foot on the brake and once the event began she applies the brakes. That takes us out of the similarity of these incidents that allegedly begin with the foot on the brake where he's testified that it's absurd to expect somebody to remove their from the brake.

THE COURT: Mr. Baker, are there certain ones of these that it's undisputed that the foot was on the brake all along so that this brake echo should have kicked in?

MR. BAKER: I don't know that the answer to that, Your Honor. I viewed these as part of this analysis. I think whether the foot was on the brake when this started, then goes to the weight of it, not to its admissibility. And part of this is to refute Toyota's position that this doesn't ever happen.

MR. CLARK: Doesn't go to weight versus admissibility, Your Honor, it goes to whether it's similar or not. Nassar is another good example of that. This fellow was entering the highway. I've entered a lot of highways, I'm sure the Court has too, and I always enter highways with my foot on the gas pedal, so that one pretty clearly there's is transition that takes it out of similarity. 81 and 85, I'm sorry, the top of that page.

THE COURT: Where is it in 81 that you said he had his foot on the brake?

MR. CLARK: He said the incident started on the second or third lines, while driving in New Jersey. He reported that while he was entering the highway the vehicle wanted to continue to accelerate. I'll admit from that we don't know for sure what pedal his foot was on, but it seems to me you're entering the highway pretty likely the foot is on the accelerator pedal. He goes then from the accelerator to the brake. And Mr. Barr has said brake echo should work in that situation, it should close the throttle. I think that is undisputed.

THE COURT: Let me ask, isn't this all being offered just for the purposes of refuting Toyota's claim that these situations don't exist. And you're not claiming that the brake echo wouldn't -- was there a brake echo in this car?

MR. CLARK: Yes.

THE COURT: So you're not saying the brake echo system, you're just offering these for the purpose of showing that unintended accelerations, some of them brake echo may have kicked in because the way of foot was applied.

I'm going to allow these with the caveat being none of the details about describing the accident or people who were injured, statements in it.

MR. CLARK: Are we to understand then that the universe of other incidents in this case is limited to the ones that Mr. Barr has described?

THE COURT: No. And we will discuss that in more detail. One of the depositions that you all gave me had somebody reading through a bunch of reports and we'll be discussing those outside the presence of the jury as to which if any of those are going to come in. But right now I would say you're probably well taken because if he hasn't laid a foundation and it wasn't a preaccident, I don't know how else they are going to get their foundation laid. Okay.

(THE FOLLOWING PROCEEDINGS WERE HAD WITHIN THE HEARING OF THE JURY AS FOLLOWS:)

THE COURT: We're on the record. The members of the jury are present as well as counsel and their clients. Mr. Barr is still on the stand and still under oath and you may continue -- how about this, you may conclude your direct examination.

Q. (BY. MR. BAKER) Looking at this slide how Toyota's inadequate software process, I think we heard a little bit of this from Dr. Koopman. Can you briefly tell us why you put it in your slide presentation?

A. Yes. What I conclude from reviewing the documents and examining Toyota's source code and other things, is that while Toyota has a reputation for being a quality producer of mechanical automobiles, that internally their software process was inadequate, and you know, they lacked internal expertise in a number of areas. This is their own internal document where this is a software development process that they've laid out. And each of boxes that's in pink with an X, Toyota is saying we don't have knowledge inside Toyota, we're entirely relying on our suppliers for these areas.

And then in the same document there is a process in place for hardware and not software. In my consulting practice, in imbedded systems of various kinds, I've seen over the years that there is not really very many companies that just specialize in Imbedded software. But most companies that make an imbedded product they make the product first and then they end up with software inside.

So they make cars first and then they end up with software inside them. They make microwave ovens first and end up with software inside them, et cetera.

And so what I see Toyota came late to the software process, maybe, I don't know about current cars, but maybe they've improved things. This was part of a document where they were trying to improve things starting about 2007, with the 2012 model year.

But at this time when these vehicles were being made, including the 2005 Camry, they did not have an adequate oversight or training of their suppliers or engineers, they didn't have an enough staff in this area, et cetera.

Q. Have you reached a conclusion whether you what determine to be an inadequate software process le to the defective software you're going to describe?

A. Yes.

Q. What's your opinion?

A. It's my opinion that that lack of process led to the defects and the detects led to the UA that's described.

Q. Let's go to the next slide. This again relates to the process and the culture within Toyota?

A. That's correct.

Q. And what is this document?

A. This is a document that's an internal Toyota document. You can see Mr.Kawana was one of the recipients. But it's dated around the same time as those business review documents about their software process and their spaghetti code. It's in September of 2007. And I pull out this quote here from this email where the author is saying "In truth technology such as failsafe is not part of the Toyota's engineering division's DNA." And it continues, "But isn't it good that it is recognized as one of the major strengths of Toyota and its system controls industry."

And then I highlighted also the portion that says, "Continuing on as is would not be a good thing."

Q. What does this tell you about your review of the documents?

A. My interpretation is that inside Toyota there was a growing recognition that they were not designing safe cars.

Q. In terms of the software?

A. In terms of the software, that's right.

Q. The next one, we've talked a little bit about NASA, you included in your chart, is that the NASA report?

A. Yes. On page 78 of the NASA report, NASA report had some chapters also, they called them appendices, but in the main report at page 78 they had a table where they laid out some possible ways that UA could happen in Toyota vehicles and there were two they couldn't rule out. I talked about them earlier. One was if both accelerator pedal sensors failed at the same time or failed together, then the software had no way of knowing that.

The other was exactly what I've described here, a systematic malfunction of the main CPU software that is not defected or not detected in time by the monitor CPU. And so the quotes on the bullets match up with the highlighted portions of NASA's assessment of what that would be like. They are saying that the fault would escape detection, so a single memory corruption would result in UA. Default would escape detection because there wouldn't be an EDAC error. And it turns out there is no EDAC to cause an error.

The idle fuel cut would not be active. The reason for that is because it's one of the five failsafes that are in the task X.

The watchdog would continue to be serviced.

Q. What does that mean?

A. Serviced means -- a lot of words are used for these watchdogs. You can kick the watchdog, pet the watchdog, stroke the watchdog. NASA used the word service. Service the watchdog, means checking in from time to time to say everything is okay. So NASA is saying during this defect the watchdog timer would still have to be getting kicked or checked in with. And indeed, Toyota's defective watchdog software will continue to check in and doesn't detect the task death.

Q. And the monitor CPU?

A. And the monitor CPU doesn't detect the failure and here because it's not designed to. Even the brake echo check that sometimes has detected and caused a sharp throttle and engine stall after the driver has acted

after the UA has occurred, it wasn't designed to do that. It's inadvertently doing that. And the way you can tell it's inadvertent is because no designer would design a safety system where the driver of a car that is accelerating away from them had to release the brake. Even in some cases. And I haven't a Toyota Camry users manual that says, if your car is accelerating and you don't want it to, try braking. If that doesn't work, try not braking.
Q. Are we done with this one?
A I think so.
Q. I think this is a point raised by Dr. Koopman about single point failure, is that significant to you?
A. Well, it's significant because it's a very point in safety critical system design. We don't want any single points of failure. And Dr. Koopman used a nice example of an airplane with one engine, or an airplane with two engines that had a common failure mode such as one fuel pump. And so this car shouldn't have single points of failure in it. And that is a normal mode of design for automotive safety systems.
Toyota tried to mitigate the risks of things like this happening, including in software, but they missed some of the single points of failure. And that is what happens when you focus on the trees and not on the forest

of having an actual safety process adopting a big MISRA like safety software building process and hardware design process.

And so some of the faults, some of the single points of failure are getting through gaps in the failsafes. Like Dr. Koopman said, there may be misbehaviors of Toyota vehicles that are getting caught by failsafes. What's really at issue here is that sometimes not only are there misbehaviors but they are slipping through the failsafes, and those are the ones that get complained about and those are the ones that injure people.

Q. Go to the next slide.

A So as I stated, there are single points of failure in the ETCS. Some of these have been demonstrated but not all of the ones that we've identified have been demonstrated in the vehicles.

And task death, although I focused a lot of task X here, because it does so much and it does throttle control and it does failsafe, it's pretty important, but there is [redacted] tasks and they can die in different combinations. It could be task 3 and task X, or task 3 and task 7 and task X, or just task 9. And those can cause an unpredictable range of vehicle misbehaviors. It turns out that unintended acceleration is just the most dangerous thing your car can do when it malfunctions.

The most thing dangerous thing your iPhone can do is crash or not let you call 911. The most dangerous thing your car can do is shoot down the road. So other lesser software malfunctions also likely occur, but those are the ones that get reported is these dangerous, the ones that cause harm.

MR. BAKER: Can we approach, Your Honor?

THE COURT: Yes.

(A DISCUSSION WAS HAD OFF THE RECORD.)

THE COURT: Ladies and gentlemen, this next document is going to have source code information on it again, so if you not been authorized to view the source code, please leave the courtroom.

Q. (BY. MR. BAKER) Go to the next line. You just finished testifying about other tasks not being identified when they die?

A. Correct.

Q. Is this chart associated with your work that has shown that?

A. That's correct. So the vast majority of the testing that has been conducted by either side's experts to date has involved killing just one task at a time. So each of the [redacted] have been tried. And so I've put together this table with [redacted] tasks. It's not their names and the source code that are here, but it's just a brief description of

the task to help me remember how to talk about them.
And then what things have happened in the tests that have been conducted by Mr. Louden, and also by Toyota's expert Mr. Arora. And so this is a summary chart and it talks about those things.
Q. And so of these [redacted] in this chart where we see some reaction by the software and then not detected by the software, is that an instance where just a single task was killed?
A. Right. So there has also been some testing where task X was killed and one other task was killed, not referring to that here. Just referring to task where one task was killed. It's as though one of the [redacted] programmers on the Toyota team never showed up for work in your car at that point. So what happened in the car. We already heard a lot about task X death by itself, and that's if the driver changes the state of the brake pedal, then the throttle will get cut and [redacted] later the car will stall. And I put in parenthesis that that's the echo check. That is the brake echo check that's detecting that.
Q. And that we discussed application of the brake in the sequence of an UA, correct?
A. Right.
Q. If we've got a person who has their foot on the

brake, but they -- I'm going to describe it as a pumping action, but they come back and forth pressing on the brake up and down, will that reset this echo and make it work every time that occurs, or is there something special that has to happen?

A. No, pumping can be without a full release of the pedal. You just move your ankle, you go up and down, but never really let off the pedal. If you don't let off pedal then it will go on forever.

Q. Is there a special, what I'll call a brake switch for lack of a term, within the mechanical brake system that has to do some special function in order for it to reset for this echo to work?

A. That's correct. First of all, the switch has to open, and then also it has to be held open at least [redacted] of a second before this brake echo will do anything.

Q. So we can have brake application and be within the constraints you just defined and not turn over the brake switch, and it won't cause this brake echo to come on?

A. That's true. And that's a good point because my slide just says brake change, but it has to be a brake change of a sustained duration. It can't just be a pump that doesn't let fully off. I was trying to summarize things here mostly so I could explain them.

Q. In terms of the other ones here that you show us, should there have been something within the software that detected the death of one of the [redacted], any of them, that were supposed to be running?

A. Absolutely. There should have been something that detected the death of any one of them as quickly as possible and reset the ECM in order.

Q. Once you have detected the death of any of them --

A. The one that makes sense to me is the watchdog supervisor. That is the easiest place to do it. That's the place where most people do it. The monitor CPU can't see which tasks are running necessarily, doesn't have visibility to all of them, but the watchdog supervisor should, and should have been designed that way.

Q. So, we exclude task X, and we just look at the other [redacted] tasks, I think I counted [redacted], is that right?

A. I think that's the same number that is in my report, yeah.

Q. So of the [redacted] tasks, excluding X, if [redacted] of them were to die, system failed, is there anything that is going to detect it?

A. There is nothing that detects it. So not even changing the brake switch detects it, so you have all these other tasks that are supposed to be doing something. For example, if spark on cylinder number one, if that task never runs again, then you're not going to have a spark in the first cylinder. Now, that is not going to because a UA, but it is an issue. You are not burning the gas, it's exhausting out of your exhaust pipe every time that the cylinder goes up and down.

Q. Have you reached a conclusion on whether this shows a defect in the software?

A I have.

Q. What's your opinion?

A. My opinion is that the watchdog is defective and should have detected all of them quickly as possible.

Q. And if a watchdog detects them, what are they supposed to do?

A What the watchdog should do, and the one I believe that it will do is for this one millisecond task, if that task dies and doesn't run again, then the watchdog correctly resets the ECM in that case, it actually happens very quickly. It can happen within one millisecond plus the [redacted] millisecond reset time. so in that 11 feet at 60 miles an hour, less feet. At half the speed it's five feet.

Q. And to the extent you can, can you describe for us what a vehicle would do in the vent you have a reset if you're driving down the road?

A. I'm familiar with testing that's been done with respect to resetting ECM, in a couple of different ways, by killing, for example, that one millisecond task and also by just forcing a reset electrically. And the observation has been that if you were sitting at a stop sign, it's possible your car will stall when it resets because the engine is turning slowly. But if you're driving down the road you'll see the RPM drop briefly and then it will go back and continue.

Q. All right. Let's move to the next slide.

MR. BAKER: Your Honor, at this point I think we can let everybody back in.

THE COURT: Okay. We will continue on.

Q. (BY. MR. BAKER) When you say that the test is effectively infinite, what do you mean by that?

A. Well, there are so many different combinations of ways and times when this can happen that it's impossible to test them all. It would take a vast amount of resources, resources that I don't have in the source code room, but resources that even Toyota doesn't have with their, you know, actual vehicles and test tracks and test engineers and, you know. It's not something you can test into submission. Because just looking at the number of tasks deaths, each one can die by itself. That is just [redacted] combinations. All [redacted] could die at once. That is just one combination.

