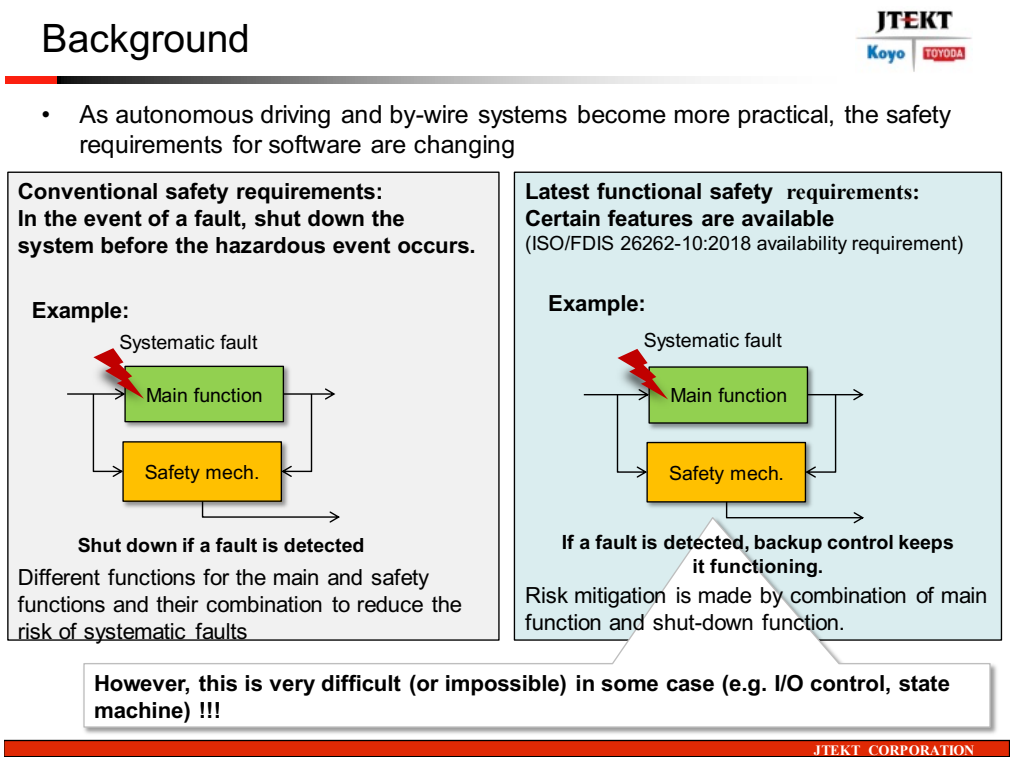
2020.09.18

Toyota and NVIDIA focus on zero defects software development mathematically

Hidemasa Kimura, Nikkei Xtech

The development method of zero defects (bugs) in in-vehicle software is attracting a lot of attention. This is because there is an increasing need to improve the safety and security of software against the background of autonomous driving. JTEKT, a major Toyota-affiliated electric power steering system (EPS) manufacturer, is in the process of introducing this method to mass production, and NVIDIA in the United States, which is involved in the production of in-vehicle system-on-chip (SoC) for autonomous driving, is also considering a similar method.

JTEKT and NVIDIA are considering what they call "theorem proving" (a type of formal method) to mathematically prove that the software has no defects. With the practical application of autonomous driving and steer-by-wire (SBW), the safety requirements for in-vehicle software are increasing rapidly. In the past, when a failure occurred in a system's main function, it was simply a matter of shutting down the system using safety mechanisms. In contrast, with autonomous driving and SBW, it is rather dangerous to shut down the system, so it is necessary to continue the minimum function (specified function) by the safety mechanism.



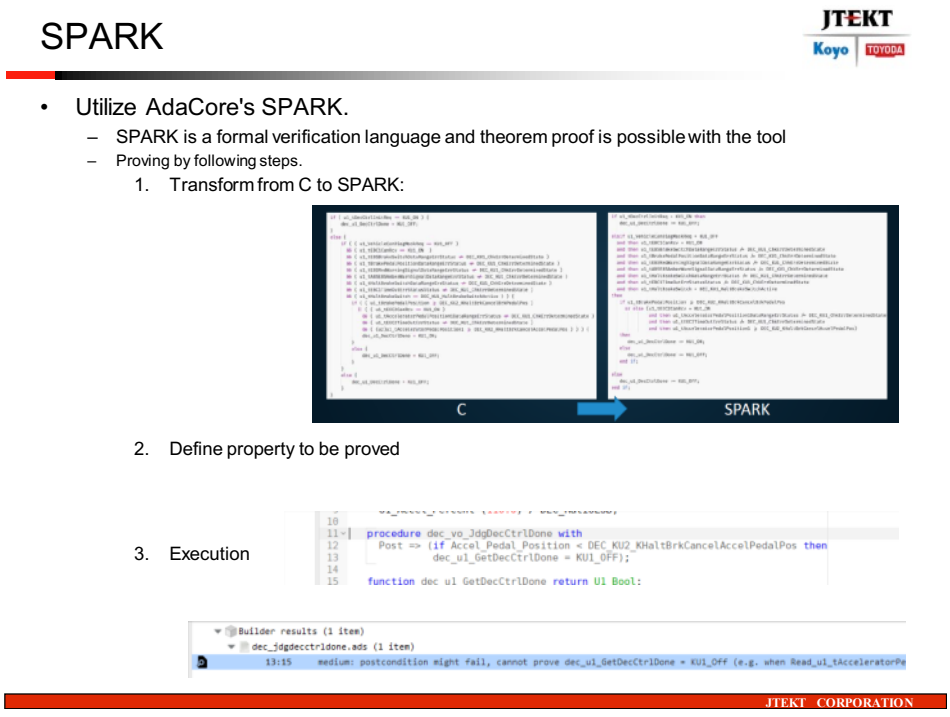
Safety requirements change against the background of autonomous driving and SBW (Source: JTEKT)

However, the implementation of such a safety mechanism is "sometimes difficult," said Shinya Yoneki, Advanced System Development Department, Steering Business Division, JTEKT. Normally, the safety mechanism is implemented with software that is different from the main function. With EPS, however, both the main function and the safety mechanism must return the same output to the same input, making it difficult to change the implementation of the software. Therefore, JTEKT was looking for a method of proving that the main function alone would not cause any problems even without the safety mechanism.

