GRAMMATECH CodeSonar®

Automated Detection of Defects and Vulnerabilities with Static Code Analysis
Technologiepark Offenburg
In der Spoeck 10-12
77656 Offenburg, Germany

- Phone: +49 781 127 8118-0 (Germany)
- Phone: +33 3 68 33 58 84 (France)
- Fax: +49 781 63 920-29
- Email: info@verifysoft.com

www.verifysoft.com
Some Info About Us:

✓ Founded in June 2003

✓ Located in Technologie Park Offenburg (Germany)

✓ 300+ customers in Europe (2013)

✓ Distribution, consulting and support of testing and analysis tools for software quality in Europe.

✓ Distributor for GrammaTech in German speaking countries

✓ Development of software tools

✓ Seminars and Trainings
Our fields of activity, we distribute software to:

- Aerospace
- Automotive
- Medical
- Automation
- Transport
- Energy

... and further fields of activity where critical software is used and needed to be relied on.
Verifysoft Partners

- **Testwell**: Tampere (Finland)
- **CONFORMIQ**: Saratoga (USA)
- **GRAMMATECH**: Ithaca (USA)
- **software diagnostics**: Potsdam (Germany)
- **Spirula**: Paris (France)
- **elvior**: Tallinn (Estonia)
- **QMETRY**: Santa Clara (USA)
GrammaTech in a Nutshell

Core Competency
Static and dynamic analysis of source code and binaries

Founders and Staff
✓ Founded in 1988
✓ Tim Teitelbaum, CEO (Prof. Emeritus at Cornell Univ.)
✓ Tom Reps (Prof. at Univ. Wisconsin-Madison)
✓ 50+ employees (16 PhD experts in program analysis)

Products and Services
✓ CodeSonar static analysis tool
✓ GrammaTech research

Headquarters, R&D
Ithaca (State of New York, USA)
GrammaTech Research Customers

- Intelligence Advanced Research Projects Activity (IARPA)
- Department of Homeland Security
- Defense Advanced Research Projects Agency (DARPA)
- U.S. Navy: ONR, NAVSEA, SPAWAR, NAVAIR
- National Aeronautics and Space Administration (NASA)
- National Institute of Standards and Technology (NIST)
- Missile Defense Agency (MDA)
- United States Air Force (USAF)
- National Science Foundation (NSF)
- United States Army
- Department of Homeland Security
Conventional testing is necessary, BUT

✓ Only as good as the test cases
✓ Number of paths in code much greater than the number exercised
✓ Most paths go untested

Even in the most stringent approaches to testing
Development Testing

- Done by developers during software development
- The sooner a bug can be found, the cheaper it is to fix!
Code Sonar® Static Code Analysis

 ✓ Finds twice as many **critical defects** compared to other static analysis tools

 ✓ Improves security

 ✓ Designed for high assurance

 ✓ Employs sophisticated algorithms

 ✓ Supports custom checks

 ✓ Provides architecture visualisation

Get Code Sonar® as FREE TRIAL
**Considers software as a whole**

- Flow-based analysis of components or program
- Identifies serious generic defects and security vulnerabilities
- Crashes, data loss, data corruption

**Easy to use and extensible**

- Works with existing build system/requires no changes to the code
- Browser-based graphics for sharing and productivity
- Extensible with models and checkers

**Provides a search interface for results database**

- Enables users to annotate warnings
- Results and annotations persist across builds
Defect Checkers

- Buffer Overrun
- Null-Pointer Dereference
- Divide by Zero
- Uninitialized Variable
- Data Race
- Deadlock
- Free Non-Heap Variable
- Use After Free
- Double Free/Close
- Free Null Pointer
- Format String Vulnerability
- Race Conditions
- Unreachable Code
- Memory/Resource Leak
- Return Pointer to Local

- Mismatched Array New/Delete
- Race Conditions
- Unreachable Code
- Memory/Resource Leak
- Return Pointer to Local
- Mismatched Array New/Delete
- Invalid Parameter
- Missing Return Statement
- Dangerous Cast
- Inconsistent form
- User-Defined Checks
- Process Starvation
- Type Overrun
- Type Underrun
- Delete Object Created by malloc
More Defect Checkers

- Delete Object Created by new[]
- Free Object Created by new[]
- Free Object Created by new
- Redundant Condition
- Unused Value
- Useless Assignment
- Varargs Function Cast
- Ignored Return Value
- Null Test After Dereference

- Double Close
- TOCTTOU Vulnerability (time-of-check-to-time-of-use)
- Double Lock
- Double Unlock
- Try-lock that will never succeed
- Misuse of Memory Allocation
- Misuse of Memory Copying
- Misuse of Libraries
- Return Pointer to Free

And still more…

www.verifysoft.com
Architecture Visualisation

- Designed to optimise visual inspection and analysis
- Shows relationships between different elements in your code base
- Able to analyse large code bases: > 10 Mio. LOC
Smooth, real-time navigation

✓ Zoom, pan, expand, collapse, and more
✓ Even as program sizes grow into the millions of LOC

Layered data presentation

✓ Detail is progressively revealed as you zoom in
✓ Detail is abstracted as you zoom out
✓ A "link bundling" option effectively eliminates the visual clutter of individual links in high-level views while preserving connectivity information.

