What We Do

Runtime Verification Inc. applies runtime verification-based techniques to improve the safety, reliability, and correctness of software systems for aerospace, automotive, and the blockchain.

- dynamic analysis
- formal design
- formal analysis framework
- 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.
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
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.
Case study – NASA cFE

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
RV-Predict automatically detect the rarest and most difficult data races in your Java and C/C++ code, saving on development and testing effort with the most precise race finder available.

RV-Predict gives you:
- an automatic debugger for subtle Java and C/C++ data races with no false positives
- seamless integration with unit tests, build infrastructure, and continuous integration
- a maximal detection algorithm that finds more races than any sound dynamic tool
RV-Predict/C and LLVM ThreadSanitizer both detected a race on the Stolz queue. However, in producing a report in 5-10 seconds, RV- Predict/C bested ThreadSanitizer by a factor of 10 as the latter took more than a minute to generate the same report.
We formalize your smart contract as a mathematical specification. We refine the specification to match the target low-level virtual machine, and then compile the smart contract from its high-level language (e.g., Solidity, Vyper, Plutus) to virtual machine bytecode. We can then prove whether the bytecode satisfies the refined specification.
We developed formal models of Casper and Algorand, and specified two classes of properties: safety (that the protocol guarantees consensus) and liveness (that the protocol will always continue to make progress). The formal models make explicit the assumptions under which these properties are satisfied, which is extremely important for properly setting the expectations from systems built on top of them.
Virtual Machines

In partnership with blockchain research firm, IOHK, we designed and developed IELE, a new virtual machine that represents an evolution of sorts of the Ethereum virtual machine (EVM). It leverages the KEVM project, which successfully demonstrated that a K formal specification of EVM can generate automatically, a "fast enough" virtual machine.
For the larger Ethereum ecosystem we specified **ERC20-K** and **ERC777-K**, the mathematically rigorous formalization of the first of its kind ERC20 and increasingly popular ERC777 token standards. These two industry first formalizations facilitate formal verification of token implementations.
Partnerships

We always valued the friends and partners who have contributed mightily to our success. Therefore, we are happy to introduce the new commissions for the following services:

**Customer Introduction by Partner** – A qualified introduction to a Runtime Verification executive, that leads to a new completed engagement. (NET 5%)

**Sales Made by Partner** – An executed contract to a new Runtime Verification customer. (NET 15%)