With delays and cost increases of aircraft test programs increasingly blamed on software problems, what technology and tactics are available to keep software testing under control?

Software everywhere

1 // As flight decks have become digital, the testing requirements for the software that runs them has increased

During a modern airplane's flight, hundreds of millions of lines of code will run. Software is present almost everywhere in the aircraft, from mundane components like galley equipment to highly critical ones such as flight control systems.

Each line of code has to be checked for faults. Each software unit has to be tested to see how it integrates with other units, and then tested at a higher systems level. It's a laborious process. Increasingly, cost overruns and delays in airplane development are linked to software testing. But it's vital to get software right.

"If software fails, you can get awful incidents like the flight 447 Air France A330 crash over the Atlantic in 2009, which killed everyone on board on the way from Brazil to Paris. It was caused by one hardware failure, but the root of the incident was the software's inability to recognize and report the hardware fault," says Dylan Llewellyn, international sales manager at software company QA Systems.

CRITICAL PLANNING

The amount of software on aircraft is set to increase. For example, as engines evolve they are becoming more software dependent. FADEC (full authority digital engine control) engines are entirely controlled by software. Among commercial aircraft, the A380 is well known for having large amounts of code. But nothing compares to the amount of code within the F-35 jet fighter, which has been beset by long delays because of the mind-boggling complexity of its software development.

To avoid disasters such as Flight 447, and to reduce costs, software testing should be done as early as possible in a development program. Massimo Bombino, an expert in avionics software and regional manager of Southeast Europe for Vector Software, recommends planning software testing from "day zero" of a development program and forming small agile teams of software developers. These teams should plan all the software unit testing and integration testing from day zero to minimize the risk of nasty surprises and last-minute regression testing, which is conducted when code goes wrong.

1,000
Software faults identified on the F-35 after more than 25 years of development in January 2018

"Regression testing is a nightmare and a big issue for the whole software industry. It's especially a challenge with safety-critical aviation software. Everything can be running perfectly, then at the last hurdle you introduce a new element and it fails. It's very tricky and time-consuming to solve unless you have advanced technology," says Bombino.

There are two ways to prevent regression testing, believes Bombino. The first is to test rigorously from the start. But if there are too many problems and it is too late, it is better to conduct change-based testing. Vector Software has technology that enables a subset of tests to be run according to just the code changes, thereby improving the efficiency of testing.

INDEPENDENT VERIFICATION

According to DO-178C, the primary document by which the FAA, EASA and Transport Canada have agreed to

F-35 DELAYS CAUSED BY UNREALISTIC GOALS AND C++

Tucker Taft, director of language research at AdaCore, says that the F-35 Joint Strike Fighter has become too complex because every one of the high number of stakeholders in the project is demanding ambitious requirements. "All the amazing technical qualities can be in conflict and they keep changing their minds."

The F-35 has taken more than two decades to develop and has been plagued by huge time and cost overruns. The lifetime costs stand at an estimated US$1.5tn, partly because of the enormous price tag for software development and testing. As recently as January 2018 the Pentagon was forced to admit that there are still close to 1,000 software faults on the jet, but won't say precisely what they are.

"The software development keeps getting the blame, but the whole project management can be seen as at fault," Taft says. "The program has goals that are almost impossible to reach. The lesson is to put a stake in the ground and say we will build it this way and stick to it."

Over-ambition may bear some of the blame, but it is undeniable that software development has also contributed. Taft believes that writing the software in C++ has also caused the overruns. "When you factor in the cost of debugging later on, it's worth doing a little more training to use a language that's less problematic, such as Ada," he says.

2 // The F-35B hovering

3 // The F-35's sensors and software detect and process terrain and threats

2

3

"The secret is to make testing a fully fledged part of the process"

AUTOMATED ADVANTAGE

One of the reasons aircraft testing is so expensive is the high cost of testing, but automating the process can save time and money, according to Dylan Llewellyn (pictured), international sales manager for software testing company QA Systems.

Llewellyn says his company's Cantata tool can test in a month software that would otherwise take several months to test. But the industry is resistant to using automated tools. "Icebergs don't move fast and outsourcing companies don't always inform the manufacturers that these tools can cut expenses substantially.

"The message is always that it will take a team of 60 a year and will cost US$1m, rather than saying you can test it quicker by licensing an automatic tool for several months," he says.

Cantata is qualified to test software to DO-178C's stipulations. It works by allowing testers to put code into the tool and telling it the standard it has to be tested to. Cantata will then run the code. Users can see the script and modify it in real time. At the end of the process, the tool tells the testers if the code has failed. It signals the reasons for failure and details the lines of code at fault. "You can test the smallest possible units of code, but it also works for integration testing when you put lots of smaller units together," says Llewellyn.

