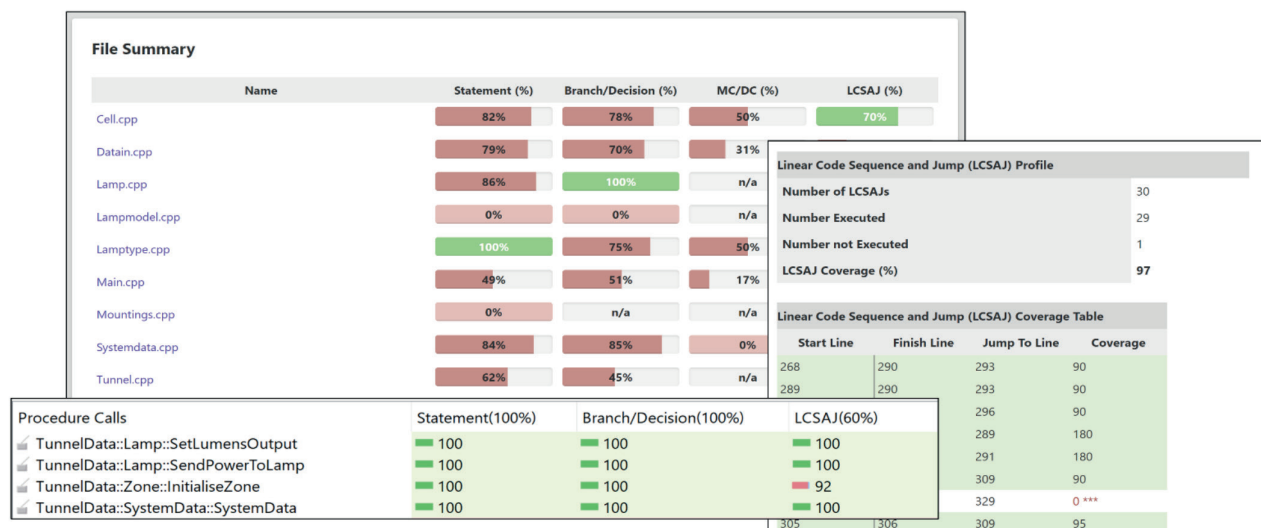


The most effective code coverage testing mechanism

An LCSAJ is a path fragment - a linear sequence of executable code. Each LCSAJ commences either from the start of the program or from a point to which control flow may jump. It is terminated either by a specific control flow jump or by the end of the program. When combined, LCSAJs form a finite set of paths through the source code application under test. The finite nature of this set of paths means that it is feasible to test all LCSAJs. LCSAJ coverage is a measure of how many LCSAJs are exercised by a particular test data set.



The implementation of LCSAJ coverage analysis in the LDRA tool suite® is uniquely complete and thorough because of its application of static and dynamic techniques in combination.

Benefits

- a practical implementation of the most thorough attainable source code coverage metric
- uniquely complete solution through the combined application of static and dynamic analysis techniques
- achieves a defined and attainable set of paths through the source code to be tested
- ensures that more, potentially error yielding, aspects of a software application are analysed than with any other available coverage method
- applicable to safety-critical development
- DAST technique applicable to the development of secure coding as part of a defence-in-depth cybersecurity strategy

Solution Details

- LCSAJ coverage is the most thorough of the attainable source code coverage metrics
- achieving a specified level of LCSAJ coverage exercises more of any given code base than achieving the same level of statement, branch, or similar coverage
- the limited number of LCSAJs in any given code base makes it practical to exercise a large majority of them
- static analysis identifies all the vital control flow information for each procedure and the inter-procedural links. The loop structure is exposed, and complexity metrics are generated
- dynamic analysis enables coverage, and forces the analysis of loops and the analysis of combinations of program