# Stratis DevEx:

# Revamp Stratis Smart Contract Tooling and Developer Experience

**Prepared For:**
Stratis Decentralized Accelerator

InterFlux Decentralized Governance Board

**Prepared By**:
Allister Beharry

**Date:**

Nov 18, 2022

# EXECUTIVE SUMMARY

The global pandemic has accelerated enterprise adoption of blockchain technology as the need for decentralized resilient global tracking of transactions, payments, shipments, records, digital assets, and the flow of goods and information, has become more apparent to organizations. Blockchain spending is expected to increase from $6.6 billion in 2021 to $19 billion by 2024 in diverse industries like financial services, supply chain management, and healthcare.

A key differentiator and strategic advantage of Stratis over other enterprise blockchain platforms like ConsenSys Quorum and Hyperledger Fabric is the strength of the tooling and eco-system and community around .NET and C#. Microsoft has invested decades and billions of dollars into making .NET one of most developer-friendly and adopted enterprise technology platforms worldwide. The switch to open-source has increased the scope and pace of .NET technology adoption: .NET Core ranked as the #1 most-loved framework by developers on the 2021 Stack Overflow Developer Survey while .NET languages like C# and Visual Basic consistently rank in the top ten of language popularity indexes like the TIOBE index. By contrast, the Solidity language for Ethereum and Quorum smart contract development does not rank in the top 50 languages on the index. By using .NET Core both as the development and runtime platform for smart contracts and C# as the primary smart contract development language, Stratis is uniquely positioned to address several existing barriers to enterprise blockchain adoption.

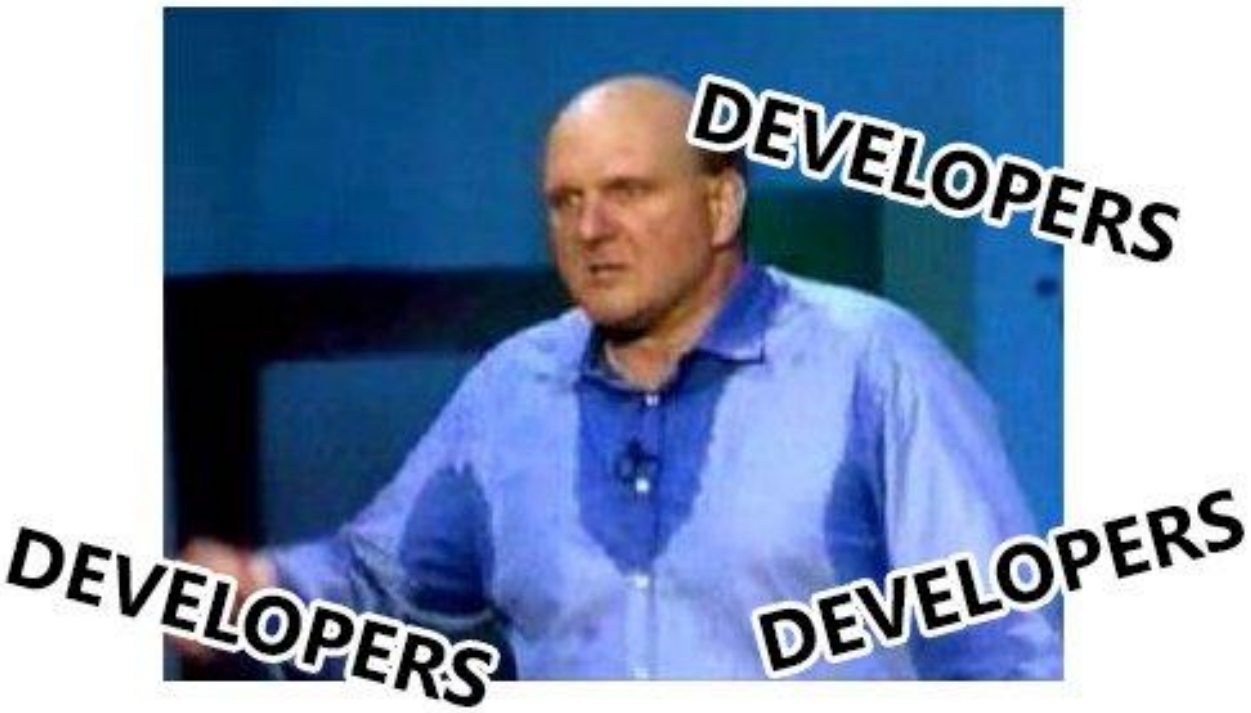
Blockchain technology is complex and enterprises desiring to adopt it must contend both with the complexity and paradigm shift of decentralized digital ledger technology, as well as the need for developers to learn entirely new languages like Solidity and new environments like Remix IDE or Truffle. Lack of knowledge and a talent shortage of blockchain developers are two of the most significant barriers to

enterprise adoption of blockchain technology, as is problematic integration with existing systems.

Security is another critical issue for most enterprises seeking to adopt blockchain technology and is directly tied to the unfamilarity by enterprise developers with current blockchain and smart contract languages and tools. According to blockchain security firm SlowMist the first half of 2022 saw 187 blockchain security incidents and damages totaling nearly $2 billion with a more than half of those incidents due to “design defects and smart contract vulnerabilities”, in contrast to other types of attacks like phishing or private key exposure.

It is far easier for developers to write secure smart contract code using languages and tools they are familiar with rather than using an unfamiliar toolset. Microsoft has invested a great deal into making a world class developer experience for .NET with the freely available Visual Studio IDE. Visual Studio contains several extension points that allow it to be tailored for specialized development tasks and workloads like smart contract development. There are also a great deal of community-built tools for .NET development including libraries for .NET static analysis and formal verification which are two common techniques for improving smart contract security.

By integrating all the current tools and processes for smart contract development and security auditing into Visual Studio and supporting smart contract security techniques like formal verification, Stratis can leverage the investment and commitment Microsoft has made to tooling and developer experience and increase adoption of the Stratis smart contract and blockchain platform by enterprises and industries.


DEVELOPERS
DEVELOPERS
DEVELOPERS

# PROJECT OVERVIEW

This project seeks to revamp the existing Stratis smart contract tooling and developer experience by integrating existing tools and libraries for smart contract validation, static analysis, and security auditing, into Visual Studio. Currently the toolset for smart contract development is split between Visual Studio for C# development, a command line tool for contract validation and compilation to bytecode, and the Cirrus Core wallet or a REST API HTTP client for deployment and calling smart contract methods. The immediate goal is to provide a familiar, frictionless, integrated interface and toolset for smart contract development, deployment, and security auditing that will allow .NET developers and teams to reuse their existing knowledge as much as possible to develop secure smart contracts. The long-term project objective is to remove some common barriers to blockchain adoption and foster interest in the Stratis platform by enterprises and developers who already have an investment in .NET and Visual Studio knowledge and expertise.

