CLEANSCAPE

# Cleanscape Grayboxx

## Automated Software Test Tool for Cleaner Fortran Development

### Key Features

- Automatic test case generation
- Automatic unit testing
- Coverage analysis with complexity metrics
- Automatic stub generation
- Test harness generation
- Automatic test execution
- Module preparation
- Results verification
- Supports
  - *Automatic graybox test methodology*
  - *Automatic blackbox testing*
  - *Automatic whitebox testing*
  - *Automatic regression testing*
  - *Automatic assertion testing*
  - *Automatic mutation testing*
- Validates software as it executes
- Tests C, Fortran, Ada, Perl & Assembly language software using the same toolset*
- Documents and saves test results
- Supports requirements based test
  - *Completeness checks*
  - *Validates that the module is implemented per its requirements*

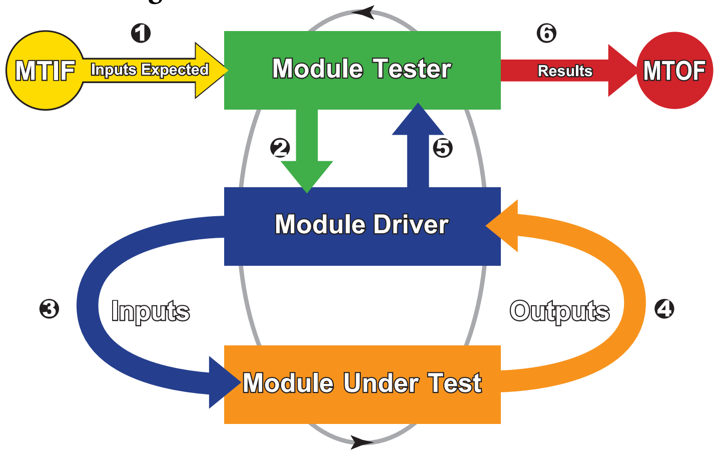
* C, Ada, Perl, and Assembly versions will be released throughout 2002

**Fortran Developers**

**An automated unit test and coverage tool for Fortran**

- Easy to use, no learning curve
- Use existing tools and processes
- Clean tests with no probe insertions
- Document results and save test runs

### Grayboxx Testing Process



*The Cleanscape Grayboxx automatic testing process is as follows: ❶ Input parameters and expected values are read from the Module Test Input File (MTIF) by Module Tester; ❷ Module Tester passes the input parameters to the Module Driver and launches the test case execution; ❸ Module Driver calls the Module Under Test (MUT) and passes the input values; ❹ MUT performs its task using the provided input values and generates results based on these values; ❺ Module Driver passes the obtained output to the Module Tester; ❻ The Module Tester compares the obtained (actual) result from executing the MUT to the expected results specified in the MTIF. The results of the comparison are stored in the Module Test Output File (MTOF).*

POORLY tested software not only risks the success of a software project, it risks the reputation and survival of the development organization. Since adequate testing can be extremely resource intensive, many software developers live with the risk. Others use test automation software designed to improve efficiency and increase software quality. However, they often find these test automation tools to be complex, time-consuming, limited in availability, and expensive. For Fortran developers, practical test automation tools have simply not been available. Instead, Fortran developers must typically use resource intensive and unreliable manual testing processes—or develop their own testing systems.

### Test system for Fortran

Cleanscape changes this with Grayboxx, an advanced software test automation system that provides Fortran developers an automatic method for testing software components and measuring test effectiveness. Grayboxx was designed to provide an easy-to-use testing system with no learning curve, using the programmer's existing toolset. This makes Grayboxx a readily-adoptable system that will quickly become essential for programmers, developers, and testers.

Whether integrating the system into established processes or using it as the foundation for a new test process, Grayboxx helps developers to increase the efficiency of testing while getting higher quality results.

# Cleanscape Grayboxx

***Automated Software Test Tool for Cleaner Fortran Development***

**Cleanscape Grayboxx** is a complete software life-cycle testing toolset developed for programs written in C, Fortran, Ada, Perl, & Assembly.* Grayboxx provides a complete software testing solution that verifies functional and structural performance requirements for mission critical applications. Grayboxx automatically conducts the following test methodologies: Blackbox Testing, Whitebox Testing, Regression Testing, Assertion Testing, and Mutation Testing.

Grayboxx speeds the development process by allowing developers and test engineers to automatically: generate test cases, conduct coverage analysis with complexity metrics, conduct unit performance testing with no probe insertions, generate test stubs, generate test harnesses, execute tests, prepare modules, and verify results. Grayboxx also allows for both full and partial regression testing, allowing the tester to run the same test more than once or to name the test titles to run with a subset of test cases.

## Application

Grayboxx supports real-time and non-real-time testing and facilitates automatic generation of the test harness for the Module Under Test (MUT). Project teams can use the Grayboxx tools to help ensure the product operates as specified, enforce system requirements, validate software proof of correctness and perform system level simulation to support software testing.

Developers working with inherited or legacy code can use Grayboxx to establish range and error values, even in applications code for which the source isn't available. It also allows developers to gain consistency across programs by tracking how each programmer or team did their unit testing and harness.

## Operation

Grayboxx provides an ideal balance between automatic test case generation and user control by allowing developers to integrate established processes and tools within the Grayboxx test methodology. By being flexible enough to allow developers to use their own debuggers and editors it reduces the learning curve of introducing a new control system into the testing process. In addition, the independent tester doesn't have to know the structure of the code to test it; the tester simply specifies inputs and outputs.

Generating test cases and conducting tests with Grayboxx is a simple process. Grayboxx scans the source code to be tested and automatically generates a list of input and initial values along with the expected results. The tester simply fills in the list of values and expected results. Grayboxx then executes the tests and prepares a results file that compares expected output to actual output. (see diagram: "Grayboxx Testing Process," front page). It also provides a percentage of tests passed and a percentage of branches covered. The results include date and time tested with tester information.

The tester can save the results for documentation purposes.

## Specifications

***Host environments***

- MS Windows 95, 98, NT, 2000
- Unix
- Solaris
- SGI
- Linux

***Languages***

- Fortran

* Ada, C, Perl, Assembly versions will be released throughout 2004/5

## Benefits

**Faster**

- *Automates the testing process*
- *Easy operation reduces learning curve, increases productivity*
- *Unit testing with no probe insertions allows programmers to run code at full speed*

**Better**

- *Increases product quality and reliability*

**Smarter**

- *Facilitates cross-platform development by providing common test environment for Unix, Linux, and Windows*
- *Provides a method to perform complete module testing*
- *Easily standardizes and automates established test processes*
- *Aids software development, selection and test improvement*

**Cheaper**

- *Increases return on investment*
- *Maximizes productivity*

**Cleaner**

- *Measures testing effectiveness*
- *Eliminates need to build test software*

CLEANSCAPE
software international

Providing software professionals with tools and solutions that make the software development process faster, better, smarter, cheaper...
*Cleaner.*

**www.cleanscape.net**

650 324-9200
sales@cleanscape.net