But when you add up all the ways that just two can die, or just three can die or just four can die, it turns out to be over 16 million possible combinations of task death. So how are we supposed to test task X, which we've already demonstrated UA, and all the other tasks that can die with, you know, one other task death, two other tasks dead, three other task dead. And then it actually gets harder than that because each task can die in different vehicle operating states. We've a seen one of those perfect examples, is if it dies when the brake was already pressed, any amount of press, lightly pressed or fully pressed, then it's completely different outcome than if the brake was not pressed.

And the same is true for if the cruise is on, not on. It matters also what happens next. For example, on that prior slide there was one task that was not detected. That task is involved in shifting the transmission. None of the testing to date that I'm aware of from either side has caused a transmission shift after killing that task.

Well, in an automatic transmission, you know, in a manual you move the gear. In an automatic transmission in Toyota's design software pushes electrons and electrons push something mechanic. And if the task that does that doesn't do that then your transmission is in an indeterminate state, and what if you needed to downshift or upshift in order for proper vehicle behavior.

So just killing that one task and saying no observed behavior, as Toyota's expert does, that's not enough information. We have to test all the things that the driver might do next, including if the vehicle then misbehaves, what will they do after that? Will they press the brake or not, pump or not, et cetera.

And there's also in addition internal software states. I talked about a million lines of code, 11,000 global variables. You would have to test each combination of task death in all of those different system states in order to -- basically there's too many tests to construct to be sure that nothing even worse could happen. That is, for example, an unintended acceleration, where no matter what you do with the brake pedal, let go of it or try it, the car won't stop.

Q. Is that infinite number of test combinations a reason for having a reasonable and appropriate design structure in place?

A. Yes. This is exactly the reason why you have to follow a process like Dr. Koopman says you have to when you're designing a safety critical system. Because those processes are designed so that even if you get something wrong on the main CPU, because you have two independent fault containment regions, the failure of one can be detected by the other, and it depends on whether it's an airplane or a car, what's the best thing to do, but when that's detected as, I don't agree with you and we both have an independent view of what should be going on, then you do something safe.

Obviously, in an airplane you don't just stop the engines and fall out of the sky. You have to do something else. But a car you do the safest thing you can with that scenario under what's known. What's working and not working.

Q. In a software development process we talked about, Dr. Koopman talked about, is the process just as important as the testing?

A. The testing -- I'm not going to say that vehicle testing like Toyota does is not important. It is important. But it tends to find the bugs that happen frequently. The ones that happen to everybody everyday. It doesn't happen to find the rare ones. So the process is equally important if not more important, because the process is what makes sure that even if you have bugs in there, which there will be, that those bugs and defects won't get through and cause a dangerous harm.

Q. Anything else with this slide?

A. Right. So in that infinite space based on reading the source code we were able to pick out a particular bit. We were interested in task X and what would happen from reading the source code and we were able to simulate in the code room that if we killed it in a certain way -- actually there's a couple of ways it could happen -- that that task would die and not run anymore.

And that's what we could predict would happen and we have test sampling from within that infinite space that confirms that Toyota, when they say we have layers of failsafe and you know, when they tell that to Congress and they tell that to NASA and they tell that to you, that's inadequate. That's not enough. They should have had the process in place.

Q. All right. Before we go to the next line, I did want to ask you. I think you told us earlier about your conclusions in terms of this case, but can you tell what you understand the facts to be in terms of the Bookout accident?

A. Sure. I understand that Ms. Bookout was driving a 2005 Camry, that she was driving south on highway, I believe it's called 69 near Eufaula, and that she was approaching an exist ramp and began to exit and slowed her vehicle, and that at some point on the exit ramp the car was not slowing when she was braking. And that she pumped the brakes in response, and told her passenger

what was going on. And that a little bit further down the ramp her passenger suggested pulling on the parking brake. And there are indications that the parking brake was indeed pulled and this resulted either from the parking brake or the service brakes or both in a skid mark of 150 leading to a crash site in a ditch passed a stop sign at the end of the exit ramp.
Q. Is it accurate to say in terms of the specific details about the reconstruction, you're leaving that Mr. McCort?
A. Yes.
Q. Okay. Do you have an ultimate conclusion in this case as to why the vehicle would not slow down in the scenario you described for us?
A. I do.
Q. What is that conclusion?
A. My conclusion is that a software defect has caused the unintended acceleration which could not be stopped through the pumping of the brakes and the braking. Not in time anyway to avoid the crash.
Q. And in terms of the specific task and death or how this occurred in this case, have you got some sample testing to show us about how you demonstrate that?
A. Yes, I have another vehicle test that was performed.
Q. Let's go to that slide. Tell us about this. Is

this one of the tests or combinations that Mr. Louden did?

A. Yes, sir. This is testing that was performed in a 2005 Camry by Mr. Louden and documented in his report in the Saint John case.

Generally we're looking at several different data plots of different signals that he was collecting during this. And I'm going to walk you through it step by step, but let me just generally orient you. That the red up here on the top is how fast the car is going. You can see that initially in the test starting from about 40 seconds he accelerated until a speed, I don't know the exact speed, I haven't looked at the chart in a while, but you can see it's around less than 100 kilometers per hour, so it's probably 45 or 50 miles an hour here. And then at the time of the dotted line, he's killed the task, and then he's collected some various data along the way. And we'll talk about what each of these mean in just a minute.

So you can see the dotted line of killing the task is at a time 59. So the first thing you notice is that the vehicle speed is about 45 miles an hour. I'm not being too precise there, might be closer to 50. And so, the next thing to notice is that you see this orange arrow right here, this is showing that just after the task died Mr. Louden let off the gas pedal. He had been accelerating steadily, he lets off here but, the speed of the vehicle remains 45 miles an hour. It's not responsive, so we have a loss of throttle control at that point.

To demonstrate that further, Mr. Louden shows that even if he tries now, let's say he wants to avoid an obstacle on the road or another car, he tries to use the accelerator, nothing happens. There is no change in the vehicle speed, no failsafe kicks in or anything like that. In fact, none of failsafes act in any way, if we're greater than 30 seconds in this test, ranging from just before 60 to -- right about here we have something a little bit before 100 that that happens, so maybe 35 seconds or so.

And if you look here, what's happened at the end that's caused this throttle cut and an engine stall [redacted] later is that Mr. Louden has let off the brake pedal, right here. So, because he was on the brake pedal even lightly when this task death occurred, you see the brake signal is this solid line is on, and then it goes down it's off the green line, so at that time he's let off the brake. And it's then about [redacted] after that that the throttle is cut by the brake echo in the monitor CPO. And then [redacted] after that we get an engine stall. And then because we're on the dynamometer we don't see the vehicle drop off before his test data collection ends.

Q. So he has his foot on the brake at the beginning of this particular test?

A. That's correct.

Q. And is this a test that explains to you that the foot on the brake, the UA can continue on?

A. Yes, I mean, my opinion is based on more than just this test, but this test is supportive of my opinion, that's correct.

Q. And let's move on -- before we go on to the next slide. Let me ask you a couple of questions.

This explains to us what can happen when you have a task death occurs, correct?

A. That's correct. That's one of the possible outcomes.

Q. And it shows or demonstrates or at least is supportive of what you said having a foot on the brake when it happens?

A. That's correct.

Q. And we've gone through a lot and I just want to try to bring it altogether if I can. And please correct me if I'm wrong. Task X dies in this test?

A. Right, so this test is a task X death only.

Q.   Task X contains throttle angle, throttle -- all sorts of things?
A.   Including failsafes, that's correct.
Q.   And you've told us I believe that one of the ways that task X can die if there is memory corruption?
A.   That's correct.
Q.   And if we have a memory corruption, task X dies, we have a corruption with the throttle angle variables?
A   But then the throttle can open wider or close, depending upon what the corruption value is.
Q.   Is this sort of a scenario that you think more likely than not occurred with Mrs. Bookout?
A.   Yes. I would just clarify that it may have involved other task deaths beyond just task X.
Q.   But it's task X that creates the UA?
A.   I believe so, yes.
Q.   Let's move on to the next slide. Talk about the -- did you do what's called a root cause analysis to reach your final opinions in this case?
A   I did.
Q.   Tell me what a root cause analysis is?
A.   Sure. A root cause analysis is a consideration of all of the possible factors that could have lead to, for example, a car accident or some other incident.
     And so, when doing a root cause analysis, it is

appropriate and scientific to consider all of the possible things that could have been involved. And so, for example, considering mechanical causes, like a mechanically stuck throttle; considering electrical causes and software causes; and also considering whether there could have been something like a pedal that was trapped under a -- a gas pedal that was trapped under a floor mat; or a pedal misapplication, human mistake.
Q. So in this case would you have considered other potential causes of a UA in eliminating those based on your analysis?
A. Yes. So in each case I studied the evidence, whether the evidence supported that as a cause or not, how strong the evidence was in relation to other evidence supporting other causes.
In some case I was able to rule out entirely a particular cause. For example, the pedal entrapment by a floor mat does not -- there is no evidence to support that in this case. And I went through step-by-step, including the software and other factors.
Q. Is it listed in here in a slide?
A. Yes, at a high level.
Q. And as far as this, did you also consider the sworn testimony we talked about earlier today of other people who claimed to have experienced similar unintended

acceleration?
A. I did.
Q. And in that process would you have looked at more specific things related to their occurrences in order to say they were substantially similar to this one?
A. I have.
Q. And did you include a list of those within your report that you used in this case?
A. I did.
Q. We'll go through the fact that you looked at it in a minute but I just want to make sure that those vehicles, are they all Camry's?
A. Yes. I looked at 2005 to 2009 Camry's.
Q. And in that range, would you consider the software related to the UA defect that you discussed today was substantially similar?
A Yes.
Q. Continuing on with -- so you evaluated what you put up here you think is the cause?
A. Right.
Q. Were you able to rule those out?
A. In some case I was able to rule them out. In some cases I ultimately concluded that they were less likely than the software cause.
Q. And let's go to the next slide.

Now, are you here to tell us that, 100 percent, you know what defect caused this wreck?
A. No.
Q. Are you telling us more likely than not what defect caused it?
A. Yes.
Q. And is it the UA we just discussed with the death --
A. That's my opinion, yes.
Q. Under same or similar circumstances to the some of testing?
A. That's correct.
Q. Go to the next one. By the way, is it possible to tell a defect in the software?
A. No.
Q. And does it relate back to the incident number of tests that would be required that are not capable?
A One reason is because of the large space of possible things that could have occurred. Another factor is that, unlike many safety critical systems I'm familiar with, there is essentially no logging of what happens inside Toyota's system. There is no, oh, we reset the processor at this time or, you know, just before the crash, for example, there is no information about the internal software state, how many tests were running or not running, what they were doing.

Effectively, you can think of it as when you reboot the engine, all of the evidence of what happened before is deleted.

Q. This jury has been told several times that the vehicle had been inspected and there was no mechanical problem with the engine or brakes or anything like that.

Assuming that to be true, what would that tell you as a software person?

A. Well, the inability to find any prior mechanical problem or mechanical problem after the accident is actually supportive of a software malfunction theory. That's what software does. It casts a misbehavior that doesn't leave any stuck mechanical throttle.

You know, a mechanical cause like a bent pedal or a stuck throttle can move mechanically, would leave evidence that the car might have malfunctioned before the incident or it would have maintained evidence after the incident.

So the software cause is -- the case where a software cause is strengthened by the lack of mechanical findings in inspection.

Q. In order to assess the software issues, you have to go through what we've only gone through here for the last four or five hours, you have to go through that process?

A. Yes.

Q. All right. The next point, please.
A. So to a reasonable degree of engineering certainty, it's my opinion that it was more likely than not, a task X death, possibly in combination with other tasks that occurred that day, causing a loss of throttle control and in inability to stop the vehicle's full momentum because of the vacuum loss. So she had a vacuum loss in the brake when Ms. Bookout pumped the brake.
Q. And you also, as far as your work in this case and others Toyota UA cases, had an opportunity to see the testimony of Mr. Arora who offers software opinions on behalf of Toyota?
A. Yes.
Q. Did you happen to see other depositions of other experts for Toyota?
A. Yes.
Q. Have you become familiar with the positions that Toyota has taken in terms of defending whether UA occurred?
A. I have.
Q. All right. Have you prepared a slide to discuss those?
A. Yes.
Q. All right.
A. So back in July of 2012, when I issued my initial

report, there was also a report that came from Toyota's expert at the same time. So we exchanged reports in the blind. And in that report, Mr. Arora took the opinion that, first of all, that Toyota had various layers of protection. We talked about hardware fail safe, software fail safes, system fail safes, et cetera.

But, and this is the important point, that just because you fail safe layers it's great that there are fail safes. And undoubtedly they are detecting some misbehaviors, but that doesn't mean that there aren't gaps and holes, as we discussed, and defects, even, within those layers. And Mr. Arora appears not to consider that.

Additionally, in the same report, he said that those fail safes would detect any single point of failure, which obviously has been proven false at this point.

Q. Why do you say they've been proven false?

A. Because we've demonstrated that a single byte can cause a UA that can go on until you run out of fuel.

Q. All right. Your next point?

A When we published those reports, Mr. Arora's response was to do additional vehicle testing that showed when the task X died it was -- the death of that caused the throttle cut and a engine stall when the driver braked.

And so then Toyota and their experts began to say, well, it's not a UA because when the driver brakes, it will stop the incident. And I said to them, no, that is not designed for that purpose, not 100 percent reliable, and depends on the what state the car is in at the time. And I told them that in October of last year, about the year ago.

From that time until this summer, Mr. Arora continued to say that this was the, quote, unquote, designed fail safe of the system, until it became apparent that if the UA began with the brake pedal pressed to any degree, that it would continue, as I just showed in that data, until the driver let go.