The core development tasks are to integrate the existing Silver library for smart contract static analysis and formal verification into Visual Studio Roslyn analyzers and a Visual Studio extension. Silver will be retargeted as a general purpose library for .NET static analysis and formal verification and will provide the core functions for the new Stratis analyzers and extensions . The Stratis smart contract static analysis libraries and tools will live under the `Stratis.CodeAnalysis` namespace and complement the existing `Stratis.SmartContract` libraries . The Stratis Visual Studio analyzers and extension will aim to bring parity between the Stratis C# developer experience and the Solidity developer experience in IDEs like Remix or Visual Studio Code:

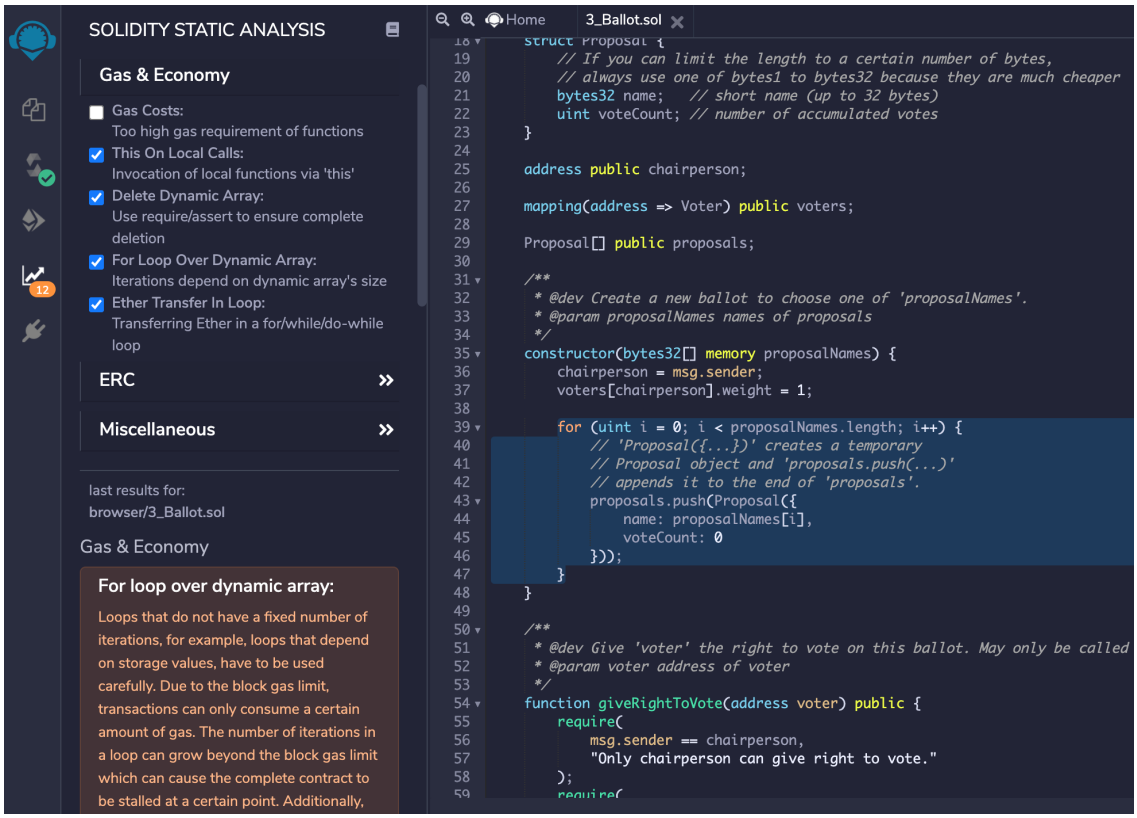



Figure 1. Solidity language static analysis in Remix IDE

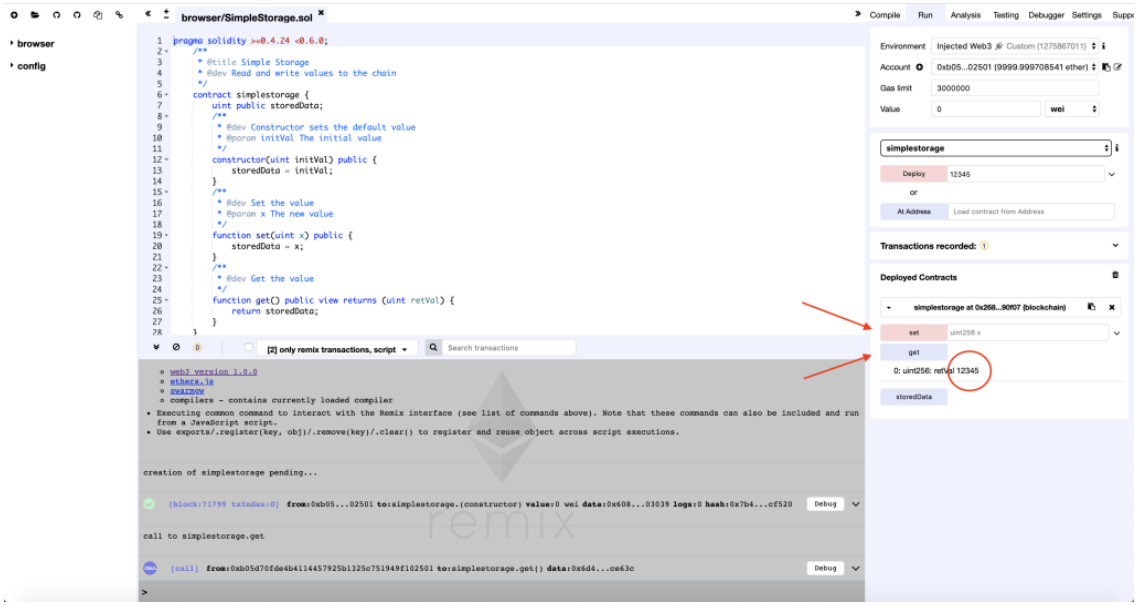



Figure 2. Solidity smart contract deployment in Remix IDE

The Stratis Visual Studio analyzers and extension will  implement the following features:

## C# Source Code Validation

All validation rules for smart contract CIL code present in the Stratis command-line validator tool will be ported to Roslyn C# syntax rules and made available to developers inside Visual Studio, providing immediate feedback on validation errors as developers code and removing the need to run a separate command-line tool.

