Verificación y Validación Formal de Contratos Inteligentes para Blockchain con Alloy

Lorena Baigorria
flbaigor@unsl.edu.ar
Universidad Nacional de San Luis
San Luis - Argentina

Ana Gabriela Garis
agaris@unsl.edu.ar
Universidad Nacional de San Luis
San Luis - Argentina

Daniel Riesco
driesco@unsl.edu
Universidad Nacional de San Luis
San Luis - Argentina

Abstract

Blockchain es la base tecnológica de una nueva forma de realizar transacciones de manera segura en una red descentralizada. Dicha tecnología permite registrar la validez y el origen de los datos, y realizar transacciones de manera digital, compartida, inalterable y sin la intervención de intermediarios. Frecuentemente, las transacciones requieren de lógica automatizada. En estos casos, se vuelve necesaria la definición de contratos inteligentes, programas de computación almacenados en Blockchain que se ejecutan automáticamente cuando se cumplen condiciones predeterminadas. Los errores en contratos inteligentes pueden tener graves consecuencias, especialmente en ámbitos como finanzas descentralizadas. Una clara definición de las condiciones es esencial; sin embargo, éstas son generalmente descriptas en lenguaje natural por las partes involucradas, lo que conlleva a la ambigüedad de interpretación por parte de los programadores del contrato. Por otro lado, los errores en la programación también pueden derivar a que el contrato no se ejecute como se esperaba. La calidad del contrato inteligente podría ser mejorada si las condiciones fueran especificadas en UML con OCL, y luego transformadas al lenguaje de modelado Alloy para llevar a cabo la verificación y validación formal a través del método Model Checking. En este artículo, se describe una línea de investigación que propone un modelo para la especificación de contratos inteligentes en UML y OCL, complementado con una transformación automática a Alloy para su verificación y validación. Dicho modelo contribuye a realizar una auditoría más rigurosa de contratos inteligentes antes de despliegue en Blockchain.

Keywords

Blockchain, Contratos Inteligentes, UML, OCL, Alloy, Modelos, Model Checking

1 INTRODUCCIÓN

Las aplicaciones descentralizadas (DApps) y la tecnología Blockchain han revolucionado la manera en que se firman y se cumplen los acuerdos entre partes; ahora lo realiza un programa, sin intermediarios. Pero esa inmutabilidad alcanzada por la automatización, una vez desplegado el contrato, exige que todo funcione correctamente. Sin embargo, la ambigüedad del lenguaje natural y los errores de programación pueden llevar a que un contrato se comporte diferente a lo acordado por las partes.

Un contrato inteligente es esencialmente código almacenado en la Blockchain que se ejecuta automáticamente cuando se cumplen ciertas condiciones, como por ejemplo transferir fondos al recibir un pago. Para su desarrollo, se utilizan lenguajes específicos del dominio como Solidity [1], los cuales, sin embargo, no ofrecen garantías formales de corrección. El desarrollador es responsable de examinar el resultado de cada llamada y gestionar adecuadamente cualquier excepción que pueda surgir. No obstante, en muchos casos, los desarrolladores omiten realizar verificación que pueden provocar inconsistencias en el sistema [2]. Esta situación se agrava debido a la ambigüedad inherente al lenguaje natural, que conlleva a especificaciones poco precisas y, en consecuencia, a posibles errores en la implementación. Si a esto se suma la falta de pruebas exhaustivas, el despliegue de contratos inteligentes puede terminar en el incumplimiento de las reglas establecidas.

En este contexto, la Ingeniería Dirigida por Modelos (MDE, por sus siglas en inglés Model Driven Engineering) surge como una alternativa que propone comenzar con modelos abstractos, los cuales luego se refinan automáticamente hasta obtener el código final [3]. Lenguajes como UML y OCL [4,5] facilitan la representación estructurada de los requisitos [6]. No obstante, estos lenguajes todavía carecen de mecanismos integrados que permitan verificar formalmente propiedades complejas en escenarios específicos.