And so most recently in his deposition in this case, Mr. Arora says, it depends on how much fuel you have, how long this will go on, or your braking ability.

I just want to go back and I missed this point. If that brake echo check was designed by engineers to be a fail safe against UA, then it would not be designed to require the act -- the driver to act before it acted. Fail safes should act before the UA starts, before the driver notices, et cetera and not require the driver to notice at all or act in some way.

It would never require that a possible action is that the driver would remove their foot from the brake pedal, counter-intuitively, and also increasing a short term risk by letting the car speed up.

As you might not have a lot of braking power against a full throttle, but I guarantee you, as you let off on that pedal, the car is going to speed up. And if you pump back down you're going to lose your vacuum and then you're going to fighting the old fashioned way without power assist.

Q. We talked earlier about -- let's go see your next slide. You have done 13 chapters of a review of Toyota's software?

A. I have.

Q. In terms of the experts that have been offered by Toyota in these other cases, have they refuted or rebutted everything you have written about the system?

A. Very little, actually.

Q. Can you show us what they have not?

A. Yes. And I won't say 100 percent because maybe there is some small part of some of these chapters that have been rebuttal. So don't tell me to 100 percent.

But by and large, of the 13 chapters, I believe the count is 11 of them are not rebutted or refuted in any way. And these involve the stack potential overflow we talked about, the code complexity being untestable and unmaintainable, not violating -- not following their own coding standard, violating MISRA.

I guess, technically, the response there is that they didn't have to follow MISRA. There's no rule. The fail safe modes being disabled when task X dies. The watchdog supervisor being abysmal. The software architecture with the kitchen sink task and the control of the throttle and fail safes in the same task has not been rebutted.

The lack of E-vac has not been rebutted. The software bugs in the -- in my software bugs chapter. I understand from his deposition just last month that Mr. Arora has not looked at those. And the operating system defects, the unmirrored variables, and Toyota's misuse of it and the nonstandard operating system has not been rebutted.

I just have one more point. And that's also that, from what I've seen, most of Dr. Koopman's opinion, he does have one chapter. It's a large chapter, but most of his opinions, most of things you've heard from him have not been rebutted in anyway either.

Q. All right. Let's go to the next slide. Other stories, we've talked about those briefly.

Are you aware of whether Mr. Arora has actually taken some of these other depositions as part of his analysis in whether UA can occur?

A   Whether Mr. Arora as reviewed other similar incidents?
Q.   Yes, sir. If you know.
A.   I don't recall.
Q.   Very good. We've been through it once. I don't want to belabor, but you looked at other instances, other sworn testimony of people that claim to have been involved in UA's?
A.   Yes. To be clear, not all of it was sworn testimony.
Q.   Okay. And I think that goes to the part at the bottom of the screen?
A.   That's correct.
Q.   Where were the sources of this information?
A   I got the information about complaints about unintended acceleration from principally three places. One is, I searched -- NHTSA has an on-line database where you can go and complain about something that happens in your car. And I searched that data base for incidents that involve descriptions of unintended acceleration and reviewed those cases and have cited to solve them in an appendix in my report.
     I also reviewed Toyota's internal documents and those are that a customer has a problem with a car, Toyota will maintain a file on that car. They call it a

field technical report, FTR. I reviewed documents that Toyota's produced that relate to those.

And then finally also, I reviewed claims like St. John, Mr. Van Alfen.

Q. Did that include other depositions and sworn testimony?

A. Yes. With respect to the claims, it's generally sworn deposition and testimony.

Q. Let me hand you a report that is St. John.

MR. BIBB: We renew our objection.

THE COURT: Okay.

MR. BIBB: Do I need to object to each and every one of those. There are certain facts that need to be brought out. I can cross-examine him now and talk about it all when we come back tomorrow.

THE COURT: No. Unless you've got something that you didn't raise when we made our record outside the presence of the jury you need to raise it now, because I obviously won't have ruled on that.

MR. BIBB: Thank you.

Q. (BY MR. CLARK) Have you found your opinions, where you start?

A. Yes. I think it starts on page 75.

Q. What I want to do is just have you, kind of in a great detail, but in terms of general facts, that you evaluated for specific instances as part of your analysis in this case, I want you to tell me about those.

MR. BIBB: We renew all the objections.

THE COURT: Okay. And that will be so noted and so you don't have to do it for each and every one.

MR. BIBB: Thank you, very much.

THE COURT: Yes. It will be carried over for each one.

Q. (BY MR. CLARK) All right. Let's start with Barris Ford Hill incident.

A Yes, Mr. Barris Ford Hill reported unintended acceleration while driving a 2005 Camry while attempting to enter a parking space. The vehicle.

MR. BIBB: Excuse me. If he's going to read it he needs to read the whole thing.

THE COURT: Okay. Well, counsel, remember, I had ruled. I granted part of your objection, so I don't know.

MR. BIBB: Okay.

THE COURT: I mean.

MR. BIBB: I mean you know, I take that back, Your Honor. I'll bring this out on cross-examination the distinct differences.

THE COURT: Okay. And you still follow my previous ruling about the stuff that cannot come in?

MR. CLARK: Yes, ma'am. That's what I'm trying to do.

THE COURT: Okay.

Q. (By MR. CLARK) Go ahead.

A. While attempting to enter a parking space the vehicle suddenly accelerated and caused a crash into a guardrail and wall.

Q. All right. How about the Brown incident, Leigh Brown?

A. Ms. Brown was driving a 2007 Camry when she experienced unintended acceleration while she was merging onto the freeway.

Q. According to the information you had, did she press the brakes?

A. She applied the brakes but was unable to stop the vehicle.

Q. Let's go to Linda Chory. And let me back up. The Brown incident occurred August 5th, 2007?

A. That's correct.

Q. And Linda Chory, when did her incident occur?

A. May of 2010.

Q. And what vehicle was she driving?

A A 2007 Camry.

Q. What were the general circumstances of her incident?

A. The vehicle surged forward three times while stopped

after exiting an onto and off ramp causing an accident.
Q. All right. How about the next page, Doris Dejoie (ph)? When did that incident happen?
A. May, 2010.
Q. And what vehicle?
A. It was a 2007 Camry.
Q. Again, are all these vehicles that we're going to talk about ones that you have found software to be substantially similar to the 2005?
A. Yes.
Q. In terms of an UA event, did it have the same defects and some of the same problems that you described for us?
A. With respect to the relevant details, substantially similar, yes.
Q. With regard to this event in Texas, can you tell us what it was?
A. She was backing out of the driveway with her foot on the brake and the vehicle accelerated suddenly and would not stop.
Q. And Ezal, first name, Buled. What was date of her incident?
A. It's actually a gentleman. It was February of 2007.
Q. And what vehicle?
A. It was a 2005 Camry.

Q. Would you describe the facts of that?
A. While entering a parking space, the vehicle accelerated over a curb, across the sidewalk, through two fences and over a cliff.
Q. Did he apply the brakes?
A. He applied the brakes but was unable to stop the vehicle.
Q. How about Elise Hazel?
A. I think it's Elsie.
Q. Elsie. When did she have an incident?
A. Sometime in 2009. I didn't note the specific date here.
Q. And what vehicle was she driving?
A. It was a 2008 Camry.
Q. And generally, what was the incident that she experienced?
A. While she was parking the vehicle, accelerated forward through a window of a store. She applied the brakes but was unable to stop the vehicle.
Q. Mr. Manfred Heinrick, what vehicle was he driving?
A. Mr. Heinrick had a 2007 Camry.
Q. Did he experience multiple incidents?
A. He did. He experienced about three different incidents over about a five-month period.
Q. One on May 24th, 2007?

A. That's correct. That was the first one.
Q. And tell me about that experience.
A. He was on a highway and the cruise control got stuck at 65. And after hitting the brakes, the vehicle accelerated up to 85. He applied the brakes but was unable to stop.
Q. Do you have a date here for the second incident?
A. August the 12th, 2007.
Q. And what did he experience the second time?
A. In this case he was merging into heavy traffic at about 30 miles an hour. He stepped on the -- though he stepped on the brake with both feet, the vehicle continued to accelerate.
Q. The last one was in September of 2007?
A. Yes.
Q. What was describe that happened?
A. He was stopped at railroad crossing and the vehicle accelerated on its own. The brakes were applied but it didn't stop the vehicle.
Q. The next one, James Highland from Ohio. What was the date of that incident?
A. It was in May 2010.
Q. What vehicle was he driving?
A. A 2009 Camry.
Q. Can you describe for us, generally, the incident?

A. While he was exiting the highway with a cruise control at 65, he touched the brake pedal and the car's engine immediately began to race to full throttle. He was able to stop to vehicle by shifting to neutral.
Q. Anita Gorge, when was her incident?
A. December of 2009.
Q. And what vehicle was she driving?
A. A 2005 Camry.
Q. Can you tell us about her incident?
A. She was slowly pulling into a parking space with her foot on the brake pedal and the vehicle suddenly surged forward. It jumped a curb in front of the parking space, hit a tree and slammed into a steel parking meter.
Q. Colleen Lambert, when was her incident?
A. July of 2008.
Q. What was she driving?
A. A 2005 Camry.
Q. What was her experience?
A. She was going about 20 miles an hour, coasting into a parking lot when the vehicle accelerated on its own. She applied the brakes, which was seen by her brother, Jim, but was unable to stop the vehicle and collided with another vehicle.
Q. Mr. Lee, when was his incident?
A. Mr. Lee was June, 2010.

Q. And what vehicle was he driving?
A. A 2007 Camry.
Q. And what did he experience?
A He was at a stop in a parking lot, and as he applied the brake, the vehicle accelerated on its own toward a vehicle in front of him.
Q. Amed Master, did he multiple events?
A. Yes.
Q. What vehicle was he driving?
A. A 2009 Camry.
Q. What his first date -- first event?
A. March of 2010.
Q. What was his experience at that time?
A. While he was entering the highway, the vehicle wanted to continue to accelerate. He applied the brakes but was unable to stop the vehicle.
Q. What was the second incident?
A. It was two months later, May, 2010.
Q. And what was the circumstances of that incident?
A. The vehicle accelerated for about 10 seconds while driving at 50 miles an hour.
Q. Do you know if he applied the brakes in that instance?
A. Not from my notes here.
Q. Cynthia Neil, when was her incident?

A. In December of 2007.
Q. What vehicle was she driving?
A. A 2007 Camry.
Q. And what was the circumstances of her event?
A. While she was pulling into a parking the space, the engine speed surged and the vehicle surged forward over a snow bank and hit a guardrail and tree. She applied the brakes but was unable to stop the vehicle.
Q. Mary Creeks Morrison, when was her incident?
A. May of 2008.
Q. And what vehicle was she driving?
A. A 2008 Camry.
Q. And what was the circumstances of her event?
A She was on a highway driving about 60 miles an hour, while passing a vehicle and it suddenly surged to 80 miles an hour.
Q. Did brake application stop?
A. She applied the brakes but was unable to stop the vehicle. She called 911 during the event and was told to put the car into the park and turn it off. Doing so stopped the vehicle.
Q. Roger Rick, when was his event?
A. September of 2010.
Q. And what was he driving?
A. A 2008 Camry.

Q. What was the circumstances of his event?
A. He was coming to a stop at an intersection and the vehicle jumped forward with high engine speed.
Q. Charles Sheppard, when was his event?
A. I just have her spring of 2008.
Q. And what vehicle?
A. A 2007 Camry.
Q. What were the circumstance of his event?
A. He placed his foot over the brake pedal when the car accelerated and caused an accident. The Toyota representative inspected the vehicle and couldn't find anything wrong.
Q. Heather Skelton, when was her event?
A. June of 2010.
Q. What vehicle was she driving?
A A 2007 Camry.
Q. What were the circumstance of her event?
A. She was at a complete stop and the vehicle surged ahead unexpectedly. She still had her foot on the brake when the vehicle surged.
Q. Margaret Schwarzman, what vehicle was she driving?
A A 2005 Camry.
Q. And when was her event?
A. August, 2007.
Q. What were the circumstances of her event?

A While she was turning left onto a residential road near her home, the vehicle accelerated out of control, causing her to hit a curb and crash into a parked vehicle. She was unable to control or stop the vehicle by applying the brakes.
Q. Paul Van Alfen, what was the date of his event?
A. November, 2010.
Q. What vehicle?
A. A 2008 Camry.
Q. Is this the one that Dr. Koopman mentioned?
A. Yes.
Q. Is this the one in which you mentioned?
A. Yes.
Q. What were the general circumstances of this event?
A. Mr. Van Alfen was traveling with his wife and two passengers. And they were exiting the highway in Utah. And the vehicle maintained its speed when he did not want it to and caused a crash at the end of the ramp.
Q. The last one here on your list is a Joel Wyenn. What are the circumstances -- what vehicle?
A. 2005 Camry.
Q. Do you have a date?
A. I have May, 2006.
Q. And what was the circumstances of that event?
A. While slowly pulling into a parking space, the

vehicle moved forward unexpectedly, jumped the parking lot and crashed into a concrete wall. He also had a prior incident two months earlier in which the vehicle engine was racing.

Q. Does -- and was this report actually written for another case?

A. Yes.

Q. What's the name of that?

A. That's the St. John case.

Q. Ida St. John?

A. Yes.

Q. What vehicle was she driving?

A. A 2005 Camry, like this one.

Q. Generally, what are the circumstances of her accident?

A. The car accelerated away from the stop sign and she went through a schoolyard and hit a concrete -- impacted a tree and a concrete column where the vehicle came to rest.

Q. And you've reviewed and analyzed the events we've just discussed?

A. Yes.

Q. Based on the information that you have, is it your opinion that these cases, more likely than not, also suffered a UA as a result of the software?

A. Well, I haven't done a root cause analysis on all these cases. But what I have done is, as an engineer, working in trying to debug complex systems over the years in my career, I have found it extremely useful in terms of understanding where the defects are, what kinds of misbehaviors can occur, to review and study complaints of users who say the system isn't working right.

