Runtime Verification Inc. applies runtime verification-based techniques to improve the safety, reliability, and correctness of software systems for aerospace, automotive, and the blockchain.
The runtime verification term was coined by Professor Grigore Rosu (UIUC) and his colleague Dr. Klaus Havelund (NASA) in three papers they published in 2001 and 2002. The papers received the Most Influential Paper award at the ACM/IEEE Automated Software Engineering Conference in 2016, the Test of Time award at the Runtime Verification Conference in 2018, and respectively the Best Software Science Paper award at ETAPS 2002.

The company was founded in 2010.
During **runtime verification** we prove that the specification and the implementation are tightly connected, hence two rigidity points.
What is runtime verification?

A subfield of program analysis and verification – just like static analysis – aimed at verifying computing systems as they execute: with good scalability, rigor, and no false alarms.

Runtime verification complements static analysis.

Runtime verification is different from static analysis because: it executes programs to analyze, observes execution traces, builds models from the execution trace, and analyzes the model.
RV-Match is a semantics based automatic debugger for common and subtle C errors, and the most advanced and precise semantics-based bug finding tool.

RV-Match gives you:
• an automatic debugger for subtle bugs other tools can’t find, with no false positives
• seamless integration with unit tests, build infrastructure, and continuous integration
• a platform for analyzing programs, boosting standards compliance and assurance
In a Toyota ITC benchmark evaluation, comparing RV-Match with various static analysis solutions, our product received the best score by finding more bugs than the static analysis tools and achieving a perfect false positive rate of zero false positives.
NASA core Flight Executive (cFE) is a development and run-time environment for enabling cross-platform embedded systems.

RV-Match detected:
• 15 undefined behaviors
• 1036 implementation-defined behaviors
Unit testing with RV-Match

RV-Match can replace GCC or Clang in unit-testing infrastructure to detect undefined behavior while executing the tests.
Analysis with RV-Match – the kcc tool

**undef.c**

```c
int main() {
    int a;
    &a + 2;
}
```

$kcc$ bounds.c
$kcc$ ./a.out
A pointer (or array subscript) outside the bounds of an object:
  > in main at undef.c:3:7

Undefined behavior (UB-CEA1):
  see C11 section 6.5.6:8 http://rvdoc.org/C11/6.5.6
  see C11 section J.2:1 item 46 http://rvdoc.org/C11/J.2
  see CERT-C section ARR30-C http://rvdoc.org/CERT-C/ARR30-C
  see CERT-C section ARR37-C http://rvdoc.org/CERT-C/ARR37-C
  see CERT-C section STR31-C http://rvdoc.org/CERT-C/STR31-C
  see MISRA-C section 8.18:1 http://rvdoc.org/MISRA-C/8.18
  see MISRA-C section 8.1:3 http://rvdoc.org/MISRA-C/8.1

**kcc detects and reports undefined behavior with ISO C11 citation.**
Analysis with RV-Match – the kcc tool

`bounds.c`

```c
#include <stdio.h>
#include <string.h>

int main()
{
    struct { int a; int b; } s = {0, 1};
    int * p = &s.a;
    printf("%d\n", *(p + 1));
}
```

```
$ kcc bounds.c
$ ./a.out
Dereferencing a pointer past the end of an array:
    > in main at bounds.c:9:7

Undefined behavior (UB-CER4):
    see C11 section 6.5.6:8 http://rvdoc.org/C11/6.5.6
    see C11 section J.2:1 items 47 and 49 http://rvdoc.org/C11/J.2
    see CERT-C section ARR30-C http://rvdoc.org/CERT-C/ARR30-C
    see CERT-C section ARR37-C http://rvdoc.org/CERT-C/ARR37-C
    see CERT-C section STR31-C http://rvdoc.org/CERT-C/STR31-C
    see MISRA-C section 8.18:1 http://rvdoc.org/MISRA-C/8.18
    see MISRA-C section 8.1:3 http://rvdoc.org/MISRA-C/8.11
```
Analysis with RV-Match – the kcc tool

overflow.c

```c
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>

void process_something(int size) {
    size += 1; // check for overflow
    if (size < 0) return;
    char *string = malloc(size);
    string[0] = 'x';
    string[1] = '\000';
    puts(string);
}

int main(int argc, char** argv) {
    process_something(2);
    process_something(INT_MAX);
}
```

$ kcc overflow.c
$ ./a.out

Signed integer overflow:
> in process_something at overflow.c:6:7
in main at overflow.c:18:7

Undefined behavior (UB-CCV1):
see C11 section 6.5:5 http://rvdoc.org/C11/6.5
see C11 section J.2:1 item 36 http://rvdoc.org/C11/J.2
see CERT-C section INT32-C http://rvdoc.org/CERT-C/INT32-C
see MISRA-C section 8.1:3 http://rvdoc.org/MISRA-C/8.1
## Analysis with RV-Match – the kcc tool