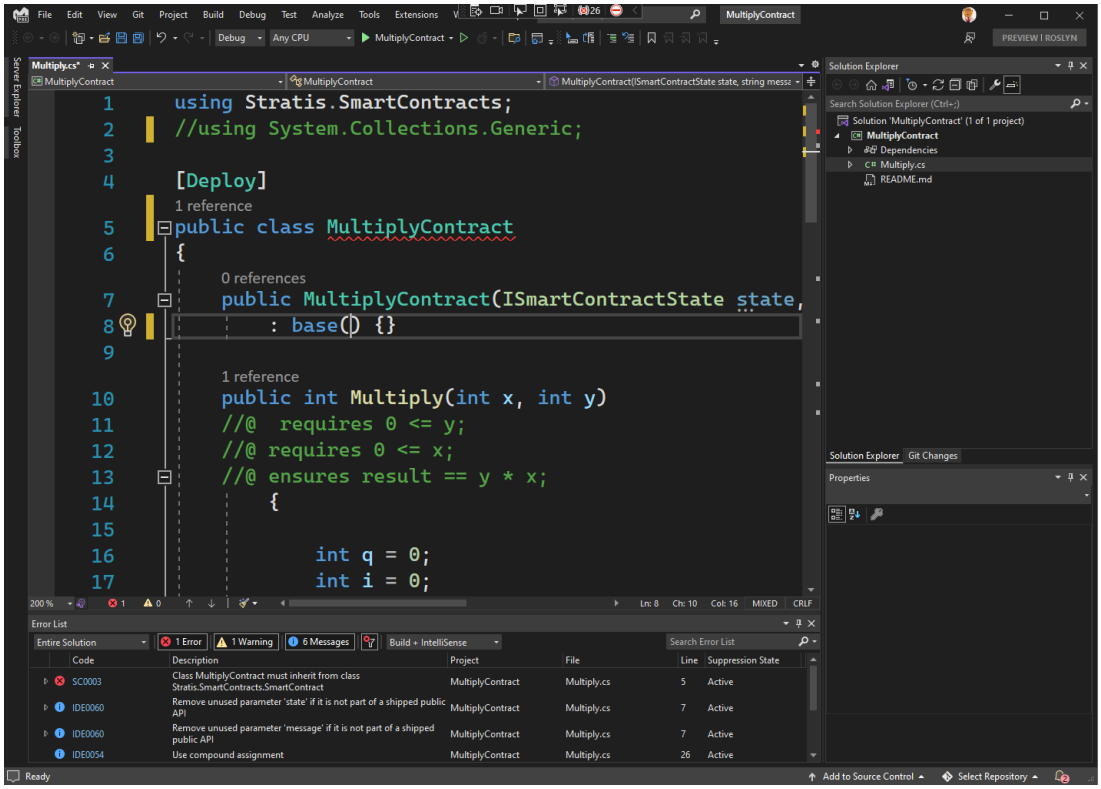



**Figure 3. Using the Silver Roslyn Analyzer validator inside Visual Studio**

# C# Source Code Exploration

The Stratis Visual Studio extension will implement for C# smart contract development many of the features of what the [ConsenSys Visual Solidity Developer ]VS Code extension does for Solidity development. One of these is source exploration: a visual way to interactively explore smart contract code using properties extracted from static analysis. Source exploration in the Stratis VS extension will be implemented using Silver.