And these incidents for which mechanical causes do not appear to be the cause, and software failure is consistent with the description of the accident, informed me, as a set, that there's a pattern and that pattern informed my analysis and source code and it informs my analysis of this specific case.

Q. And have we gone over all of your cases, specific opinions in this case?

A. Yes.

Q. And I think you mentioned earlier, but to be sure, are those to a reasonable degree of engineering certainty?

A. Yes.

Q. All right. Now I want to shift gears just for a minute and ask you some questions about the work that was done by Mr. Arora.

Have you reviewed his deposition in this case that was taken?

A. I have.
Q. And specifically September 24th.
A. Sounds about right.
Q. Does Mr. Arora address some of the issues you've discussed here today?
A. Yes.
Q. Did Mr. Arora do any vehicle testing on track that he talked about in his deposition?
A. Yes.
Q. And did he perform some tests at 45 miles an hour?
A. He did.
Q. Do you understand that Mr. McCort has testified that he believes that from the skid marks being left, that the speed of vehicle in this case was around 40 miles an hour?
A. I do.
Q. Did any of Mr. Arora's tests that you reviewed change your mind about your opinion in this case?
A. No.
Q. Can you describe for us, generally, the test that you performed, in terms -- and I know he did some at different speeds, but I want to focus on 45 miles an hour.
A. There was a set of tests. As I understand, the vehicle was always operated at 45 miles an hour. It was

always a 2005 Camry. And experiments were performed with tasks, one at a time. And that a certain spot on the track where there was a cone or a marker, the brake was pressed with 60 pounds of force. And then the vehicle was stopped and there were cones placed at 50, 100, 150 feet and every 50 feet beyond that.

Q. And did he also run a test applying 112 pounds of pressure?

A. He did.

Q. And we're going to focus on the 60 pounds?

A. That's correct.

Q. Was there any specific paperwork put together that describe the exact distance, stopping distance, for a test?

A. If there was I couldn't find it. I got a big hard drive with 50 gigs of stuff.

Q. In terms of the stopping distance of the vehicle, once it goes through, is the only way to determine the distance, is to look at the cones?

A. Yes. That's a reasonable way to do it.

Q At the time the brake is applied as it goes through and we look at these tests, what position was the throttle in based on your review of these cases?

A. My understanding of the tests is that the throttle was not open at the time.

Q. All right. So in terms of the test, as they are headed toward the gate, the brake is applied at 60 pounds of pressure and the throttle is released?
A. Yes. But that may not apply to all of the cases. But the ones that you and I focused on, certainly that's the case.
Q. All right. Can you pull it for us to look at?
A. Yes.
Q. All right. Let's take a look at ATS-10511. Is this your starting gate?
A. Yes.
Q. Is that two cones there?
A. At the starting gate? Yes.
Q. Does that what drives through those?
A Yep. That's what I see there. Yeah.
Q. As you go through there, are there cones and this would be at 100 feet and this would be at 50?
A. That's what I see.
Q. All right. Is that the vehicle at a stop?
A. Yes.
Q. All right. Can you -- and to help us did we finally come to the location?
A. Yes. A little zooming helps.
Q. So two gates were entered, correct?
A. That's right.

Q. So in terms of stopping the car with no throttle and the service brakes only at 45 miles an hour, where did the vehicle stop?
A. Before 100, certainly.
Q. Let's take a look at -- just so the jury is clear. All of these are at 45 miles an hour?
A. They are all 45 miles an hour and with 60 pounds of braking force, which is the lesser amount of braking force that he applied in his experiments.
Q. There's no throttle?
A That's correct. This is 13. Can we see the enhanced photo of 13.
Q. Based on your review of this test was he able to stop the vehicle with service brakes only, no throttle in less than 100 feet?
A. Yes.
Q. At 45 miles an hour?
A. Yes.
Q. Let's to go 15.
MR. BAKER: We had to reboot, Your Honor. That one won't play, Your Honor, let's do 23. Can we have just a second?
THE COURT: Certainly.
Q. (BY MR. BAKER) If we can see the still. So again this is test 15. Addition 45 miles an hour when brakes

are applied, no throttle, correct?
A. That is correct.
Q. This one looks like it got almost to 100 feet before this stopped?
A. Almost to a 100.
Q. 23, Your Honor, two more. Can we see the still for 23? Again, 45 miles an hour at the time brake is applied, no throttle, was this vehicle able to stop in less than 100 feet?
A. Yes.
Q. The last one is 25. Is the vehicle again stopped at less than 100 feet?
A. Yes.
Q. All right. Put up 5726, please. The jury has already seen this in evidence, Mr. McCort's scene diagram. You've seen this before?
A I have.
Q. And there has been a great deal of discussion about the skid mark that is out there, do you understand that the total length from beginning to the back tire where the car rested was approximately 100 feet?
A. No.
Q. What did I say? I'm sorry, 150.
A. You said 100, 150.
Q. I need to reboot. Can we see here, we see 101 on

the pavement and then another 24 for 125 on the pavement?
A. That is my understanding, approximately.
Q. And assume for me there is another six feet of improved payment for approximately 131.
A. Okay.
Q. Assume that. If we assume at the beginning of this skid mark that Ms. Bookout is applying her service brake and not her accelerator, and she is going 45 miles an hour and her throttle's not open, based on Mr. Arora's test that we just saw what should have happened?
A. I think the vehicle would have stopped. There's 101 foot section there to the fog line, I think that the vehicle would have stopped in that distance.
Q. If Toyota's correct in what they've been talking about in this case and there is no UA, and Ms. Bookout left this skid mark by her service brakes alone, she's not on the throttle, what does Mr. Arora's test tell you when she is traveling?
A. His test, the one we showed you with the throttle closed, so demonstrates that if it takes 150 feet or more to stop, more since an impact speed of 20 miles an hour, the throttle must have been open.
Q. If the throttle was not open should the vehicle have stopped according to Mr. Arora's test?
A. Yes.

MR. BAKER: Move to admit all the prior exhibits including the pictures. At this time I would tender the witness.

THE COURT: Mr. Bibb, do you wish to wait until the morning.

MR. BIBB: I would really would, Your Honor, it's been a long day.

THE COURT: It has. Ladies and gentlemen, we are going to be in recess for the day, it is 20 till five. I will see you tomorrow morning at 9:00. Do not discuss the case, do not begin to form any opinions about the case. And remember to check in the jury assembly room in the morning.

Thank you very much and have a good evening. All rise when the jury is exiting.

IN THE DISTRICT COURT OF OKLAHOMA COUNTY
STATE OF OKLAHOMA

Jean Bookout; Charles Schwarz, individually and as Personal Representative of the Estate of Barbara Schwarz, deceased; Richard Forrester Brandt, as Personal Representative of the Estate of Barbara Schwarz, deceased,	)	
Plaintiffs,	)	
vs	)	CJ-2008-7969
Toyota Motor Corporation; Toyota Motor Sales, U.S.A., Inc.; Toyota Motor Engineering and Manufacturing North America, Inc,; Aisan Industry Co., Ltd.,	)	
Defendants.	)	

* * * * * *

TRANSCRIPT OF PROCEEDINGS

HAD ON THE 14TH DAY OF OCTOBER, 2013

AFTERNOON SESSION

BEFORE THE HONORABLE PATRICIA G. PARRISH

DISTRICT JUDGE

Reported by: Kim Lewin, CSR

I n d e x

THE COURT: We're back on the record in CJ-2008-7969, members of the jury are present as well as counsel. And I assume that Ms. McAdams is still sick?

MR. BIBB: McAndrews.

THE COURT: Pardon me. Mr. Barr, if you would please come back to the stand, sir, I'll remind you, you are still under oath. And Mr. Bibb, you may continue or being your cross-examination.

MR. BIBB: Thank you very much, Your Honor.

CROSS-EXAMINATION

BY MR. BIBB:

Q. Good morning, Mr. Barr.

A. Good morning.

Q. Let's begin by talking about your experience with automotive software and unintended acceleration. Before -- before you were hired by plaintiff's counsel to do work on cases against Toyota, you had never done any research into unintended acceleration, had you?

A. I had not.

Q. None of your work before getting involved in this work against Toyota involved software design work or analysis for automotive engine control systems, is that correct?

A. That's correct.

Q. And you'll agree with me that you have not seen

other automobile manufacturer's software source code, have you?
A. That's correct.
Q. And you have not talked with anyone who has actually seen other automobile manufacturer's electronic throttle control systems to know whether Toyota's use of global variables is unusual in the field, have you?
A. I've talked with those in the automotive industry about software generally, but with respect to other manufacturer's electronic throttle control systems, I don't know specifically and don't have any information about how many global variables they use.
Q. I think you told the jury yesterday that you spent countless hours working on matters involving the Toyota electronic throttle control system, is that correct?
A. I do.
Q. Do you have any estimate as to how many hours you put in on this?
A. I don't.
Q. Would it be literally hundreds of hours?
A. Probably be thousands of hours.
Q. Like two, three thousand hours?
A. I don't know for sure.
Q. And I understand that for each of those hours you charge $400 an hour, is that correct?

A I don't think that is correct.
Q. What is correct?
A. My current rate in this case is 525 an hour.
Q. I didn't mean to undersell you. Has all of your work been done at $525 an hour?
A. No, it hasn't.
Q. Has some of it been done at $400 an hour?
A. I think about that price.
Q. But these countless, perhaps thousands of hours you charged between 400 and $525 an hour, would that be correct?
A. That's correct.
Q. All right. Now, you understand, Mr. Barr, that the reason we're all in this courtroom is we're trying to determine the cause of Ms. Bookout's crash on September 20, 2007, do you understand that?
A. Yes, I do.
Q. Let's talk a little bit here about the circumstances of the crash. You understand that Mrs. Bookout and Ms. Schwarz were traveling south on Highway 69 towards Eufaula, Oklahoma, right?
A. In the vicinity of, I'm not sure if they had reached Eufaula yet.
Q. I just said they were going towards Eufaula?
A I don't know whether it was north of Eufaula or not

but in that area, she was traveling south.
Q. It's your opinion the cruise control was not on at the time of the crash, right?
A. That is my understanding of the facts.
Q. And you know the speed limit on Highway 69 was 70 miles an hour, are you aware of that?
A. I'm aware of that.
Q. Now, you don't believe that the unintended acceleration incident that Ms. Bookout claims occurred on Highway 69, do you?
A. It's my understanding that it began on the exit ramp.
Q. And you understand that Ms. Bookout successfully slowed her vehicle and exited on the exit ramp for Texanna Road, correct?
A. I do.
Q. And you know that Ms. Bookout told us in her deposition that she applied the brake to slow the car so as to exit on Highway 69, do you understand that?
A. I do.
Q. Now, you also understand Ms. Bookout and Ms. Schwarz took the wrong exit to get to where they were going, do you understand that?
A. I read that.
Q. And you understand that the exit ramp there from the

Highway 69 to Texanna Road is fairly long, somewhere in the neighborhood of a thousand feet or more?
A. That is about the distance I understand.
Q. It's your belief that the alleged unintended acceleration incident began somewhere on the exit ramp, correct?
A. That's my understanding.
Q. It occurred somewhere before the tire mark that is at about 150 feet from the point of rest, correct?
A. I would agree with that.
Q. Now, you don't know what the exact speed of the vehicle was when the malfunction allegedly began, do you?
A. I don't.
Q. And you don't know precisely the throttling at the time the malfunction allegedly began, correct?
A. That's correct.
Q. Now, in your deposition did you not tell us that it was very likely that the throttling was significantly less than halfway open when the vehicle began to malfunction?
A. That's correct.
Q. All right. Now, we don't -- in your deposition you said that you didn't have sufficient information to determine whether Ms. Bookout's foot was on the gas pedal or brake pedal or neither pedal when the alleged

unintended acceleration began, is that correct?
A   That's correct, the precise timing, we don't know.
Q.   Well, let's take a look at those three possibilities that we've got. And Mr. Barr, if you can't see this, let you me know, okay?   So we've got one and I'm just going to put gas pedal, that her right foot was on the gas pedal, do you agree with me, that's one option?
A.   That is one of the possibilities for when the unintended acceleration or software malfunction began.
Q.   Based on all the vehicle testing that either Mr. Louden did with you and Mr. Arora's vehicle testing, when Ms. Bookout stepped on the brake pedal, the distance from the gas pedal to the brake pedal, when she does that we get brake echo check as soon as she held the brake pedal down for longer than 2/10ths of a second, right?
A.   In the limited testing that's been done within the essentially infinite space of vehicle and software states, it is true in that limited testing the brake echo check has stepped in if her foot was on the gas pedal and then she transitioned to the brake pedal. However, that doesn't rule out that the brake echo would not act for a number of reasons.
Q.   In all the tests, it can be limited or extensive, we'll talk about the testing later. Every time you go from the gas pedal to the brake pedal in a 2005 Camry

like Ms. Bookout's, what happens then is you get a [redacted] degree throttle, correct?

A. The first effect of the brake echo failsafe, if it acts, is to cut the throttle to [redacted] degrees or about, between five and 10 percent.

Q. So if Ms. Bookout had her foot on the gas pedal and allegedly the task died, and puts her foot on the brake pedal, the throttle would go to [redacted] degrees based on the testing that we've done and the way the software is written for Toyota and that would be the condition of the throttle, correct?

A. It would.

Q. Now let's go to number two, and that's -- she has got her foot on the brake pedal. Okay. Now if she has got her right foot on the brake pedal, she doesn't have her foot on the gas pedal, right?

A. That's correct.

Q. So we've got no gas pedal. And if we don't have our foot on the gas pedal, then the throttle would be at idle, correct?

A. I don't think we know that with certainty.

Q. If the car's operating properly, the throttle would be at idle, correct?

A. It takes some time to return to idle and we wouldn't know precisely when the software malfunction began and so we can't say that with certainty.