Normally, the safety of in-vehicle software is verified during the testing process. However, it is difficult to prove that there are no defects. There are an infinite number of test conditions, and no matter how comprehensive the testing process is, it is impossible to say that there are no defects. Therefore, it is up to the customer to ultimately decide whether the product is safe or not. In contrast, theorem proving can mathematically prove that the software has zero defects. "It could be used as a basis for a decision that everyone can agree on," he said.

Select the SPARK language

The tool is SPARK Pro, provided by AdaCore (Press Release). Since we are currently in the implementation stage, we are converting the C language source code into the theorem proving language, SPARK, and verifying it. After confirming that there are no mistakes, we will convert the code to C source code again. In the future, we plan to use SPARK from the beginning when we develop new software and convert it to C after theorem proving.



Utilize SPARK (Source: JTEKT)

The usage is "similar to code review," he said. However, the language is SPARK, not C. In addition to the program, SPARK allows you to describe the attributes you want to verify (such as input and output ranges in the specifications).

When you run the verification tool, you can immediately see whether the program is wrong, or the specification is wrong. "It's like replacing code review with a machine," he said.

For example, if a program is found to be inconsistent with the specifications, even though they know from past experience that the program is correct, "we can find out that there was an omission in the specifications," he said. Such defects could be found in the testing process as well, but "if the software engineer could check the program by himself while writing it, it would result in faster work," he said.

In the products that JTEKT had worked on in the past, the majority of defects found during the testing process were caused by errors in the specifications and implementation (program) of the software itself. It is a great advantage to be able to reduce errors in specifications and implementation, the two most common causes of defects, at the development stage.

Not all-purpose

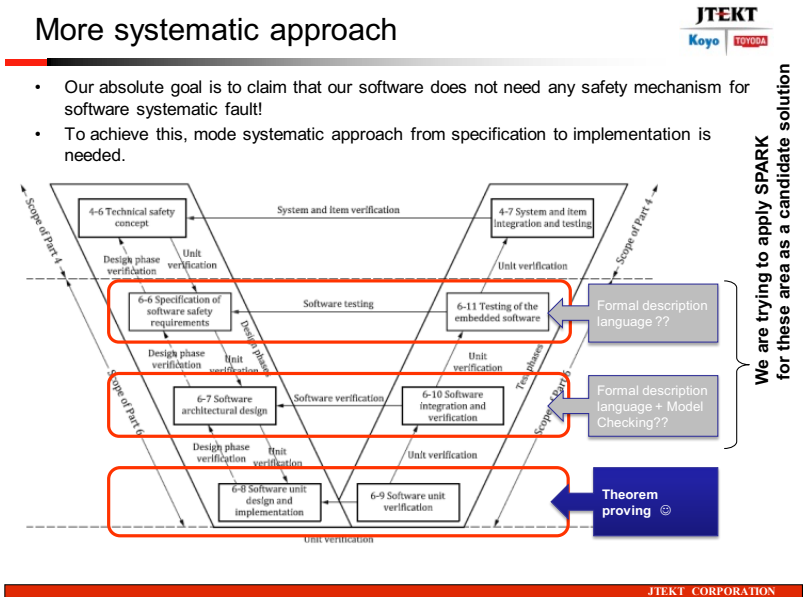
However, "this method is not all-purpose," he said. Although we have confirmed that the theorem proving can be used as an alternative to unit testing to some extent, it is necessary to weigh whether the defects found in SPARK can be found in existing tools. "We are currently working on that," he said.

Mr. Yoneki Shinya, Advanced System Development at JTEKT
(Source: JTEKT)

And the biggest challenge is that the programs to which the theorem proving can be used are limited. "It's difficult to apply unless the program has a high affinity with Hoare logic, which is the basis of the theorem proving," he said. For example, "Programs that can clearly describe the upper and lower limits of the input/output range are suitable," he said. On the other hand, "In the case of programs such as feedback control, where the output changes depending on the previous state, the

input/output range is difficult to describe and difficult to apply," he said. It is necessary to determine the compatibility of each program.

JTEKT intend to use it first for end software components that would communicate directly with the hardware and for basic control programs. These software programs do not have very complex controls, but if they malfunction, there are significant safety implications.



Applied to the bottom of the V process currently (Source: JTEKT)

The size of the software that applies theorem proving (number of lines of code) is small, less than a few percent of the EPS system as a whole. Nevertheless, it is important to be able to say that the part is zero defects. They want to increase the overall safety of the system by first proving that the small end software components are zero defects and then building on top of that the functions they want to achieve.

NVIDIA is also looking at AdaCore's SPARK language and Ada language, which is the basis for SPARK (Press Release). NVIDIA is considering rewriting some security-critical firmware from the C language to Ada and SPARK. As Autonomous driving cars are connected to the outside world through communication functions, vulnerabilities in the in-vehicle SoC firmware make it a target for cyber attacks. If the vulnerabilities can be eliminated by using theorem proving, it could increase security.

Published in Nikkei xTECH, September 18, 2020
https://xtech.nikkei.com/atcl/nxt/column/18/00001/04594/
Translated by AdaCore with permission of the rights holders.