Display only the elements you are interested in

✓ Dedicated focus areas for arbitrary portions of the program
✓ Hide links and nodes that are not currently relevant
# CodeSonar®: Software Metrics

## Basic Counting Metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Project</th>
<th>File</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blank Lines</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Code Lines</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Comment Lines</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Include file instances</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lines with Code</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Lines with Comments</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Mixed Lines</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Top-level file instances</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Total Lines</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>User-defined functions</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Cyclomatic Complexity and Variations

<table>
<thead>
<tr>
<th>Metric</th>
<th>vG</th>
<th>evG</th>
<th>ivG</th>
<th>S1</th>
<th>mvG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyclomatic Complexity</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Essential Complexity</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Module Design Complexity</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Integration Complexity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modified Cyclomatic Complexity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
## Halstead Metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Symbol</th>
<th>Value</th>
<th>Value</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Operators</td>
<td>( N_1 )</td>
<td>( V )</td>
<td>( L )</td>
<td>( D )</td>
</tr>
<tr>
<td>Total Operands</td>
<td>( N_2 )</td>
<td>( V )</td>
<td>( L )</td>
<td>( D )</td>
</tr>
<tr>
<td>Distinct Operators</td>
<td>( n_1 )</td>
<td>( V )</td>
<td>( L )</td>
<td>( D )</td>
</tr>
<tr>
<td>Distinct Operands</td>
<td>( n_2 )</td>
<td>( V )</td>
<td>( L )</td>
<td>( D )</td>
</tr>
<tr>
<td>Halstead Program Length</td>
<td>( N )</td>
<td>( V )</td>
<td>( L )</td>
<td>( D )</td>
</tr>
<tr>
<td>Halstead Program Volume</td>
<td>( V )</td>
<td>( V )</td>
<td>( L )</td>
<td>( D )</td>
</tr>
<tr>
<td>Halstead Program Level</td>
<td>( L )</td>
<td>( V )</td>
<td>( L )</td>
<td>( D )</td>
</tr>
<tr>
<td>Halstead Program Difficulty</td>
<td>( D )</td>
<td>( V )</td>
<td>( L )</td>
<td>( D )</td>
</tr>
<tr>
<td>Halstead Programming Effort</td>
<td>( E )</td>
<td>( V )</td>
<td>( L )</td>
<td>( D )</td>
</tr>
<tr>
<td>Halstead Programming Time</td>
<td>( T )</td>
<td>( V )</td>
<td>( L )</td>
<td>( D )</td>
</tr>
<tr>
<td>Halstead Intelligent Content</td>
<td>( I )</td>
<td>( V )</td>
<td>( L )</td>
<td>( D )</td>
</tr>
</tbody>
</table>
MISRA-C 2004 and MISRA-C++ 2008

Built-in Support for Rules:

- 1.2
- 6.3
- 8.7 and 8.10
- 9.3
- 11.1
- 13.1
- 14.4
- 16.2 and 16.10
- 17.5
- 19.1, 19.5, 19.6 and 19.7
- 20.3 and 20.4
- 21.1

And others…
Security Checks

Common Weakness Enumeration (CWE) compatible

Supports

✓ “Build Security In” (BSI)
✓ Power of Ten Rules
✓ JPL Rules (NASA Jet Propulsion Laboratory)
Case Study: Recalled Medical Device

Software Forensics Lab at FDA Case Study
 ✓ 200 KLOC C program for old device
 ✓ Device had failed in the field and had triggered an FDA investigation, during which the maker had to provide all source code to FDA investigators
 ✓ The FDA used static analysis to find issues in the code

Results
 ✓ 127 serious problems
 ✓ Manufacturer was previously aware of 82 of these
 ✓ 45 were not previously known

Case Study: Mars Rover

- NASA JPL Mars Curiosity Rover
- Most ambitious Mars rover to date
- 2.7 meters long and 900 kg

Rover electronics module
- RAD750 CPU, 256 kB of EEPROM, 256 MB of DRAM
- 2 GB of flash memory
- VxWorks multi-tasking RTOS
- 2 million lines of code

Results: NASA/JPL decided to use Gramma-Tech CodeSonar® to analyse the software and enforce the Power of 10 Coding Standard.
Case Study: Satellite Ground Station

- NASA’s Tracking and Data Relay Satellite System (TDRSS)
- Nine geosynchronous satellites and three ground stations
- Launch support and on-orbit communications and tracking services for most NASA missions
- Striving for 99.9% availability (excl. scheduled down times)
- Detailed study of ROI for the usage of CodeSonar on ground station software
- Conducted by NASA White Sands under the sponsorship of NASA Ames Research Center and the NASA IV&V Facility
- Found that tool price plus cost of labor to use was about equal to debugging labor cost
- However, tool use prevents outages, whereas reactive debugging does not

Results: CodeSonar was adopted to reduce software errors.
More Case Studies at www.verifysoft.com

GrammaTech CodeSonar®: Fallstudien

- NASA Mars Rover Searches for Signs of Life with the Help of CodeSonar
- GrammaTech Helps RCA Ensure Reliability of Life-Saving Mobile App
- FDA Uses GrammaTech to Analyze Recalled Medical Devices
- NASA Uses GrammaTech to Increase Satellite Uptime
- Boston Scientific Streamlines Analysis of Medical Device Software
- CodeSonar Helps Vivante Deliver Reliable GPU Cores On Time
- Critical Link: GrammaTech Ensures High Reliability of DSP Software
- CodeSonar Helps Harvard Apparatus Tackle the Medical Device Market
- CodeSonar Streamlines Certification of High-Security Devices
CodeSonar® Demo

Evaluation available - Try on your own code
Technologiepark Offenburg
In der Spoeck 10-12
77656 Offenburg, Germany

✓ Phone: +49 781 127 8118-0 (Germany)
✓ Phone: +33 3 68 33 58 84 (France)
✓ Fax: +49 781 63 920-29
✓ Email: info@verifysoft.com

www.verifysoft.com