Q. Well, let me ask -- while they get the throttle body out. You'll agree with me when you say it takes some time, the springs on the throttle body, actually when the motor is -- is giving instructions to no longer keep it open, the springs will close this plate in the throttle body, right?

A. What you said is true, but that's not what happens when someone takes their foot off the accelerator pedal.

Q. What actually happens is -- you've got a lot of mass in that engine, right? The rpms will slowly come down but the throttle itself will close within a second when the pedal is released and it's in the idle position?

A. When the pedal is released, software will drive the throttle closed, if it's working properly like the hot water valve that I talked about. But that's different than what you're talking about with the [redacted] degree spring return. That is a mechanical return that happens in that failsafe, but would not happen here.

Q. A mechanical return on that throttle like I showed to the jury, but that happens irrespective of the software, I mean, that is a mechanical system separate and apart from the software?

A. It will only happen if the software stops controlling the throttle.

Q. Let's assume she's got her foot on the gas -- on the brake pedal, she's got no gas pedal, assume your car is running right so you're at idle, and then your task dies while she has got her foot on the brake pedal, right? In none of the testing that's been done has the throttle ever opened unless you had the cruise control on, right?

A. That's correct, but that is because that's a test that's not been performed.

Q. Well, we saw -- talked about that cruise control test that you showed the jury yesterday on the slides, because the throttle does open because it's trying to get up to the set speed, do you remember that one?

A. Yes.

Q. If we have a task death and the gas pedal is at idle, like everything is working right on this car like it had every day for the two years she owned the car, then it's going to stick the throttle at idle when the task dies, right?

A Again, sir, the time matters down to the millisecond level, in that scenario maybe less than a millisecond, and so it matters when the task dies relative to when she lets her foot off the gas pedal. The idle return you're talking about requires software control and if the task dies in between when she is releases her pedal, but before it goes to idle it could continue to be open wider

than idle.
Q. You don't know what the throttle position was though, correct?
A. That's correct.
Q. If she is coming down that ramp riding the brakes slowing down because she knows she's got a stop sign at the foot of the ramp, let's assume she has been on the brake for two or three seconds, the engine is going to be at idle, right?
A. If she's been on the brake for two or three seconds then that hypothetical I would agree it would likely be at idle.
Q. And when you're applying the brake pedal and your engine's at idle, it's no different than pulling up to the stop sign or a red light here at Park and Harvey in front of the courthouse, isn't it? Because that is the way you normally pull up to a stop, right?
A. That's correct.
Q. And so the brakes are going to stop the car even if the task is dead and you're on -- and it freezes the throttle at idle, right? The brake's going to stop that car, right?
A. That's correct.
Q. Now I've got the third option. And that's no gas, no braking. Okay. Now, under this circumstance she

doesn't have her foot pressing on either pedal. Then if she goes off of the gas for two or three seconds, she's clearly going to be at idling, right, do you agree with that?

A. That's correct. If she has been off for several seconds.

Q. And whatever happens, if this mysterious task X dies, then again, you'd be at idle, when it dies and the throttle would be stuck at idle. And when she puts the brake on, she transitioned the brake switch to go to a failsafes, right, on every test that you've seen run, isn't that correct?

A. If the only memory corruption affect was to kill task X and in the limited testing that it's not 100 percent definitive for a number of reasons we can get into if you want to look at my report, what you say hypothetically would occur.

Q. Now, so these are the situations that we've got with this car coming down off that exit ramp, either as soon as she steps on the gas pedal, what you call limited testing, essentially all of the testing shows a failsafe, she's on the brake pedal, not on the gas pedal, she's at idle, she stops the car. She's not on the gas and not on the brake, it's idle, goes into failsafe when she steps on the brake pedal and she would stop the car then too,

wouldn't she because it would be idle?
A.   In those hypothetical scenarios, which are not consistent with my understanding of this accident, that's what would happen.
Q.   You know that Ms. Bookout said that she had her foot on the brake and that she pumps the brake, you are aware of that?
A.   I am aware of that. I believe she said six or seven times.
Q.   And she even said she removed her foot from the pedal during that pumping, you're aware of that?
A.   I don't remember her precise words with respect to that.
Q.   Have you read Ms. Bookout's deposition?
A.   I have.
Q.   Do you recall her saying on page 35 at the end of that page -- could we see that, Mr. Doyle? There about page 35, line 24. Again, this was Mr. Jennings asking the questions. "Do you remember pumping the brake? You mean applying the brake and then taking your foot off of it and applying it again?" And her answer was "yeah."
     Now if she takes her foot off the brake and applies it again, in that situation the brake echo check would operate as long as she took her foot off the brake for more than two -- two hundreds of a thousandths of a

second, 2/10ths of a second, she would transition that brake switch and as a result the brake echo check would come in and would shift to failsafe, isn't that true?
A. If during the pumping her foot came fully off the brake pedal, and not 100 reliably, that the brake echo acted, then the failsafe of cutting the throttle would have kicked in. However, pumping can occur -- and I don't think this answer is clear from a technical point of view, pumping can occur, as I mentioned yesterday, without full removal of the foot. When most people pump, more than half of the people pump in one study that I've reviewed, they don't, they never get it completely off there.
Q. We're going to talk about that, that is the Cooper study?
A. That's what I'm referring to.
Q. You told us yesterday that you've written some reports on Toyota unintended acceleration, right?
A. Correct.
Q. One of the reports you issued was in September, September 17, 2012, do you recall that? It would be your rebuttal report?
A. I do.
Q. And you described the testing -- you described this vehicle testing that you're talking about today as

limited. The vehicle testing performed by plaintiff's experts using a testable version of Toyota's ECM, that is the engine control module software, was necessary to scientifically confirm our discovery of a single point failure. So this testing scientifically confirmed these findings, isn't that right?

A. The testing that was performed confirmed that the Toyota failsafes have gaps and that there are single points of failures and unintended acceleration can occur while no failsafe acts, and they can also confirmed that sometimes there's a failsafe that will act after the driver, after the UA has already occurred.

Q. But the one thing that the scientific testing that you did was confirmed every time, in every one of the tests of the vehicle that you ran that when the brake switch was transitioned, it triggered a failsafe, correct?

A. In the limited testing that was performed in much larger test space, all the tests eventually if the driver transitioned the brake switch, that is the unintended acceleration already occurred, the software malfunction already occurred, then when the driver acted, sometimes counterintuitively, by removing their foot from the brake, then there was a failsafe that a source code shows is not reliable 100 percent of the time. But in the

limited testing it did act every time.
Q. It did act 100 percent of the time when tested on a real vehicle, didn't it?
A. It will not act 100 percent of the time in real vehicles across a larger test scenario such has a billion driver hours.
Q. Have you ever in any test seen a failsafe not activate in a vehicle test?
A. In a vehicle test, that has not been shown. However, that's not how you prove a negative hypothetical in science.
Q. Now, you know after the crash the car has inspected by at least a dozen engineers and scientists, you're aware of that?
A. I hadn't counted them, but I was aware there were several inspections of the Bookout vehicle.
Q. And you're aware that among other things the accelerator pedal was removed from the vehicle and tested?
A. I understand that no problems were found with the accelerator pedal.
Q. And the brake switch was removed from the vehicle and tested, are you aware of that?
A. I am.
Q. And you're aware of that brake switch was found to

be operating normally, correct?
A. I do.
Q. So now, when we took your deposition you told us that you had not ruled out pedal misapplication as the cause of Ms. Bookout's crash, had you?
A. At that time of my deposition in early August, that was correct.
Q. And in fact, this accident can be explained by simple pedal misapplication, can it not?
A. No, it cannot.
Q. So by a combination of applying the gas pedal and applying the brake pedal at different times, coming down that ramp, you don't think you can reproduce this crash?
A. That's correct.
Q. Have you been to the scene?
A. I have not been to the scene.
Q. Have you seen the car?
A. I have not -- I have not seen the car in person.
Q. Have you tried to do any testing yourself of a vehicle to see if you can apply the gas pedal and then apply the brake pedal and have this accident occur?
A. I have not. That's not my role here.
Q. Now, you are aware, are you not, from looking at some of the research that you've done that there was this incident in Santa Monica, California where a gentleman

drove by the farmers market, correct?
A. I'm familiar with that incident.
Q. And that incident covered over 750 feet, did it not?
A. I don't know the precise distance.
Q. And it lasted for more than a few seconds, correct?
A. That's correct.
Q. And then you're aware in the Cooper study that you just mentioned to me that there was one driver in that study that froze up and plowed through the cone barrier at the end, correct?
A. I am aware of that.
Q. And you're aware in one of the tests where they were having people pump the brakes, one of the subjects pumped the accelerator pedal, you're aware of that too, are you not?
A. I am aware of that.
Q. And so it is -- you just can't rule out that Ms. Bookout might have pressed the accelerator pedal when she meant to press the brake, can you?
A. Yes, I can.
Q. And of course, if she did do, that when she finally got on the brake, even if there was a task death, the limited testing or the scientific testing that you mentioned would put it into failsafe, correct?
A. Yes. And then she would not have skidded for over

100 feet -- 150 feet.
Q. Well, if you put on the brakes you can press them hard enough to cause the brakes to skid, can you not?
A. I'm sorry, sir.
Q. Let me rephrase that. You can, whether you apply the parking brake or service brakes or both at the same time, can get tire marks from this vehicle, correct? You've seen that in some of the testing, have you not?
A. I'm sorry, I still don't understand the question.
Q. Never mind, I'll move on. Let me ask you about the vehicle testing, and we're going to look at one of those slides you showed in a few minutes. The vehicle testing that was done by Mr. Louden, in order to run those tests he ran that vehicle not on the road like we saw with Mr. Arora's test, he ran that vehicle on a chassis dynamometer, correct?
A. That's correct, we felt that was safer.
Q. And the chassis dynamometer has big rollers to allow the vehicle to simulate by rolling its tires against these rollers, simulate being on a road, correct?
A. That's correct.
Q. All right. But this is a no load dynamometer, correct?
A. I believe so.
Q. And so the vehicle is kind of free to spin against

those, even when the throttle is cut. It would take longer for the vehicle to slow down than it would if it was out on the highway, right?

A. That's correct, as I mentioned yesterday.

Q. Now, to cause task death in the tests that Mr. Louden performed, you had to modify the software source code, isn't that right?

A. That's correct, with Toyota's help.

Q. And when you modified the source code that allowed you to do fault injection testing, correct?

A. That's correct.

Q. And in other words, Mr. Louden would use a computer to inject faults to cause the task to die, is that right?

A. That's correct.

Q. I mean, these tasks did not die on their own, did they?

A. We were conducting testing and we wanted to see what happened when tasks died, so we killed them at a time of our choosing so that we could observe the outcome.

Q. In order to kill the tasks, you had to do it by reprogramming the engine software so that as to prevent certain portions of the software source code from running, so that allowed you to then stick the throttle, correct, by killing the task?

A. That's inaccurate, sir.

Q. In other words, you did have to remove parts of the source code, did you not?
A. No, we did not.
Q. But had you ever achieved task death in a Toyota engine control system without killing it?
A. All the testing that we did was under the conditions of the injecting a fault and observing the results.
Q. Never had one just die a natural death, have you?
A. That's not something we were standing around looking for, sir.
Q. The answer to that is no, you never had one just die, you had to kill it?
A. Not that we know of it, sir. It might have died and we wouldn't know it necessarily.
Q. Fair enough. You don't know of any tasks that have just died on their own, do you?
A. I do not, in the testing.
Q. Now at the end -- at the end of your testimony yesterday we looked at some of Mr. Arora's tests, do you remember those?
A. Yes, the videos.
Q. And you know from -- you've read his deposition, correct, taken back in August or September?
A. I've read several of Mr. Arora's depositions. I'm not sure which one you're referring to.

Q. The one in this case, I think you might have been asked about some of the testing that he did that we saw yesterday, do you remember that?
A. Yes.
Q. And you know that the purpose of Mr. Arora's testing was to see whether killing any task or combination of tasks resulted in longer braking distances, you understood that was the point of his testing?
A. That's correct.
Q. And in fact none of the braking distances were longer than expected because of task death, correct? It didn't make the car take longer to stop, did it?
A. Not that I was aware of, no.
Q. And you're also aware from Mr. McCort's testimony that if Ms. Bookout was applying the service brakes to cause that tire mark that we see, and the cops had talked about in the case, rather than the parking brake -- and let me back up for just a second. The test we saw with Mr. Arora, he was applying the service brake, correct?
A. My understanding is he was applying the service brakes only, not the parking brake, and he was applying that at 60 pounds of force.
Q. And the service brakes, those are the brakes that are operated by the brake pedal down underneath the dashboard, right?

A. That's right. What you think of as the brakes.
Q. And so Mr. McCort, if you recall, maybe you read his testimony, maybe the jury will recall, that if you were going to cause that skid mark that the police officers have told us about out there on the exit ramp at Texanna Road with the service brakes as opposed to the parking brake, the car would have had to have been going 60 or 65 miles an hour, do you recall that?
A. With respect to Mr. McCort testimony, I don't remember that specifically. But it's my understanding of Mr. McCort's testimony that it's his opinion that the parking brake was pulled. It's my understanding also of the police officer's testimony and other eyewitness testimony that the parking brake was pulled and involved in those skid marks.
Q. The jury will recall what the officer's testimony was, Mr. Barr. But the long and short of it, none of Mr. Arora's tests were run with the parking brake being appear applied to slow or stop the vehicle, were they?
A. They were not. My understanding would be that if you added the parking brake to the service brake, the stopping distance would have been even shorter.
Q. Now, Mr. Barr, I know you explained to the jury yesterday, but at your first try at artificially producing task death you made some errors, did you not?

