DARTT

Technical Brief

Major Features

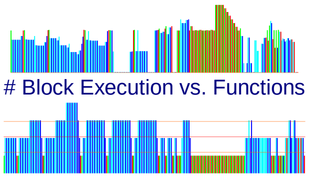
- Automated Function Testing
- Random, Statistical and Incremental Test Steps
- Automated Test Vector Generation (Input and Output)
- Automated Robustness Testing and Fault Injection
- Automated Coverage Analysis
- Automated Control Flow Analysis *
- Automated Exception Recording
- Determination and Recording of Execution Times
- Instrumentation support for Integration and System Testing
- Automated Documentation
- Code Analysis: Function and Type Properties
- Automated Instrumentation: Assertions, Data Monitoring *
- Auto-Porting
- Auto-Lock Removal
- Full support for Ada
  GNAT, Aonix, TLD1750

DARTT automates testing of C functions from test case generation to test evaluation and documentation. The product offers high productivity of testing, coverage analysis, identification of exceptions and execution times. Starting of a script in the source directory is fully sufficient to run the automated tests and to get the results in an automatically generated document.

Automated Test Procedure

- **User Interaction:**
  provide the source files, wait for completion, analyse the documented results
- **Test Case Generation** for all C and user-defined types incl. fault injection
- **Test Vector Recording** in ASCII and spreadsheet input format (e.g. MS-Excel)
- **Execution Time Recording** per function
- **Exception Recording:** file, function and line of raising statement, sorting according to user-defined categories
- **Data Monitoring** by automated instrumentation for function parameters, static and stack data, identification of limits.*
- **Assertions** on function parameters, static and stack data by automated instrumentation. *
- **Coverage Analysis:**
  block coverage (C0),
  MC/DC coverage (C3)*
- **Source Code Analysis:**
  Subprograms, types, source lines, comment and blank lines

* in preparation



# Block Execution vs. Functions

Block Nesting Level of Functions

Graphical presentation of results as available for DCRTT in preparation.

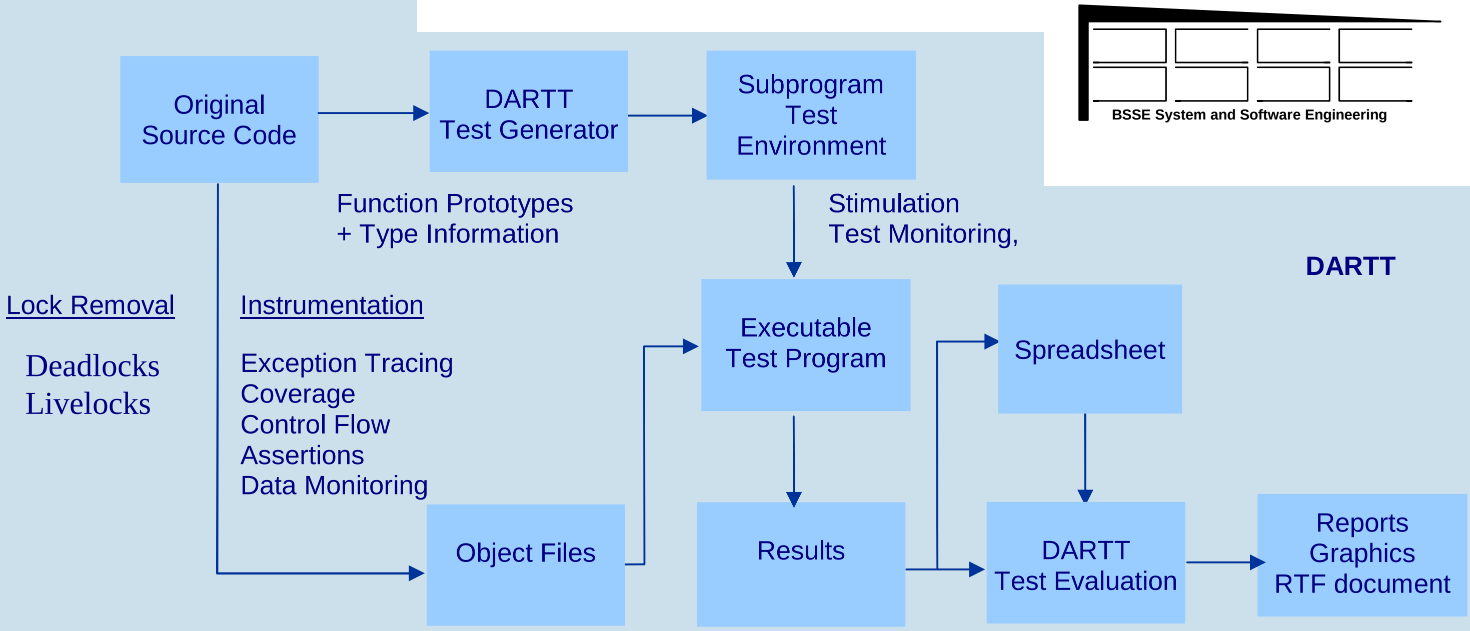
Auto-Porting

- **Adaptation of pragmas:**
  All pragmas which are not compliant with the target compiler are adapted.
- **Harmonisation** of package and file names
- **Adaption** of attributes like soie and objectsize.
- **Stubs** for functions not availabe on or non-portable to the target compiler, like assembler or library functions.
- **Immediately executable code** for target compiler

Auto-Lock Removal

- **Deactivation** of code which will block subprogram testing
- **Deadocks:** deactivation of code causing deadlocks like accept/select serviced by interrupts not avaiallbe after porting.
- **Livelocks:** deactivation of code causing deadlocks like forever-loops.