<table>
<thead>
<tr>
<th>Error</th>
<th>Message</th>
<th>ISO C11 Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>UB-CB1</td>
<td>Types of function call arguments aren't compatible with declared types after promotions.</td>
<td>6.5.2.2:6, J.2:1 #39</td>
</tr>
<tr>
<td>UB-CB2</td>
<td>Function call has fewer arguments than parameters in function definition.</td>
<td>6.5.2.2:6, J.2:1 #38</td>
</tr>
<tr>
<td>UB-CB3</td>
<td>Function call has more arguments than parameters in function definition.</td>
<td>6.5.2.2:6, J.2:1 #38</td>
</tr>
<tr>
<td>UB-CB4</td>
<td>Function defined with no parameters called with arguments.</td>
<td>6.5.2.2:6, J.2:1 #38</td>
</tr>
<tr>
<td>UB-CCV1</td>
<td>Signed integer overflow.</td>
<td>6.5:5, J.2:1 #36</td>
</tr>
<tr>
<td>UB-CCV2</td>
<td>Conversion to integer from float outside the range that can be represented.</td>
<td>6.3.1.4:1, J.2:1 #17</td>
</tr>
<tr>
<td>UB-CCV3</td>
<td>Floating-point value outside the range of values that can be represented after conversion.</td>
<td>6.3.1.5:1, J.2:1 #18</td>
</tr>
<tr>
<td>UB-CCV5</td>
<td>Empty value to type other than void.</td>
<td>6.3.2.2:1, J.2:1 #23</td>
</tr>
<tr>
<td>UB-CCV6</td>
<td>Casting void type to non-void type.</td>
<td>6.3.2.2:1, J.2:1 #23</td>
</tr>
<tr>
<td>UB-CCV7</td>
<td>Conversion from pointer to integer of a value possibly unrepresentable in the integer type.</td>
<td>6.3.2.3:6, J.2:1 #24</td>
</tr>
<tr>
<td>UB-CCV11</td>
<td>Conversion to a pointer type with a stricter alignment requirement (possibly undefined).</td>
<td>6.3.2.3:7, J.2:1 #25</td>
</tr>
<tr>
<td>UB-CCV12</td>
<td>Floating-point overflow.</td>
<td>6.5:5, J.2:1 #36</td>
</tr>
<tr>
<td>UB-CEA1</td>
<td>A pointer (or array subscript) outside the bounds of an object.</td>
<td>6.5.6:8, J.2:1 #46</td>
</tr>
<tr>
<td>UB-CEA2</td>
<td>Pointer difference outside the range that can be represented by object of type ptrdiff_t.</td>
<td>6.5.6:9, J.2:1 #50</td>
</tr>
<tr>
<td>UB-CEA5</td>
<td>Computing pointer difference between two different objects.</td>
<td>6.5.6:9, J.2:1 #48</td>
</tr>
<tr>
<td>UB-CEB2</td>
<td>The right operand in a bitwise shift is negative.</td>
<td>6.5.7:3, J.2:1 #51</td>
</tr>
<tr>
<td>UB-CEB3</td>
<td>The right operand in a bitwise shift is greater than or equal to the bit width of the left operand.</td>
<td>6.5.7:3, J.2:1 #51</td>
</tr>
<tr>
<td>UB-CEB4</td>
<td>The left operand in a bitwise left-shift is negative.</td>
<td>6.5.7:4, J.2:1 #52</td>
</tr>
<tr>
<td>UB-CEB6</td>
<td>The right operand in a bitwise shift is negative.</td>
<td>6.5.7:3, J.2:1 #51</td>
</tr>
<tr>
<td>UB-CEB7</td>
<td>The right operand in a bitwise shift is greater than or equal to the bit width of the left operand.</td>
<td>6.5.7:3, J.2:1 #51</td>
</tr>
</tbody>
</table>

More than 200 reported issues
At the heart of RV-Match is a complete formal semantics of the ISO C standard powered by the K framework.
True semantics-based analysis

- KCC
- C source (.c, .h)
- Native build system
- Preprocessor, parser
- Translation interpreter
- Translation semantics
- Natively-compiled program
- K abstract syntax
- Execution semantics
- Execution interpreter
- Target platform
Partners & Customers

- NSF
- NASA
- DARPA
- DENSO
- TOYOTA
- Boeing
- INPUT\OUTPUT
- Ethereum
- Algorand
- Maker
- Gnosis
- Casper
Our company is fueled by people. We are pioneers in the runtime verification community, with hundreds of publications that shaped the field.
Main Offices

University of Illinois at Urbana-Champaign
Ranked #2 worldwide in Formal Methods

University of Bucharest
Ranked #1 University in Romania