A. I believe I made one error, sir.
Q. And when you wrote your original report you didn't know that a brake echo would cut the throttle to failsafe, did you?
A. At the time of my July 2004 report in the Van Alfen case I was not aware of that.
Q. And I understand and you explained you had limited time to review that monitor CPU source code prior to issuing your July 17 or July 18 2012 report, is that right?
A. That's right, this was the source code that was proceeded just a few weeks before.
Q. Now, you did have a chance to review it, and in fact, did you not remove about 20 percent of that source code to make it -- to facilitate your review of it?
A. The error that I made was based on not having the tool that's used to compile that, the assembler that I spoke of yesterday, and not having that assembler, it was my understanding that about 20 percent that of code belonged to another chip called the Sigma 2, and that the 80 percent that I reviewed as in the ESP-B2.
I later learned that the situation was flipped and that I should have looked at that 20 percent of the code as well in my analysis, which I did in my supplemental report which I filed in that case and the judge accepted.

Q. Well, you filed a rebuttal report two months later on September 17, 2012, did you not?
A. I did not, but I was not aware at that time.
Q. And you had additional on two months to work with this source code and you made the same error in that report, did you not?
A. I don't think that was -- I was still not provided the tool that would have changed my mind another any other evidence.
Q. So you made the same mistake two months later after you had two more months to work on this, as you made on July 17, right?
A. I'm not sure how active I was is in Toyota work during that time. I don't recall.
Q. But you didn't correct the error that you stated in your July report even two months later in your September report, did you?
A. As I stated earlier, as soon as I became aware of it, within ten days, I studied the extra 20 percent of the code, I wrote a supplemental report. The report I'm using in this case, the Saint John case, incorporates that analysis, that full analysis.
Q. Your theory has kind of changed over time because you had one theory, and then you discovered that it was not accurate so you had to do something else, haven't

you?
A. I don't believe that is accurate, sir.
Q. But you'll agree with me that your initial theory was that that monitor CPU would not cut the throttle, right, when it detected a brake change?
A That was an element of my understanding of the failsafe misbehavior, but my theory was then and remains that tasks can die, that as a by-product of tasks dying, unintended acceleration can occur and not all of those unintended accelerations will be caught before there is harm to the drivers, passengers and pedestrians.
Q. And you didn't learn of the mistake that you had made until you learned about it from Mr. Arora, did you?
A. That's correct. You said September 17th, I don't remember the date, when I issued my rebuttal report to his, he issued his rebuttal report to mine and it was at the time that I learned that I needed to look at the additional 20 percent of the code.
Q. Each side had to submit their expert reports written in the same day, is that correct?
A. That's correct. That's how this works. I pointed out errors in Mr. Arora's analysis at the same time.
Q. And you had to go back and kind of retract the statement that this brake echo would not shut down the throttle motor, correct?

A. That's correct, but it didn't change my ultimate conclusion in that case, and it's not relevant here to this case.
Q. Now, in addition to setting this failsafe, if this fault persists for more than, what is it, three seconds or so, the vehicle will stall out, correct?
A. That's correct. The brake echo check, if it's responding to task X death, will first, after that 2/10ths of a second, will cut the power to the throttle and then three seconds later, it could be slightly longer, three and a half seconds, but approximately three seconds later it will stall the engine.
Q. Ms. Bookout's vehicle didn't stall in this accident did it?
A. That's correct.
Q. We can all degree on that. And we can also all agree that this transition of the brake switch it has to occur for 2/10ths of a second, correct?
A. That's correct. And it will not be reliable. It may not work every time --
Q. I'm sorry, I apologize. The 2/10ths of a second, that's as fast as a blink of an eye, isn't it?
A. That's accurate.
Q. Now, you have not reproduced in vehicle testing your theory that there's a software bug that opens the

throttle and then the task dies, have you?
A. No.
Q. And you have not reproduced in vehicle testing your theory where there's task death and then the throttle is opened farther by a software bug or corruption, correct?
A. Right. So the second corruption that I talked about yesterday has not been demonstrated in a vehicle. We've not attempted to.
Q. All right. Now, do you know how big the throttle angle would have to be for the vacuum to the power assist not be replenished when the brakes are pumped?
A. I've seen various percentages for the throttle angle of which the vacuum brake -- vacuum assist to the brake doesn't replenish. NASA quoted around 30 percent, other experts have said -- NASA said 30 degrees, which is about a third of the way open; other experts have said smaller amounts, bigger amounts. I don't know the precise number, and in fact vary by vehicle.
Q. And you know Mr. Hannamann who's back here, he's done a little testing on that to try to evaluate that, correct?
A. I understand that.
Q. But you know that all the testing of Mr. Hannamann or NASA or the engineers that Toyota has retained in the case, the vacuum is always replenished if you're [redacted]

degrees, correct?
A. I don't know that I've reviewed all that data to say always, but certainly data that I've seen is consistent with that.
Q. And the data that you've seen would certainly indicate you would not lose vacuum assist if your throttle is at [redacted] degrees, correct?
A. My understanding is that's extremely unlikely.
Q. Let me talk to you for just a moment about these MISRA violations that you talked about yesterday. The first version of MISRA was put out in 1998, correct?
A. Yes, I believe it was March or April.
Q. Now, you understand from Mr. Ishii's testimony that Toyota's coding standard started being developed starting in 1994, you're aware of that?
A. I don't remember the year he stated. I believe he stated the 1990s. But it was updated during the time frame that this vehicle was designed. In fact, I reviewed the 2002 version which my understanding would be the version -- even though they weren't enforcing it, they were updating it every year or two.
Q. And Toyota's standards were established in 1997, you understand that?
A. You said 1994 earlier --
Q. That's when they started being developed. They were

then established by '97?
A. That's the number that sounds right, the date that sounds right to me.
Q. And when Toyota issued its coding standards there weren't any MISRA standards, were there?
A. If they issued them in 1997, that would be true.
Q. Now, MISRA was updated in 2004 to improve from the 1998 version, right?
A By and large the rules are the same. Some rules that had previously been recommended became required. And there was a rearrangement, the chapters were rearranged and the rule numbering system was rearranged a bit. By and large they are the same rules.
Q. Now the Camry that Ms. Bookout was driving was a 2005 model year vehicle, right?
A. That's correct.
Q. And you mentioned just a couple of moments ago that this was introduced as a new model in the 2002 model year, is that right?
A. No.
Q. This car was not -- this series Camry was not introduced as a 2002, and then updated year after year through I think about 2006, right?
A. Okay. I misunderstood your question. You're referring to the electronic throttle control use in the

Camry began in 2002.

Q. 2002 was the first year for the Camry. And since it was a 2002 model year vehicle, it wouldn't surprise you that the car actually went on sale some time during calendar year 2001, right?

A. That's correct.

Q. And you're familiar I assume in some of the work that you've done, that it generally takes three or four years to develop a new car, are you aware of that?

A. Yes, I am.

Q. So if we go back to 2001, we're looking at 1997 or 1998 when the development work to begin on this Camry with the electronic throttle control system, correct?

A. I think that is fair to say, but I would just point out that the actual development of the software is one of the later things that would be done in the three to four year process.

Q. So when the development of this vehicle began in '97, the MISRA standard weren't even in effect, were they?

A. Not for the vehicle we're talking about here. That's the 2005 Camry, you're talking about the 2002 Camry.

Q. And they went from year to year with the software being developed and adjusted and modified each year, is

that right?
A.   They did, in a very sloppy way. But in the 2005 model year, I believe it was, they redesigned the electronics. Went from two processors to one processor, switched from the ITron operating system to the OSEK operating system, and the updated their coding standard, and they designed a new monitor CPU. So given that level of effort, they certainly in around 2002-2003 time frame when they were doing these things for the 2005 Camry model, could have adopted at least the 1998 MISRA system.
Q.   Thank you, Mr. Barr. Now, Mr. Barr, I want to talk to you and talk now to the jury about just some of your slides, we're not going to go through all the slides. We'll talk about some of the slides with you, all right, and the first one is slide number three. This was when you had the book up here and I really want to focus on this third one here. I don't know if the jury can read it there. But this book that you published, it has Michael Barr's coding standard in it, doesn't it?
A.   It's the Barr Group's coding standard.
Q.   Right and that's different from the MISRA coding standard?
A   There is considerable overlap between the two, that's correct.
Q.   But it's different from the Toyota coding standard?

A. It is, and it's aimed at a slightly different set of imbedded systems developers than MISRA is aimed at.
Q. I know you're critical of Toyota creating their own coding standards, but the Barr coding standard is another coding standard out there, right?
A. It is.
Q. And if you compare to the Barr coding standard to the MISRA coding standard, you have violations in the Barr coding standard that didn't match up with MISRA's coding standard, right?
A. First of all, I would say there's considerably more overlap between the imbedded C coding standard book and MISRA. And second of all, someone who followed that coding standard would not violate those MISRA rules.
And this book is really not aimed specifically at safety critical systems developers. It's trying to take some of the lessons learned in safety critical and in MISRA and bring them down to things like this Nike fuel ban. In fact, I've been in touch with engineers who work there who are adopting this coding standard. And they don't need to design a safety critical system but they still want to maintain good software and they want to keep bugs out because if they have a million of these in the field and there's a bug, then they have to get everybody to update their software or do a recall, or

their reputation suffers. So that's what this book is trying to do. And actually in it says, look at the MISRA rules, those are smart people, they came up with good rules and you should follow as many of them as possible. And this book augments that by adding some additional stylistic rules to make it easier for programmers to understand their own code later when they review it and also the code of other developers.

Q. I think the question was, if you ran a MISRA checker on code written to Barr standard, you'd end up with a violations of the MISRA code, correct?

A. Possibly for the rules that are not overlapping.

Q. Now, did you tell us yesterday that there's always going to be some bugs in software, right?

A. Yep.

Q. So on the cover of your book, though, you put up here "zero bugs". That is not true, is it?

A. If you read what that means in the book, it says zero bugs is not achievable but you should structure your software process, your software architecture, your coding standard and your company safety culture for the purpose, the aim of trying to get there.

Q. Right. But you'll never get to zero bugs, will you?

A. That's correct. And that's not what that implies.

Q. All right. I won't judge a book by its cover.

Now, let's go to slide number six. And this was an example of writing software that you gave us yesterday, okay? Now, here, to make sure I understand it, the way it's read here you've got two numbers, two values, right?
A. That's correct.
Q. An A value and a B value. And the way this reads if A is bigger than B, like it's 10 and 4, and then you use the larger, right?
A. That's correct.
Q. If it's A is four and B is 10, then you'd use B?
A. That's correct.
Q. Use the larger value always, right?
A. That's correct.
Q. Now, if A is 10 and B is 10, this doesn't tell us what to do, it's going to use B, isn't it?
A. No -- well, t's going to use B, but that's going to be the right answer because whichever one you return in that case --
Q. It's not a larger value than A?
A. Well, the name of the function was chosen to explain the recipe.
Q. Make sure I understand, 10 is not larger than 10, is it?
A Another name for the function could have been chosen to emphasize that.

Q. I mean, this is a bug, isn't it?
A. No, it's not.
Q. Not a bug. Doesn't function the right way, though, does it?
A. It doesn't function according to the way I named the function, but real software would have a specification that said what was supposed to happen when they were equal. When I put together this slide, I certainly thought about the fact that they could be equal and what to do. And I thought, well, I only have view lines of PowerPoint and I'm explaining it to a lay audience, and so I didn't include a specific representation. But I would not describe that as a bug, sir.
Q. Not a bug. Not a bug. Let's go then to slide number 7. And yesterday you were talking about the throttle being like a shower that you turn on, do you remember that?
A. Yes.
Q. Do you think that is an apt analogy?
A. It's referring actually to the hot water.
Q. And because the throttle isn't like a shower, is it, because the shower you turn on and let go and take your shower and then turn it off. The throttle you'd have to turn that handle and then hold it open to keep the water coming, wouldn't you?

A. That's correct. In the throttle in the car you have to keep holding it open, because if you don't keep holding it open electrically, then it has a mechanical spring that will return it back to closed. But as long as the software is controlling it, to be clear, it will act just like you letting it -- set it and forget it.
Q. What we know is when the check -- the brake echo check kills power to the throttle motor, the throttle closes, right?
A. That's correct. If the software, whether it's functioning properly or not, stops controlling that, then the mechanical turn spring will take over.
Q. You also showed us a photograph of an engine control module, correct?
A. Correct.
Q. This is a 2008 Camry, is it not?
A The one I showed is a 2008.
Q. Is this a 2005 here, do you recognize that?
A. Yes.
Q. Okay. Now, you talk about the hostile environment that the system has to operate under. Do you know where the 2005 electronic control module in located in the Camry?
A. I believe the 2005 is under the dash.
Q. It's inside the passenger compartment, isn't it?

A. That's correct.
Q. It's in front of the glove compartment, correct?
A. That's a good description.
Q. Now, look at slide 10. You talk a lot about -- I'm not going to try to go over lengthy -- but you talk some about the NASA report, do you remember that?
A. Yep.
Q. And you've got a quote here on this slide from the NASA report, right?
A. Yes, I do.
Q. Now, you gave a lot of quotes from that NASA report, but the conclusion that NASA reached, could I see page 17 of the NASA report in executive summary? The last paragraph there, Mr. Doyle. You quoted first sentence there several times, but you didn't quote the second sentence, which is the final sentence of the executive summary that says the testing and analysis described in this report did not find that TMC, and you know that means Toyota Motor Corporation, and the ETCS-i means the electronic throttle control system, correct?
A. Correct.
Q. The testing and analysis described in this report did not find TMC ETCS-i electronics are a likely cause of large throttle openings as described in the VOQs. And the VOQs for the jury, are what, vehicle owner

questionnaires?
A Yeah, that is the -- when people call the National Highway Traffic Safety Administration, NHTSA, or going on to the web site to complain about their vehicle, they fill out a vehicle owner questionnaire.
Q. And you used some of those or reviewed some of the VOQs for those other accidents that you were talking about at the end of your testimony yesterday, correct?
A. Yes.
Q. So in other words, what this is saying here is the NASA testing and analysis described in this 100 and some odd page report did not find that Toyota's electronics are a likely cause of large throttle opening as described in reports from other consumers, correct? That's what they concluded?
A. That's correct.
Q. Now, go to slide 13. You quoted this second postulated scenario as a systematic software malfunction of the main CPU, opens the throttle without operator actions, and continues to properly control the fuel injunction and emission, correct?
A. Yes.
Q. That is one of the things you highlighted for this jury. But if we could look at the whole paragraph that sentence comes from, Mr. Doyle. 16, and it's the

paragraph right there -- that begins with the second postulated, it's the next to the last paragraph. This was the paragraph that you picked that sentence out from. And down at the foot of it, it goes on to say that the main CPU malfunction would be required to open throttle beyond five degrees with the accelerator not pressed and leave no failure code.