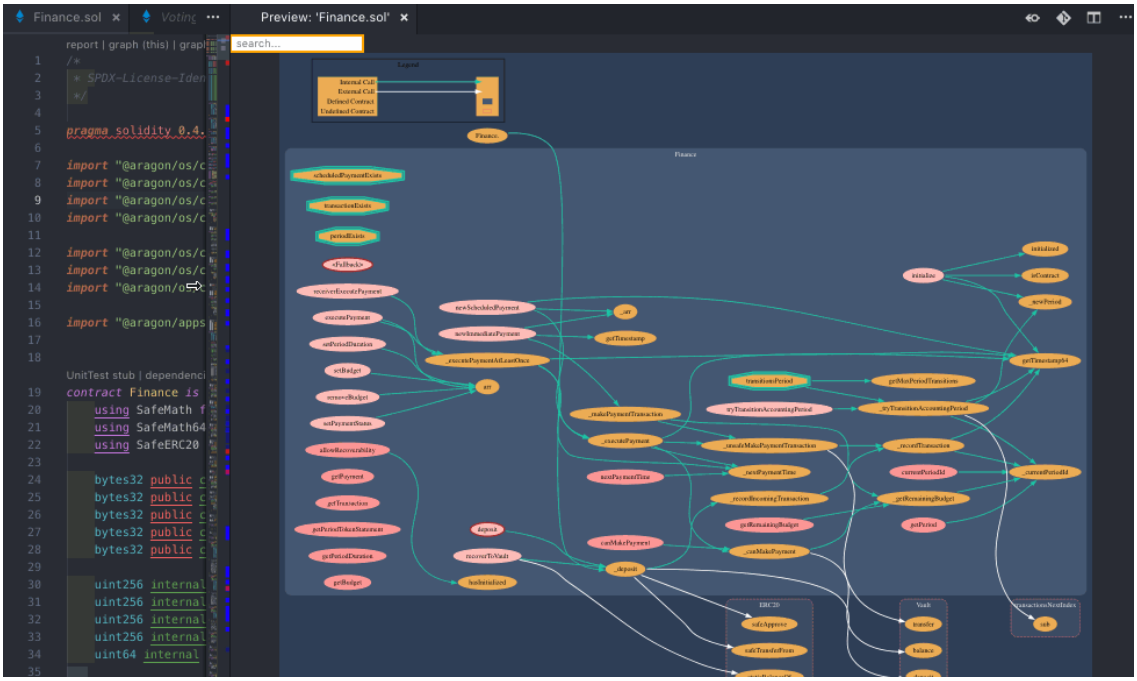



**Figure 4. Consensys Visual Solidity Developer source exploration interactive call graph**

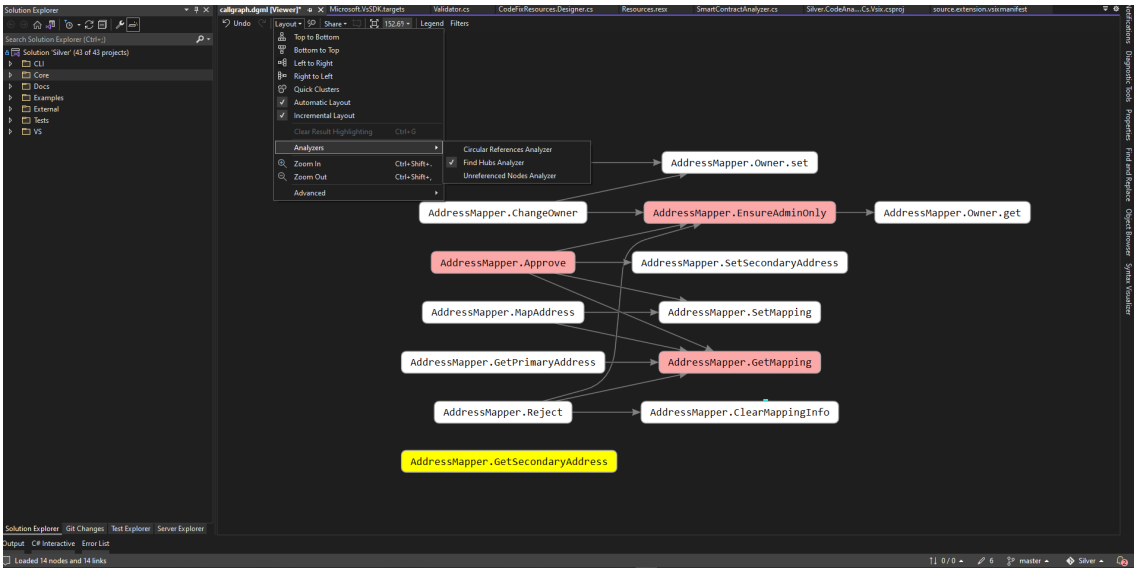



Figure 5. Silver call graph in Visual Studio

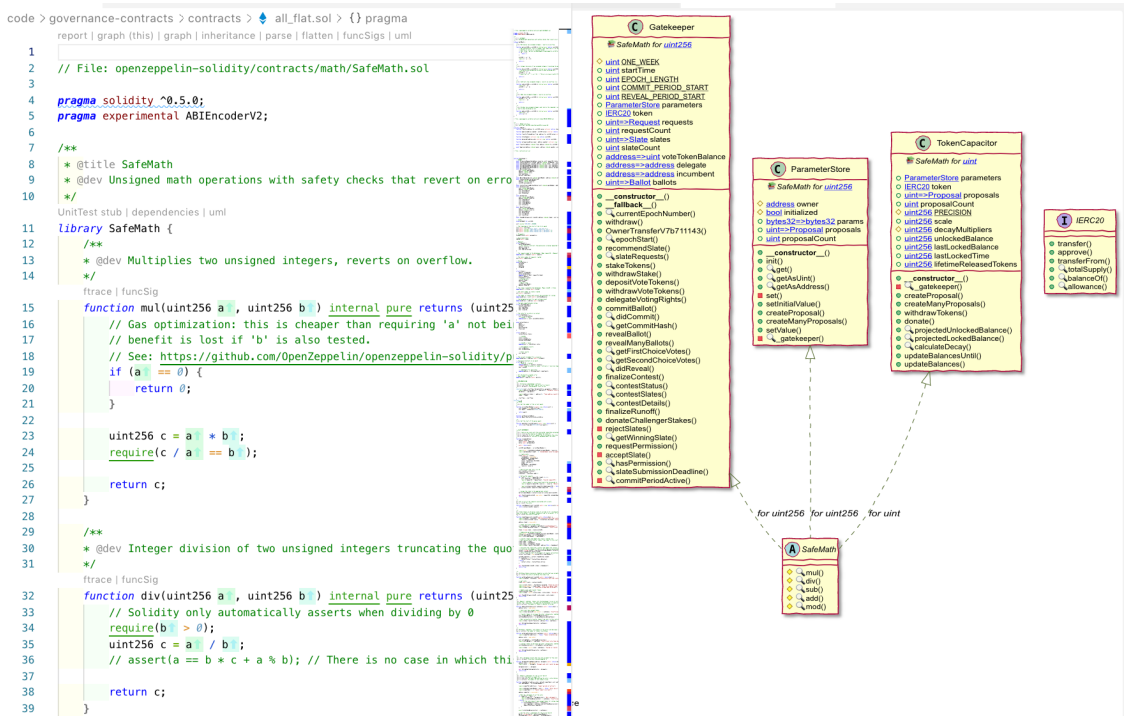



Figure 6. Consensys Visual Solidity Developer source exploration inheritance diagram

```
[INFO] Get smart contract assembly from Stratis pull request https://github.com/stratisproject/CirrusSmartContracts/pull/31 com
pleted in 1322ms.
[INFO] Target assembly is Stratis.PR547793172.dll.
```

```
In [6]: an.GetSummary()
```

```
[INFO] Analyzing class hierachy...
[INFO] Analyzing class hierachy completed in 0ms.
```

Out[6]:

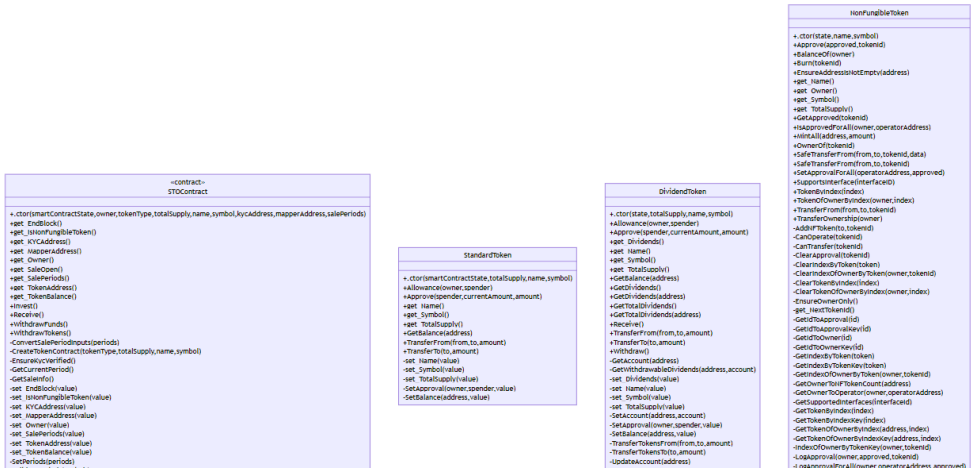



**Figure 7. Silver inheritance diagram**

# C# Method Disassembly and Gas Cost

Gas in Stratis smart contracts is metered according to the type and number of .NET CIL instructions executed by the smart contract runtime. Silver can disassemble each method in a .NET assembly, count the number of instructions and measure the gas cost of a smart contract method.

