AUTOMATIC RECOVERY FROM SOFTWARE FAILURE

By PAUL ROBERTSON *and* BRIAN WILLIAMS

A model-based approach to self-adaptive software.

In complex concurrent critical systems, such as autonomous robots, unmanned air vehicles, and space systems, every component is a potential point of failure. This is true not only of embedded systems but also of purely software systems such as distributed and cyber applications. Typical attempts to make such systems more robust and secure are both brittle and incomplete due to reliance on manual identification of and solutions to potential failures such as by using exception mechanisms. That is, the security is easily broken, and there are many possible failure modes that are not handled. Failures may be rare events so it is less easy to test for good coverage of fault scenarios. Techniques that expand to handling component-level failures are very expensive to apply, yet are still quite brittle and incomplete. This is not because engineers are lazy—the sheer size and complexity of modern information systems overwhelms the attempts of engineers, and myriad methodologies, to systematically investigate, identify, and specify a response to all possible failures of a system.