The NESC team, that's the NASA team, right?

A That's right, it's the research group within NASA.

Q. The NESC team examined the software code of more than 280,000 lines for paths that might initiate such an unintended acceleration, but none were identified, right? That was what they report?

A. That's what they said.

Q. Now, you mentioned that you had had some assistance in doing the software code review, is that right?

A. That's correct.

Q. And how many folks did you have helping you?

A. Well, I have three from the Barr Group team, I also relied from time to time on three others.

Q. That would be six folks?

A. That's correct.

Q. Or is that six including you?

A. That's seven, six helping.

Q. Mr. Doyle, can we go to page 11 of the NESC report?

This was the team that NASA had put together here. This is just the first page of it. They had 33 scientists and engineers working on this project, didn't they?

A. They did, but I think if you count the software engineers you'll find there is significant less than seven.

Q. Because they looked at it from all different angles, not just the software angle, right?

A. That's correct, and we were in the source code room looking at the software and we were focused on the software system.

Q. You can see the second page of those engineers.

And this report was issued in January 18 of 2011, correct?

A. I know that date, that's right.

Q. And they were charged with this research in March of 2010 so they worked about 10 months on it, correct?

A. That's sounds like the total length of time, something like that.

Q. With all of these people looking at not just the software, but the electronics, the computer scientists on this list, mechanical engineers, everybody looked at it, correct?

A. i don't think it says here whether someone contributed one hour or 100 hours, so we really don't know how much of a contribution some people made.

Q. But these people were all on that team, correct?

A. That's correct.

Q. Thank you, Mr. Doyle. Now I want to look at slide number 20, about halfway through. And I want to ask you about, this is this test, and apologize I've writing on your slide here. I want to make sure the jury and I understand this slide. I'll just focus on the graph here. So, in this slide what we have is the cruise control set at 68 miles an hour, right?

A. That's correct.

Q. And you start out, and Mr. Louden is he in the vehicle? This is one of those dynamometer tests, right?

A My understanding he's in the vehicle, that's correct.

Q. And he's in the vehicle. Does he have a computer or switch box or something in there with him?

A. He's able to log things that are happening on one computer, and I don't know if it's a separate computer that Toyota Tech Stream, that he was able to inject the fault.

Q. And he actually also had to switch that he could switch on the brake rather than having to apply with his foot, didn't he?

A. I think he was applying with his foot.

Q. We'll look into that later. But so we start out, and the speed is below 68 miles an hour. And unfortunately, these numbers are in kilometers, aren't they?
A. The left access, vertical access is in kilometers per hour.
Q. So you've got to convert, so like 100 kilometers per hour is roughly 62 miles an hour, correct?
A. I think it's almost exactly 60, sir.
Q. So the jury can kind of figure this out. 50 miles an hour would -- be 50 kilometers an hour would be about 30 miles an hour, right?
A. 50 and 30, that's right. 80 and 50 is one that I know.
Q. So for 80 kilometers an hour is like 50 miles an hour, right?
A. That's correct.
Q. So if we're below 68, and you can see down here this red line is how open the throttle is, correct?
A. That's correct.
Q. And you also used this graph to represent percentage of throttle opening, don't you?
A. That's not a 20 percent on the left there.
Q. What is it?
A. I don't recall as I sit here.

Q. As a matter of fact, this acceleration rate is pretty gradual, isn't it?
A. Yes.
Q. If you go from 50 miles an hour to 90 miles an hour it takes you like 30 seconds to do that?
A. That's correct.
Q. I mean, I know this is a four cylinder Camry, but it will accelerator faster than that, won't it?
A. Cruise control resume function will not accelerate as fast as some other functions.
Q. That's exactly where I was going, because now you're at cruise control, so any of us that have a cruise control car now when we press that button to resume, the acceleration rate is a lot more gradual than flooring it, right?
A. That's correct.
Q. So that means the throttle isn't open as wide it a possibly could be. Do you know if this 20 percent throttle?
A. I don't believe that access applies to that. I don't know the percentage of it.
Q. But in any event, at this point when you kill the task, because the speed is still below the requested speed, it just keeps applying the throttle. It's trying to accelerate up to that set speed, right?

A. That's right. So what should you have happened is that when the blue line crossed the set speed at 68 miles an hour right there.
Q. It should flatten out?
A. It should flatten out at that point. And software that does that, the cruise control is in the task X.
Q. Mr. Louden killed that task. And we've also seen the throttle memory increases it. It was killed there, it just stayed flat?
A. Right, because when we were injection faults, we were only flipping one bit. That's not how real memory corruptions happen in software systems. Real memory corruptions bounce around, ricochet and cause multiple damages. For example, if one bit flips and that's in the pointer address, then when that pointer is used, then you go to the wrong place and you write something else. So they can be cumulative.

When we did our fault injection testing it was like taking a rifle shot, just flipping one bit or what we were interested in flipping and nothing else.
Q. And you've -- we've already agreed that none of the testing that you've done or Mr. Arora's done, have we ever had this memory corruption to cause the throttle to open greater, have we?
A. Nobody has tested that as far as I'm aware.

Q. Nobody has done that. Not only not tested, we have no documentation that that's ever happened, do we?
A. No. But unfortunately for Toyota, you can't disprove it. It's trying to prove a negative. It's trying to prove, for example, that there are no aliens in the universe. You can only say that you have not found any on Mars, but you can't say definitively that they don't exist.
And so the same is true here that just because today as we sit here I don't have this test data, it would only take one, and it could depend on timing and other factors that we can't control right now in our testing, but it would only take one experiment to see that throttle open wide from the second memory corruption to prove our point, whereas a million experiments that didn't open would not prove what Toyota would like you to believe.
Q. And we don't have that experiment, that experiment has never been done?
A. That experiment as far as I understand hasn't been done by either side to date.
Q. As soon as this brake switch was transitioned, this speed -- the engine throttle closed, and then because it was still, the task was still there, the engine stalled, correct?
A. When the brake switch was transitioned for more than

2/10ths of a second, that's what happened.
Q. Unlike Ms. Bookout's crash, right?
A. Ms. Bookout's engine did not stall, as I understand,
Q. No, on this stack analysis, which is 25 -- finger over that -- that's the stack where is the throttle control is, correct?
A. The throttle control, that very complicated function in task X executes on that stack, is that correct.
Q. Now, if you look at slide 29. This may be -- maybe just -- NASA didn't call any of the MISRA coding issues violations, they called them deviations, didn't they?
A. I don't remember what word they used, sir, they're violations or deviations both.
Q. If you look -- you're talking about MISRA -- talking about these violations in this document here, but if we looked at the NASA record appendix A, page 29, Mr. Doyle, what they really said was it was deviations up at the top photograph. They never refer to these as violations, did they? Deviations?
A. I'm not going to say never, because I haven't reviewed the report for that, but there it says deviations.
Q. And this is the section they are talking about the MISRA rule check, right?
A. That is a section.

Q.   Now, on slide 32 I've got to take issue with you, Mr. Barr. You quoted part of what Mr. Ishii said. Right there. He had a much longer answer than the stuff that we got right there, didn't he?
A.   I don't recall. I think that's the salient point.
Q.   Did you put this slide presentation together?
A.   I did, and I took that quote from my report.
Q.   Can we me see Mr. Ishii, page 90. And we heard Mr. Ishii's testimony a couple of days ago. This is page 89. I'm going to focus you on line 13 -- you don't even have the question there, we just have an answer, don't we?
A.   That's correct.
Q.   This is again, you quoting from page 91 here, but what he actually had to say starts up here. The question was, Is it your position, Toyota, that there are no major bugs in the engine control module software from 2008 Camry engine?" And his answer was "One thing I can definitely state here is that with respect to the engine control software bug or problem that leads to a UA, that's unintended acceleration, and I will define unintended acceleration as an unintended engine rpm increase unintentioned by the system designer. That sort of bug is not there. That I can state definitively, since the term major bugs is not -- as used in your question was undefined and vague. I defined UA and

provided that definition to you in answer to your question.

And then here's the answer you quoted right there. When it comes to software, you had some ellipsis in there, but the next question was -- and this is a really material question -- are there any bugs in the software that can cause the main CPU to open the throttle inappropriately? And you left this answer out. I will repeat what I said before and that is with respect to software bugs that could result in unintended acceleration and my definition would be an engine rpm increase contrary to what is designed by the system designer, that kind of bug does not exist.

That's what he said about that, didn't he?

A. That's what he said, but it doesn't sound like science to me, but that's what he said.

Q. Now, almost done here. You showed us this fish bone diagram. Mr. Doyle, I'm going to need to have the NASA report in just a moment. Appendix B. You showed us this fish bone diagram up there. Now, you understand that this diagram was for the software error only, right?

A. I do.

Q. And you modified this chart for your purposes in this case, did you not?

A. I modified that chart to make it clearer what was upstream.

Q. You also modified it to put in the words "UA" or the letters "UA" up in the corner, because that's not on the chart in the NASA report, is it?

A. That's right. There it says a global concern and it continues on to another chart.

Q. You didn't tell the jury yesterday that you modified this chart to change it from global concern to UA, did you?

A. You're right, I didn't. I was trying to make the chart's ultimate purpose, which is a UA clear, which if you continue the global concern up in their fish bone which gets more complicated, that's ultimately what they are looking for.

Q. And so if we go to the -- after this chart there are two more pages which list the disposition detection -- not that one. That's where it says global concern. Now the next page, Mr. Doyle. There are two pages that go through for each of those concerns -- if you give me the last two columns. And here it talks about what will happen and talks about how it's detected and how it's mitigated, each one of those software errors, right?

A. It talks about how NASA believed it was mitigated at the time they wrote the report. And I mentioned those pages when I put the chart up, my slide says page 36 to

39, whatever the numbers are.
Q. But you didn't show that they were mitigated, these software errors had a mitigation strategy for each one of them?
A. I think I clearly stated that NASA misunderstood that some of them didn't exist. And that's what I would have explained if I went to this level of detail.
Q. Now, let's go to slide 43.
A I'll give an example just to continue my answer. One of those --
Q. I don't think there is a question pending.
THE COURT: You can ask him on redirect.
Q. (BY MR. BIBB) And at slide 43 you again quoted Mr. Ishii, but if we look at what he actually said on page 37, what we left out, what he said was that, of course, Toyota would conduct its test to see if the chips that are delivered would have the requisite functionality and performance. But admittedly they didn't do a design review of somebody else's source code, correct?
A. They didn't do a design review, and I think the point stands for itself.
Q. It was a source code provided to Toyota by suppliers, right?
A. Right. So here he's referring to the ESP-B2 monitor chip and he's saying they didn't review the source code

for this very important chip, including through the date when they were telling Congress and the world that there couldn't be possibly be a software error, though they already knew that it was spaghetti code.

Q. They knew -- they tested it and they knew it performed and functioned as it was intended to, right?

A. In the testing they performed of ESP-B2 chip, which is obviously not as much testing as a vehicle fleet of millions of vehicles driving around.

I did a calculation that I included in my report. Toyota told NHTSA and Congress about the amount of testing they had performed on this series of Camry's. And if I remember the numbers correctly, it was something like 400,000 hours of testing. That sounds like a lot of testing, sounds very impressive, but if you simply do a little math, the first, I think it was -- I'd like to refer to my report to get the exact numbers -- but I think I calculated the first 3,000 people to buy that car in their two weeks of ownership would conduct more than 100,000 hours -- morn than double, more a million hours of testing.

And so, when Toyota says we tested our car, they tested a couple of cars that came off the factory line first, under certain conditions, and then they started selling the cars and there were now 3,000 of them or

400,000 of them. I think they sold 400,000 2005 Camrys ultimately, and that is a much larger universe of testing. People are driving it in different weather conditions, they're driving with more miles, there are manufacturing variances between vehicles, electrical, mechanical, and all of those experiments are taking place in the real world.

Q. Let's look at slide 53. This is another one of Mr. Louden's tests you talked about, right?

A. That's correct.

Q. Now we saw the earlier one we looked at, that was actually a 2008 Camry being tested, wasn't it?

A. It was.

Q. Is this one a 2008 or a 2005?

A. This is a 2005 Camry, just like the Bookout vehicle.

Q. And here the thing I want to focus on is this last row down here. And there Toyota has a brake switch that's connected to the brake pedal, and when you tap the brake pedal, that switch, it's a mechanical switch, electro-mechanical switch, isn't it? When you tap the brake pedal, one switch goes from open to close, and the other part of the switch goes closed to open, right?

A. That's right. The first one is called STP, that's the primary stop brake switch. That's the one that lights the lights on the back when you see someone's brake lights come on. And there's a secondary switch called ST1 minus.

Q. If we look at this last column down here, you can see there's the one line, and I would have to go and look at this, but suffice one of them is ST1 and one of them is STP, right?

A That's correct.