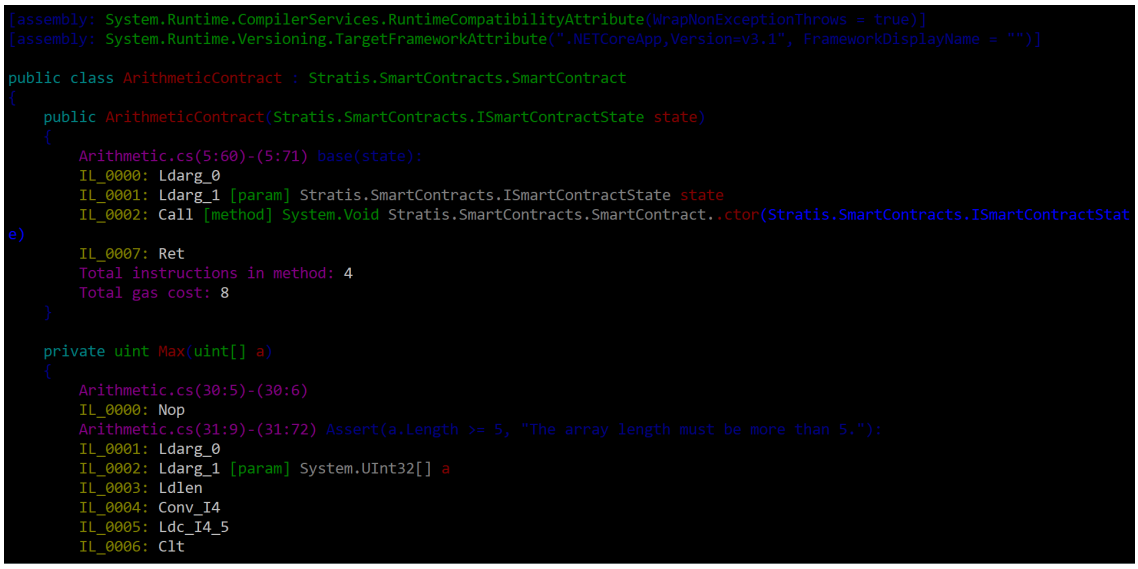



**Figure 8. Silver disassembly of a C# smart contract method**

The smart contract disassembler and gas cost will be integrated into the Stratis Visual Studio extension as a custom tool window allowing developers to be able to immediately see the CIL instructions executed for each smart contract method and the resulting calculated gas cost.

# Smart Contract Deploy and Run

The Solidity Remix IDE has the ability to deploy Solidity smart contracts to a particular environment or chain and execute the contract there with specified parameters, optionally recording the results.

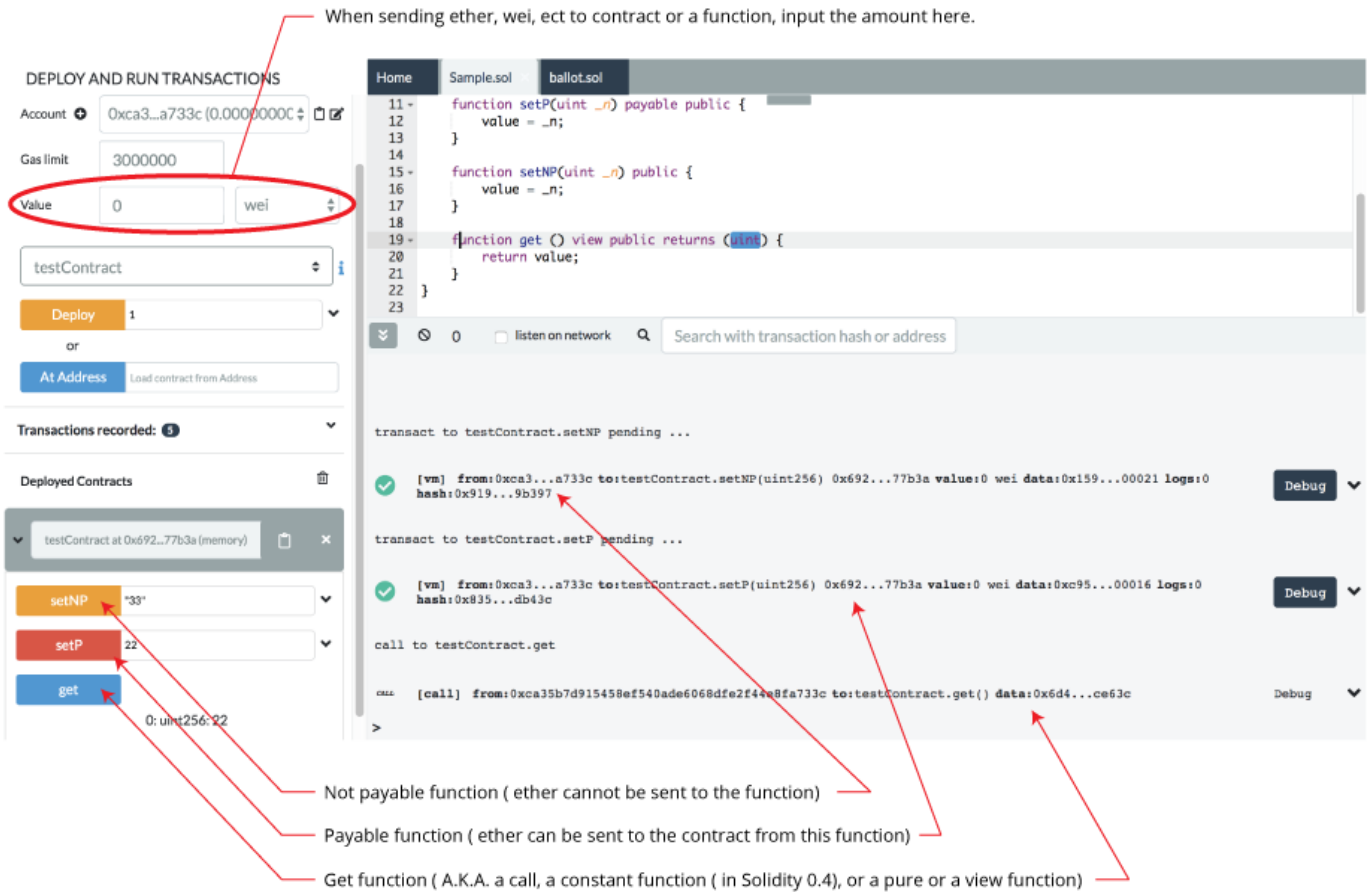



**Figure 9. Remix IDE Deploy and Run**

Using the extension Solidity developers can specify the HTTP endpoint address of an Ethereum node API to deploy using that API.

This feature will allow developers to use the REST API of a Stratis FullNode to deploy and interact with Stratis smart contracts inside Visual Studio. The developer will have a GUI interface to the full set of operations available from the REST API such as creating and calling smart contracts.