8 MILLION
Lines of code on an F-35

approve all commercial software-based aerospace systems, software must be tested by independent parties. The software process is therefore mostly outsourced to third parties, although some aircraft OEMs use separate in-house teams. These independent parties perform verification and validation testing to establish that all the bugs in the code have been removed.

Tucker Taft, a computer scientist and director of language research at software developer AdaCore, says that another problem is that if testers find lots of faults at the verification and validation stage, the errors can be hard to trace back to the original software developers.

"The old-fashioned way of putting software together until you think it's worth testing doesn't work with hundreds of millions of lines of code," says Taft. "Smart companies today frontload their testing processes by finding faults during design and not waiting until integration testing.

"The most modern, agile companies use test-driven development, where you don't start writing code until you've written the pre-imposed conditions that it must pass. The secret is to make testing a fully fledged part of the process."

When Taft tests software for faults, he carries out static analysis, in which a tester tries to mathematically prove that errors will manifest without running the code. Experience has taught him that certain types of software are hard to examine using the conventional tests written by programmers and that it can be more straightforward to obtain formal proof of software unit's safety using static analysis. However, the exact opposite can be the case with some other types of software units.

"For part of the system, you focus on mathematical proofs and for other parts you use a more dynamic testing strategy," Taft says. "Customers often like a combined strategy because they get the confidence of mathematical proofs."

As part of more formal testing for certification, Taft prefers to define pre- and post-conditions for software.

4 // The cost of developing the Boeing 777 is reported to be US$800m

4

Pre-conditions imply that before the software sends data to a component, the code has to have certain properties. Unless the software is in an appropriate state, it will not be allowed to send the data. Testing can verify that the pre-conditions are satisfied. Later, the post-conditions state what predetermined data the tester should get back from the software.

"When you use pre- and post-conditions, you can do a lot more formal verification and determine whether the software all fits together," Taft says. "It makes it easier to do integration testing, which has always been one of the greatest challenges, as each contractor builds and tests in isolation, but when you put it together something inevitably goes wrong."

120 MILLION
Lines of code on an A380

MODELS AND FUZZ

Developments in technology are helping to reduce the software testing workload. As well as automated testing, the use of mathematical modeling for software testing has evolved rapidly in recent years. It can now simulate with great precision what an airplane will do when the software is installed.

Large manufacturers, such as Boeing and Airbus, have embraced the model-based approach to testing software and have created detailed, accurate models.

"When you use pre- and post-conditions, you can do a lot more verification"

These models can be shared with subcontractors so that they can carry out hardware-in-the-loop tests, where hardware is tested within a simulation of an aircraft's software systems. "These simulations require a lot of horsepower, so it's not trivial to create them, but they're worth their weight in gold," Taft says.

Another trend is 'fuzz testing', which involves blasting the software with large amounts of random data, called fuzz, to try to break it.

Taft compares fuzz testing to the tricks white-hat hackers use to expose weaknesses in corporate security systems. "Fuzz testing hasn't yet reached the practical level, but there's a lot of research in academic circles and I think it will play a big part in future testing. You never know when a hardware failure might generate random data, or it could be a hacker trying to break in," he says.

Despite these technical advances, most aerospace companies find software testing onerous. QA System's Dylan Llewellyn was part of a team testing the power distribution software on the 777X. The code was written in blocks and tested to ensure that when it was added to the mix it didn't have a negative effect on what was already there. "If you have bad code in legacy code it causes havoc," he says. "That's why when they went from the Boeing 777-300 to the 777X, they didn't use any of the legacy code."

PART OF THE PROCESS

The procedure was fairly typical, with multiple teams, each consisting of four or five engineers each, working independently to develop and test the software. Meanwhile a team of around 60 independent experts carried out the verification and validation work.

"There was a huge amount of testing done for the 777X. We had dozens of engineers testing one block of code for 10 months," says Llewellyn

Testing of the power distribution software started on what is called non-flyable code A. The teams continued all the way to non-flyable code W, testing various units and integrations. By the time they got to that stage they had tested everything, including at systems level. The next stage was to test the flyable Y level. The teams then installed the software on an iron bird and tested it for redundancies and multiple systems failures before it was installed in a test aircraft.

With software playing such an integral role in modern airplanes, it is perhaps not surprising that software testing is such an involving endeavor that can often prove problematic. But as the complexity and amount of code on aircraft increases, aerospace companies will be forced to get to grips with software testing through both better technology and better management of the process. \\