Q. Admittedly I"m color blind, but I think that's a blue line down there, and I can't really tell what color this is, it's either red or green.

A. It's both red and green because it's got data from inside the computer and data from the actual brake pedal.

Q. You made me feel a lot better with that answer. But when you press the brake pedal right over here, this blue line should have gone up here?

A. That's right. It's not relevant to this particular test. Mr. Louden conducted a number of -- in the Saint John case there was early on some indications that there might have been a problem with the secondary brake switch, and so he conducted these tests with a number of different combinations of that switch not working and working. This set of data -- that doesn't change the outcome here. It's not related to the monitor CPU and no other part of the software that's relevant to this really.

Q. And it's not relevant to Ms. Bookout's case because we know her brake switch worked?
A. Well, this graph is absolutely relevant. That doesn't change the relevance to Ms. Bookout's case.
Q. The other thing we notice up here is when this -- when that task is killed here, this is the throttle opening, right? This is degrees now, we can actually look at the throttle opening, correct?
A. That's correct.
Q. And so when this task is killed and she's doping 45 miles an hour, or Mr. Louden is moving 45 miles an hour, the throttle is open and at about, what, 14 degrees, correct?
A. Yes. If you look at this curve, you'll see that it's up there around 15 degrees, I guess, close to the task death killing.
Q. Here's 10 and there's 20. And task death occurs at that dotted line, and this thing -- it settles in, looks like roughly 13 or 14 degrees?
A. I would agree with that. It looks to me what Mr. Louden did was he had his foot more on the accelerator, before he killed the task, he got up to his target speed about 45 miles and hour, and then as he let off the accelerator there, before the task death, it dropped down to the 13 or 14 you're talking about.

Q. If it's at 14 degrees of throttle opening, you're never going to lose vacuum assist if you pump the brakes in that situation, are you?
A. I've not heard a number as low as 13 or 14 degrees.
Q. So the answer is that you'd always replenish the vacuum boost if you had this scenario here of 45 miles hour, right?
A. As an engineer I've learned to be cautious about absolutes, so I won't say always, but I think it's unlikely.
Q. Tell you what, from what you know though, you still have vacuum assist, right, based on everything you've heard so as far, correct?
A. I think that's right.
Q. Now, all I want to do, Mr. Barr, here to finish up. You talked to us a lot at the end of the day yesterday about some other people's wrecks that you told the jury you thought were similar to Ms. Bookout's, correct?
A. Yes, they informed my analysis.
Q. And the first one was a Mr. Beresford Hill, do you remember that name?
A. I remember the name.
Q. And Mr. Beresford Hill, did you read his deposition?
A. I believe so, if I had his deposition and cited to it there, I read it.

Q. Page four of Mr. Barrister Hill's deposition, because I'd like to look at how he describes -- this is his deposition beginning at line 2 and just go down to line 11. And he said, "As I pressed my foot on the accelerator it was as if the car took on a life of its own."

So now, Mr. Hill stepped on the gas and the car took off. Is that the way the car is supposed to work?
A. I think Mr. Barrister Hill was describing a situation where the accelerator pedal press was a lot less than the car's acceleration.
Q. Let's assume that this could be caused because he pressed his foot a little harder than he expected on the gas pedal, couldn't he?
A. I don't know why we would assume that, sir.
Q. But he instinctively put his left foot on the brake, and when he did that, it would have brake echo check, wouldn't he?
A. No, not necessarily.
Q. Just every time you've ever tested it, it worked, right?
A. We don't know that this particular UA, which I believe was caused by software malfunction, was caused by task X death specifically. It's my understanding it was a software malfunction. I don't know whether it was task

X or task X in combination with other things. So I don't know whether that failsafe should have acted in that particular situation.

I know that even if it was task death X, it won't be 100 percent reliable, that brake echo.

Q. And you're telling this jury that this one, software failure, stepped on the gas and the car took off, right?

A. It's my view that that is a description of a software malfunction.

Q. Now, how about, one of the other ones was a Chory, C-H-O-R-Y, did you review the event data recorder for that crash?

A. I don't believe I had the actual event data recorder, but I'm familiar that Toyota reviewed it.

Q. Have you seen that data readout?

A I don't think I've seen that data readout.

Q. You've seen these before, haven't you?

A. Yes.

Q. You understand how to read them. If we look at this paragraph right there, that box shows that the brake was never applied, right?

A. That's what it shows. I've written a separate chapter about how these pre-crash recorders have their own defects. In fact, Mr. Arora in his September 17th report last year, he actually demonstrated for us that the car he was pressing the brake on, the recorded black box data sequence said he didn't press the brake. And that's cited in my chapter on the pre-crash EDR, which is not really directly relevant to this case because the 2005 Camry wasn't equipped with that, but the point being that in this later Camry that had it, this is not something we can rely on to disprove a software malfunction. In fact, with the UA occurs and task K is dead, the pre-crash EDR will be wrong about the brake signal specifically. That's what Mr. Arora's data showed.

Q. And you know that NHTSA disagrees with you on that?

A. No. The analysis this NHTSA did was a very different analysis. What NHTSA did was to evaluate that if data was stored in the black box, that it was reliably read out the same way that it was in the box. NHTSA didn't evaluate -- they did evaluate in one bumper crash that they got the right data. But that didn't prove -- we read the code and said -- and we even got the pre-crash EDR code and we saw that it could be confused also by task X death, specifically about the brake pedal.

So NHTSA always assumed that these black boxes were reliable, but they're not. And that's been demonstrated by Toyota's own expert.

Q. But they dud a study of those and no matter how you

want to characterize it, they validated the validity of these EDR readouts, didn't they?
A. As I explained, they validated that the data could be read properly by either a tool from Bosch or a tool from Toyota. They didn't validate properly scientifically like we did that this could be wrong.
Q. They did some testing with vehicles to confirm with accelerometers and their data acquisition that the data that was being recorded in the EDR was the same data they were getting with their external recording devices, correct? You are aware of that study?
A. Again, sir, it doesn't matter how many tests showed that the EDR worked. We have one test that was conducted by Toyota's own expert that proves it can be wrong. And that is sufficient to prove there are aliens in the universe. That is sufficient to prove that the EDR is not reliable. So one test like that disproves this view that Toyota would have you have that this is reliable.
Q. I'll tell you what, Mr. Barr, I'm going to do one more person's incident here. Mr. Heinrich or H-e-i-n-r-i-c-h?
A. Is that the gentleman with three incidents?
Q. That's the gentleman with three incidents. Let's talk about a couple of the incidents. You ruled out floor mat interference in that case?

A. Yes. Well, I should say I didn't do a root cause analysis but my understanding was it was not likely floor mat.
Q. You know though he had a set of all weather rubber mats placed on top of his carpeted mat in the vehicle?
A. That's not true the third incident.
Q. I'm going to get to the third one, but the first incident and the second incident he had an all weather rubber floor mat on top of his carpet mats, is that right?
A. I believe that's accurate.
Q. And that was exactly a concern that Toyota had, right?
A. Just because they are there, doesn't mean that's what happened.
Q. So if we can then go to page 42. Let's look at his third incident, and this is the one where he was waiting for the train, do you remember that?
A. I do.
Q. And if we could look starting at line 9 how he described it. And he said it was a normal day, crossing gates came down, waiting for the train, put it in park because they are usually freight trains and they are long so I just waited my time. When it came to time to go, the crossing gates went up, I put the car in gear, gave

it a little bit of gas because of school zone just a quarter of mile and the car took off.

And he said, it was gradual, I just had no control over it. Again, this is a gentleman who's putting his foot on the gas to accelerate away from a railroad grade, isn't he?

A. Yes, and his car is giving him an unintended amount of acceleration when he does.

Q. And but again, maybe Mr. Heinrich just pressed on the gas pedal harder than he expected, he got more result than he wanted. That's another way of looking at that accident, isn't it?

A. That's a possibility.

MR. BIBB: May I have a moment, Your Honor?

THE COURT: Yes.

MR. BIBB: Was an a little longer than 15 minutes, but thank you very much, Mr. Barr for coming. Do you want to take our morning break?

THE COURT: Mr. Baker, how much do you have? More than a few minutes?

MR. BAKER: Yes, ma'am.

THE COURT: We will take our morning break then. Ladies and gentlemen, it's 10: 30, we'll be in recess for 15 minutes. I'll remind you, do not discuss the case and form no opinions about the case. All rise argument.

THE COURT: All right.

(THE FOLLOWING PROCEEDINGS WERE HAD WITHIN THE HEARING OF THE JURY AS FOLLOWS:)

THE COURT: Ladies and gentlemen, I apologize for the delay. We had a deposition that we're going to play as soon as Mr. Barr's testimony is completed and I had to make some evidentiary rulings. I wanted to let you know after we completed Mr. Barr's testimony, the next witness will be video testimony again. We are going to start that and go until about 12:30 and then we will break at 12:30. I have a matter at 1:00 in another case, believe it or not, I still have other cases that I need to take up, so I have a 1:00 but it shouldn't take over 15 minutes, so we will break from 12:30 to 1:30 today for lunch. Thank you very much.

You may proceed.

REDIRECT EXAMINATION

BY MR. BAKER:

Q. Let me kind of start at the back where you all ended and I want to talk a little bit about the EDR you all had a great field discussion about EDRs, when it occurs and whether NASA looked at it with respect to a particular OSI or other similar incidents. Do you remember that discussion?

A. Yes.
Q. As I understood, is it true that the EDR event data recorder in 2005 Camry would not record anything that happened before a crash?
A. That's right. And the black box in the air bag computer in the 2005 Camry simply recorded that there was a crash and information about the crash. And it could record I believe up to three total crashes. And then the later models had a black box that recorded not only if there was a crash, but also the five -- sample data in the five seconds before the crash.
Q. Our car doesn't have that?
A. That's right. This car in this case doesn't have any pre-crash data.
Q. But the EDR that is in our vehicle was downloaded?
A. That's correct.
Q. And you've seen the information downloaded from our EDR?
A. I have.
Q. Did it record anything?
A. No, despite the crash, there was no data recorded, no crash was recorded in the air bag control unit.
Q. Even though there is a 30 mile an hour impact at the end of this sequence, nothing recorded?
MR. BIBB: Objection, no foundation.

THE COURT: Overruled. You may answer.
THE WITNESS: That's right, despite the crash there was no crash recorded in the air bag computer.
Q. (BY MR. BAKER) You were also asked about the OSI in terms of your root cause analysis in this case. Do you recall those questions?
A. Yes.
Q. You looked at those OSIs for particular things and used them as part of your analysis, do you remember that?
A. I do.
Q. In terms of doing a root cause analysis and trying to determine what causes these unintended acceleration events, would it be reasonable to overlook a 400 percent increase in UA events starting in 2004 in the Camry?
A. No, it would not be.
Q. You were also asked some questions about the NASA report. Be very brief on this. Yesterday you discussed several aspects as we've already seen. Is it true that NASA made its conclusions based on some inaccurate information given to them by Toyota?
A. That's correct.
Q. For example, you mentioned they had told NASA there was EDAC on the 2005 Camry?
A. That's correct. On that basis NASA said, well, it can't be hardware bit flip because there's EDAC. But

since there's no EDAC, then there can be a hardware bit flip, and NASA was concerned about hardware bit flips, rightly so.

Q. Is it one of these bit flips that we talked about that we can have the throttle angle variable become corrupt?

A. That's one way it could happen, that's right.

Q. Is it a corrupted throttle angle variable that could make the throttle go anywhere?

A. That's right, anywhere between, up to [redacted] degrees or 100 percent.

Q. So in this case, if Ms. Bookout has her foot on the accelerator and it's at whatever, I think one of the numbers we talked about was 15 degrees opening, if you have this memory corruption on the throttle variable angle, could it send it anywhere?

A. That's correct.

Q. And you were criticized for the quote you put up of what Mr. Ishii said. Mr. Ishii said there is absolutely no bugs in the software.

MR. BIBB: Objection, misleading.

Q. (BY MR. BAKER) Ultimately what he said was on the power train software that he was discussing was the part put up by Mr. Bibb, he said there was no bugs.

MR. BIBB: Objection, misstates testimony, Your

Honor.

THE COURT: Overruled.

THE WITNESS: I understand Mr. Ishii's testimony in the portion Mr. Bibb cited to be not that there were no bugs, but that there were no bugs, he thought or believed that there were no bugs of a specific type.

Q.   (BY MR. BAKER) And my only point asking that question is, there is bugs in every software?

A.   Any reasonably complex software has bugs. This software certainly has bugs.

Q.   And you were also asked about some examples up here in the brake echo, and some of the tests that you ran that showed that if you had your foot on the brake and this task death occurred, or you had concluded the variable angle malfunction of the throttle control, that you have to take your foot back off the brake in order to have --

A.   That's correct. And even then it may not happen.

Q.   But if it does come into effect, as I understand your testimony, it stalls the vehicle?

A.   Three second afterwards approximately.

Q.   We know Ms. Bookout's vehicle did not stall, correct?

A.   That's correct.

Q. So under the scenario that you've described for the jury, would that mean that she never transitioned the brake switch?
A. That's correct.
MR. BAKER: That's all I have.
THE COURT: Mr. Barr, you may step down.
THE WITNESS: Thank you, Your Honor.
THE COURT: Mr. Baker, you may call your next witness.
MR. BAKER: We have a few exhibits with Mr. Barr, we'll take up at the break.
THE COURT: Okay.
MR. BAKER: We're going to call by videotape Mary Pries-Morrison.
THE COURT: Okay thank you.
(Playing video.)
THE COURT: ladies and gentlemen, there's still about 25 or 30 minutes left on this, because of my schedule in needing to take care of another matter we're going to go ahead and break for lunch at this point in time. We'll be in recess until 1:30, again remind you don't discuss the case, don't form any opinions about the case during the break. If you did not check in at the jury assembly room this morning, please do so during the lunch break.