# Stratis Network Explorer

The [Truffle Visual Studio Code extension] allows developers to connect to different Ethereum networks and deploy contracts:

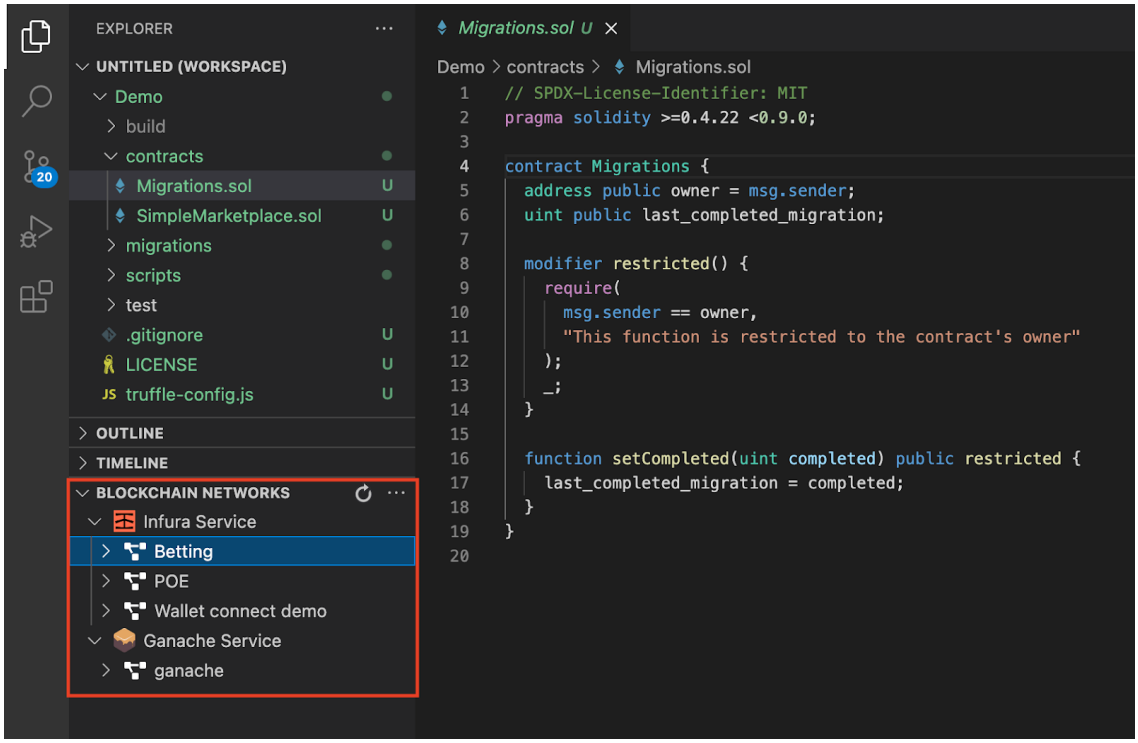



Figure 10. Truffle VS Code Blockchain Networks Explorer

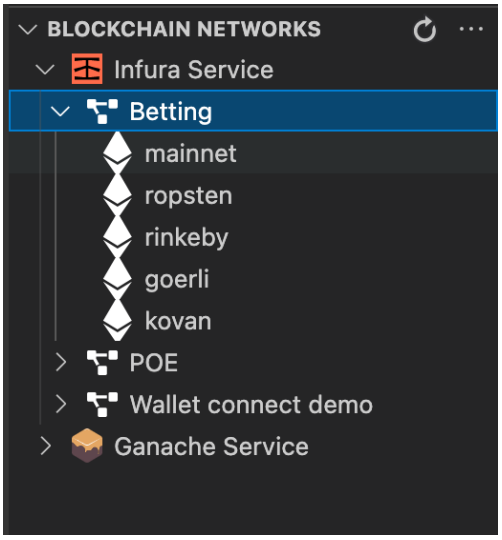



Figure 11. Truffle VS Code Blockchain Networks Explorer

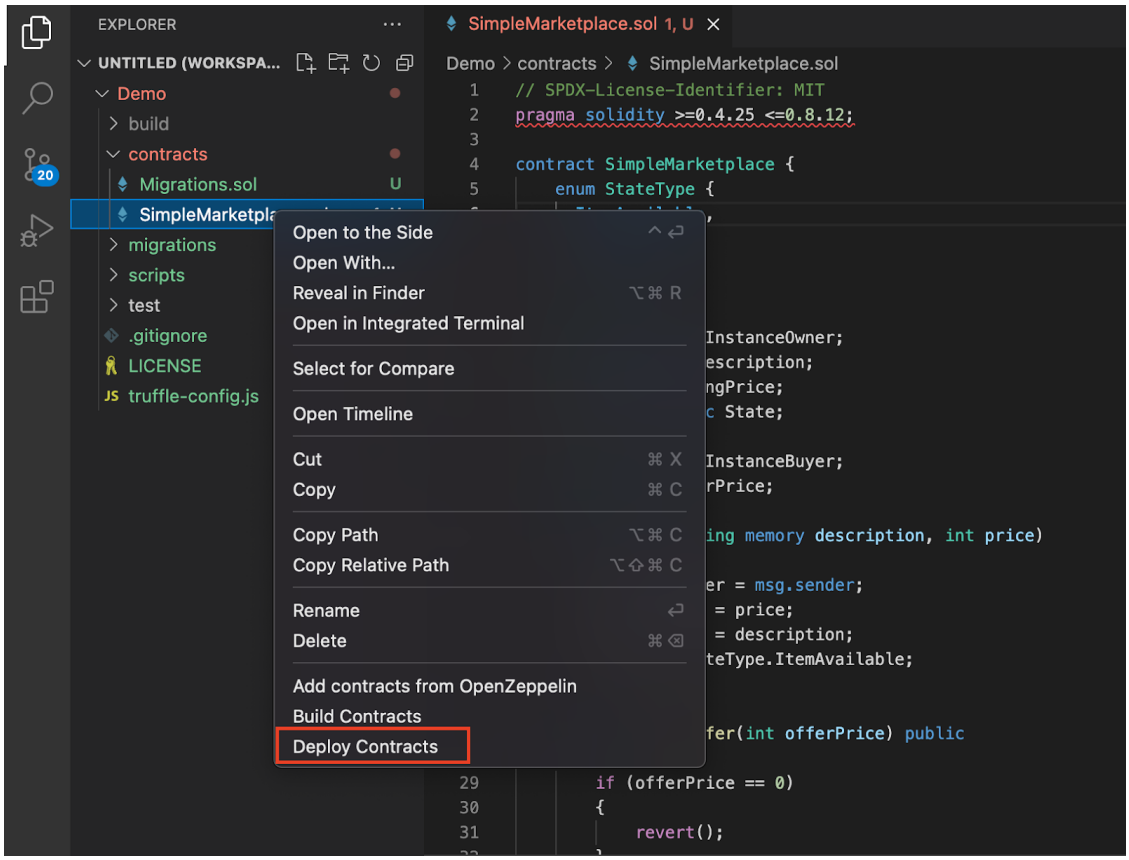



**Figure 12. Truffle VS Code Deploy Contracts**

The Ganache service allows individual developer machines to run local instances of Ethereum blockchains for development and testing and can be controlled from the Truffle IDE..

