CLIENT	**AdaCore**
PROJECT	**White Paper: When Testing is Not Enough**
OBJECTIVE	**Generate leads and educate executives and engineers on the increasing importance of formal methods in the development of software and software-driven systems**

COPY EXCERPT

When testing is not enough

Call or write CopyEngineer to receive a PDF of the complete case study.

Or view/download it online at: https://www.adacore.com/papers/when-testing-is-not-enough

The impact of complexity on software reliability

The complexity of software in embedded systems has grown at an exponential rate for years.

In software, size is an important measure of complexity. Larger software has more inputs, more states, and more variables. In other words: larger software has more things to test and more opportunities for things to go wrong. Over the last few decades, increasing software complexity has meant an ever-increasing challenge in the verification of critical systems.

Critical systems include those requiring a high degree of dependability, including safety and security, and are typical of numerous industries, such as aerospace and defense, automotive, medical, energy generation and distribution, hazardous material management, and cybersecurity. The required maximum probability of failure in the most critical elements of these systems may be on the order of 10-7 to under 10-9 failures per hour.

Cost and schedule impacts

Software complexity impacts the cost and schedule of system development. The National Research Council (NRC) found that large software projects show very high rates of delay, cost overrun, and cancellation.[iii] Both dependable and typical commercial software programs suffer similar low success rates. The NRC found this unsurprising, “because dependable applications are usually developed using methods that do not differ fundamentally from those used commercially. The developers of dependable systems carry out far more reviews, more documentation, and far more testing, but the underlying methods are the same.”

Accidents and disruptions

As software has grown more complex, these underlying software-development methods have frequently failed to prevent disasters, even in systems developed to standards and extensively tested.

- On January 15, 1990, an undetected defect in a new version of switching software brought down ATT’s global long-distance network for more than nine hours. ATT lost more than $60 million in unconnected calls and suffered a severe blow to its reputation.[iv]
- In January 2017, the US FDA and Dept. of Homeland Security issued warnings against at least 465,000 St. Jude’s Medical RF-enabled cardiac devices. Software vulnerabilities in the devices could allow hackers to remotely access a patient's implanted device and disable therapeutic care, drain the battery, or even administer painful electric shocks. Short-selling firm Muddy Waters had revealed these flaws in August 2016, based on a report by the security firm MedSec, alleging negligence in St. Jude Medical’s software development practices.[v]