BSSE System and Software Engineering



DARTT

Technical Brief

Coverage Analysis

Coverage analysis provides objective measurement of how effective testing has been in executing the source code.

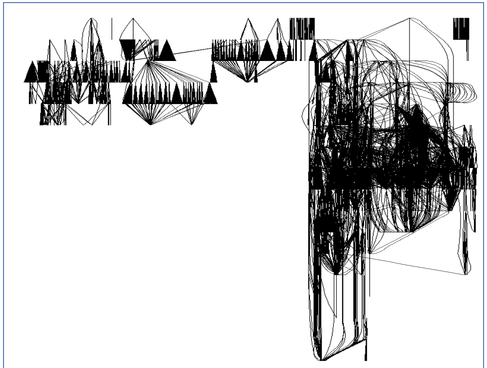
- **Code Coverage Metrics**
  Entry points, Call Returns
  Statements, Basic Blocks
  Decisions (branches)
  Conditions*
  MC/DC (DO-178B) *
  Exceptions

Code Analysis

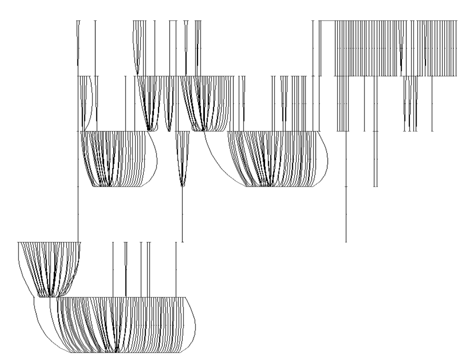
Static analysis provides information on code qunatities, functions and types.

- **Procedural Metrics** including code lines, comments, functions, and counts of most code constructs.
- **Function and Type Information:** prototypes, caller-callee dependencies, type dependencies

* in preparation

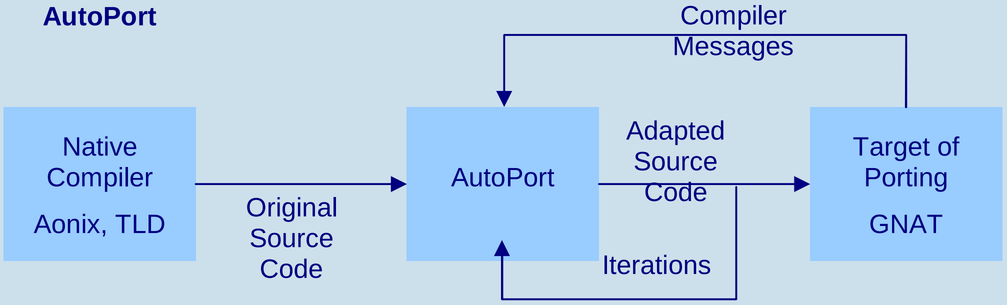


Function Hierarchy



Type Hierarchy

AutoPort



AutoPort: Fault Identification by Platform Diversification

Platform Diversification is a powerful capability for identification of faults very difficult to detect by usual tests and anylsis methods like “dormant faults”. It benefits from different and complementary capabilities of platform components such as processor type, OS and compiler. AutoPort supports platform porting at zero human effort and identifies differences in compiler messages. DARTT supports automated execution of the ported source code and identification of “hiding” faults at run-time.

Integration and System Tests

The capabilities for coverage analysis and exception identification are also available for integration and system level testing.

Integrations

The capabilities for coverage analysis and exception identification are also available for integration and system level testing.

DARTT

Supported Native Compilers

DARTT is available for GNAT, Ada compiler, other compilers on request

Supported Target Compilers

DARTT is available for GNAT, Aonix and TLD 1750 compilers, other compilers on request

Native Development Platforms

Windows 2000, XP
Linux in preparation

Further Information

BSSE’s testing tools support the unit / module, integration and system level testing of ANSI C (**DCRTT**) and Ada (**DARTT**) high integrity and business critical development projects. Further information and detailed product presentations are available from the IPL and BSSE websites or contact your local supplier.

BSSE System and Software Engineering

Dr. Rainer Gerlich BSSE System and Software Engineering

Voice : +49 (0)7545 911258
Fax : +49 (0)7545 911240
Email : dartt@bsse.biz