The Stratis Network Explorer feature would provide an explorer-style view similar to the existing Visual Studio Server Explorer and Cloud Explorer views that would allow developers to connect to the REST APIs of different running Stratis nodes and perform smart contract and account operations like getting the byte code of a smart contract or inspecting smart contract data stored persistently, as well as admin operations like starting, stopping, and restarting the node or specifying the blockchain network to join.

# C# Static Analysis and Visual Linting

Roslyn analyzers are used to enforce C# coding guidelines in particular domains iike security via syntactic, semantic, and other kinds of static analysis. This feature implements visual linting for C# smart contracts using Roslyn analyzers to enforce smart contract coding guidelines and to give advice on secure usage of smart contract features and patterns via static analysis of C# code. Guidelines can be given for things like recommended constructor initialization patterns, gas-efficient code, or proper access patterns for persistent storage, and which are visible in the code editor or issue pane.


Figure 13. Remix IDE flagging potential Solidity security iisue

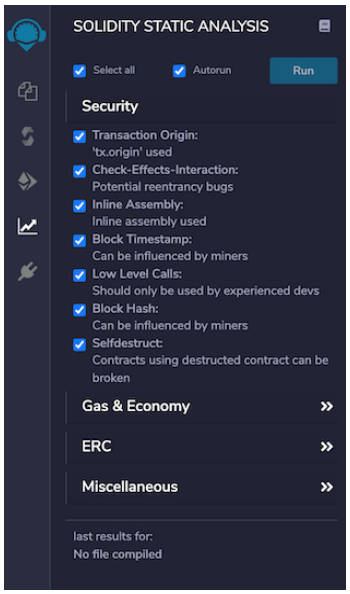



Figure 14. Configuring Remix IDE Solidity static analysis

# C# Formal Verification

Formal verification is a type of static analysis where programs are proven to have certain properties according to a formal specification. There has been a great deal of interest into formal verification of Ethereum smart contracts as a way to assess the correctness and increase the reliability of smart contracts which are high-value typically immutable applications where simple errors in design can lean to disastrous financial and reputational losses.

Silver has the ability to formally verify Stratis C# smart contracts using annotations and specifications embedded as comments. This feature will integrate the formal verification capabilities of Silver into a Visual Studio Roslyn analyzer which will allow developers to formally verify their smart contract code with immediate visual feedback on which properties of their code can't be proved by the verifier.

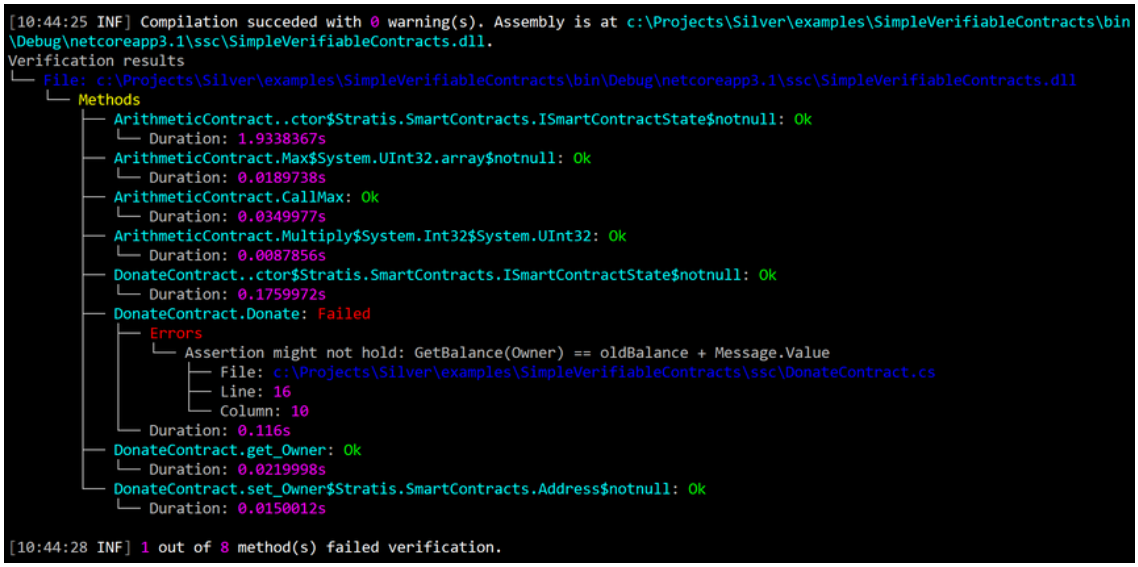



**Figure 15. Silver formal verification output**

# Review GitHub PR smart contract

Smart contracts must first be reviewed before they can be deployed to the Cirrus sidechain, usually by the developer submitting a pull request to the Cirrus smart contracts repo. This feature will allow smart contracts to be reviewed directly from Visual Studio. An “Import PR” command and dialog will be added which will fetch the content of a GitHub PR and open the smart contract source code as C# files inside Visual Studio if available, or analyze the attached smart contract bytecode. This feature will rely on what is already implemented in Silver.


**Figure 16. Reviewing GitHub PR in Silver**

# TECHNICAL OBSTACLES

The major technical obstacle to faced is the complexity of Visual Studio extension development. Extension development is still a bit of a dark art with a lot of hidden knowledge required on the best ways to accomplish certain things and maintain stability in the Visual Studio environment that isn't fully covered by the official docs. Microsoft has recognized the situation and begun work on a new extensibility model. Although the new model is the future of Visual Studio extension development, the Stratis extension will continue to use the old model as all the existing knowledge in the wild on extension development relates to the old model.

# INDUSTRY AND MARKET ANALYSIS

One of the most distinctive differences of the Stratis platform is the use of the .NET Core CLR as the smart contract VM instead of a bespoke VM like Ethereum's EVM. Stratis smart contract libraries are just regular .NET assembles and smart contract bytecode is just CIL code. C# smart contracts are compiled using the regular Roslyn C# compiler and all the tools and methods for compiling, debugging, testing, and transporting .NET code can be used for smart contract development and deployment. In this way Stratis leverages the full C# .NET toolchain for smart contract development together with advanced tools like Roslyn analyzers in a way other blockchain platforms that support C# like NEO can't with their custom smart contract VMs and compiler.

.NET has been around for more than two decades and the ability of Stratis to use the full C# .NET toolchain and to benefit from the considerable of research that has gone into strengthening C# and .NET code security through techniques like static analysis should give it a major advantage over other blockchain platforms.

In spite of this unique advantage, the Stratis platform is currently not ranked as a top enterprise blockchain platform.

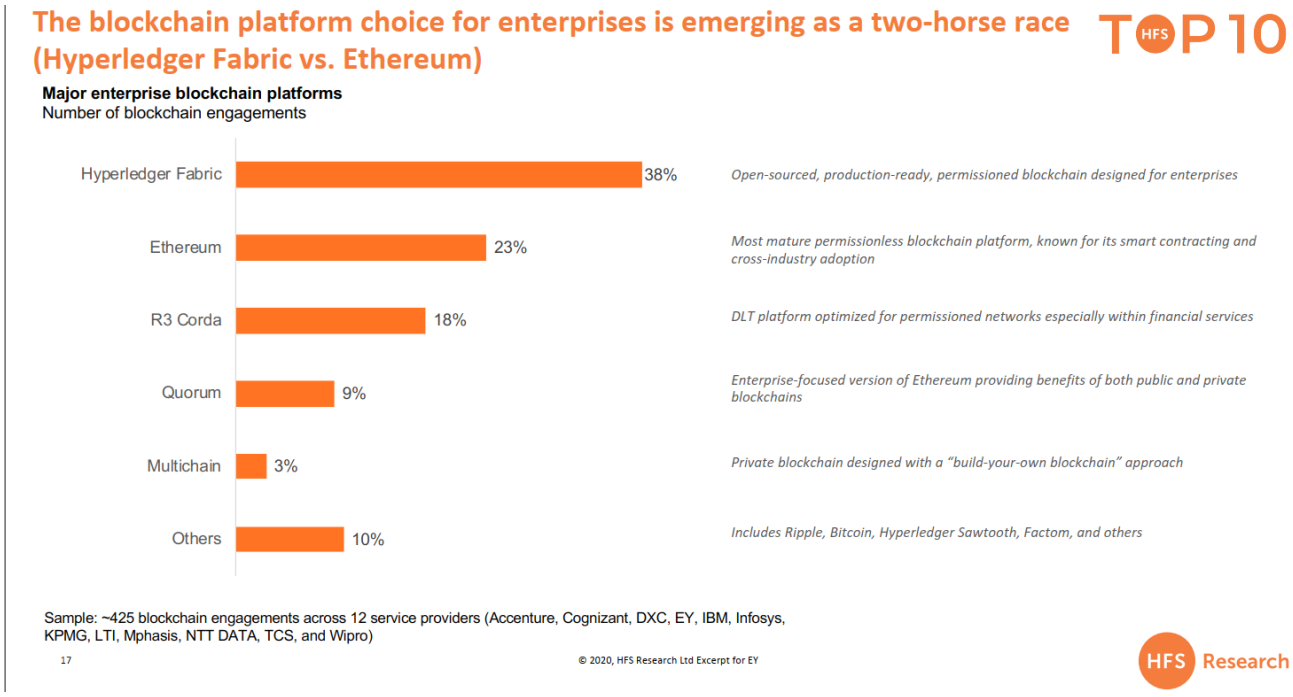



**Figure 8. From "HFS Top 10 Enterprise Blockchain Services 2020" HFS Research. April 2020.**

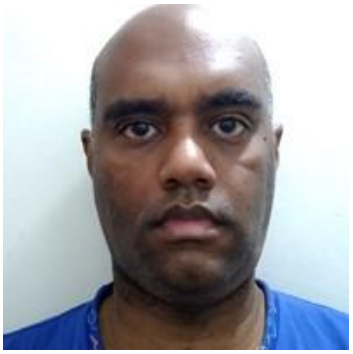
Stratis isn't even listed in Gartner's list of 53 blockchain platform products whereas NEO is.

Microsoft launched an Ethereum-based blockchain-as-a-service on Azure in 2015 but stopped offering it in September 2021. It now recommends customers adopt ConsenSys Quorum Blockchain Service. Stratis is currently the only blockchain platform built end-to-end using .NET.

Given the popularity of .NET in enterprises and the sheer number of C# developers (rivalled only by Java in the enterprise) and the accelerating pace of blockchain adoption by industries there is a huge growth opportunity for Stratis in this area.

The long-term goal of this project is to make Stratis appealing to .NET enterprise developers over other blockchain platforms by providing for smart contract development the language and tooling support they are accustomed to for enterprise development.

# TEAM BIO



Allister is a developer with more than twenty years experience specializing in open-source development on .NET. He's worked for a leading regional gold-certified Microsoft solution provider where he completed the MCSD and MCDBA certification. His most recent experience was as a contract developer for OSS Index where he was the main developer for the DevAudit security auding tool and the Audit.Net Visual Studio extension until OSS Index was acquired by Sonatype.

Allister is the developer of Silver - a static analysis and formal verification tool for Stratis smart contracts.

Allister has participated in several online hackathons and technical writing contests including the Stratis Build Hackathon and CodeProject's AI for Good and recently the Kyiv Tech Summit hackathon.

# BUDGET AND TIMELINE

The requested amount is $72 000 to cover development costs for 1 developer for 12 months of .NET and Visual Studio development

| Milestone | Tasks | Artifact | Est. Date |
|---|---|---|---|
| **1 C# Source Code Validation** | | Visual Studio Roslyn Analyzer | 01/23 |
| 1.1 | Port CIL format validation rules to Roslyn syntax rules | | |
| 1.2 | Port CIL determinism validation rules to Roslyn syntax rules | | |
| **2 C# Source Code Exploration** | | Visual Studio Extension | 03/23 |
| 2.1 | Call graph exploration tool window | | |
| 2.2 | Control-flow graph exploration tool window | | |
| 2.3 | Class + struct diagram tool window | | |
| **3 C# Disassembly + Gas Cost** | | Visual Studio Extension | 04/23 |
| 3.1 | CIL Disassembly + gas cost tool window | | |
| **4 Smart Contract Deploy and Run** | | Visual Studio Extension | 05/23 |
| 4.1 | .NET interface to node REST API | | |
| 4.2 | Deploy and Run tool window | | |
| **5 Stratis Network Explorer** | | Visual Studio Extension | 06/23 |
| 5.1 | Network Explorer tool window | | |
| 5.2 | Bookmark/organize nodes by folder | | |
| 5.3 | Nickname/organize account address | | |
| 5.4 | Decompile smart contract bytecode | | |
| | | | |

| | | | |
|---|---|---|---|
| **6 Review GitHub PR smart contract** | | **Visual Studio Extension** | 07/23 |
| 6.1 | Integrate Silver GitHub PR fetch | | |
| **7 C# Formal Verification** | | **Visual Studio Roslyn Analyzer** | 09/23 |
| 7.1 | Integrate Silver formal verification | | |
| **8 C# Static Analysis and Visual Linting** | | **Visual Studio Extension** | 11/23 |
| 8.1 | Static analysis options configuration | | |