Para abordar esta necesidad, se propone recurrir a Alloy, un lenguaje formal basado en álgebra relacional que permite describir modelos y restricciones, y validar su consistencia mediante verificación formal con la ayuda de un SAT Solver. Finalmente, la técnica de Bounded Model Checking utilizada en Alloy posibilita una exploración del espacio de estados del modelo [7], lo que permitiría detectar violaciones a las restricciones antes del despliegue del contrato.

En el presente trabajo se proponen métodos, técnicas y herramientas para generar contratos inteligentes de alta calidad, utilizando UML/OCL y Alloy [8] para especificar, verificar y validar formalmente contratos inteligentes. Los lenguajes de modelado UML y OCL son ampliamente conocidos en área de la ingeniería informática. De esta forma, la especificación de los contratos con UML/OCL permitiría que las partes interesadas realicen un análisis de los acuerdos pautados. Por otro lado, Alloy es un lenguaje formal soportado por una herramienta que habilita la verificación y validación de modelos. Una transformación de UML/OCL a Alloy permitiría llevar a cabo la verificación y validación para explorar diferentes escenarios posibles y detectar inconsistencias antes de desplegar el contrato en Blockchain.

La transformación automática desde UML/OCL a Alloy es una herramienta entre el modelado tradicional y la verificación formal [9], haciendo accesible esta última a perfiles no especializados en lógica matemática. Esto permite reducir la brecha entre la facilidad del modelado y la robustez de la verificación formal de contratos inteligentes en Blockchain.

El presente trabajo contribuye a mejorar el proceso de definición de contratos inteligentes de alta calidad, con ventajas significativas como: la verificación de propiedades deseables en forma automática, la exploración de configuraciones posibles, la reducción del riesgo evitando errores que podrían causar, por ejemplo, pérdidas económicas.

El articulo está estructurado como se sigue. La sección 2 detalla los fundamentos que sustentan la propuesta. La sección 3 muestra los resultados preliminares en esta línea de investigación. Finalmente, la sección 4 explica los trabajos en desarrollo y la conclusión.

2 PROPUESTA METODOLÓGICA

El presente trabajo de investigación tiene como objetivo general definir métodos y técnicas e integrar diferentes herramientas de la ingeniería de software en forma automática para la verificación y validación de contrato inteligentes para Blockchain. La metodología planteada para llevar a cabo el trabajo de investigación consta de las etapas detalladas a continuación.

Revisión de literatura. El estudio del estado del arte en la temática abordada se centrará en tanto en el uso de UML y OCL como lenguajes para especificar contratos inteligentes, como en la aplicación de métodos y lenguajes formales para la verificación y validación en el contexto de Blockchain.

Modelado de contratos inteligentes con UML y OCL. Se definirá un método y técnicas para la especificación de diagramas de clases UML con restricciones expresadas en OCL a partir de un contrato inteligente. El método podría ser luego automatizado a través de una transformación.

Transformación de UML/OCL a Alloy. Se estudiarán enfoques y herramientas existentes, tales como [9], para la transformación automática de los contratos inteligentes definidos con UML y OCL a Alloy.

Verificación y Validación con el analizador Alloy. Se definirán empleará Bounded Model Checking para detectar inconsistencias. También se indicarán las pautas para la especificación en Alloy de propiedades deseables en contratos inteligentes para su posterior verificación, así como otras técnicas tales como el análisis de escenarios posibles que permitan establecer casos de prueba.

Generación de código de un contrato inteligente. La especificación Alloy final representa un modelo abstracto de un contrato inteligente. Siguiendo el enfoque MDA, sería posible generar contratos inteligentes implementados con diferentes lenguajes. De esta manera, se automatizará la transformación desde Alloy al lenguaje Solidity; no obstante, es posible plantear nuevas transformaciones a otros lenguajes tales como Rust. El código generado luego puede ser testeado con pruebas unitarias y de integración. Este enfoque permite auditar contratos desde etapas tempranas del desarrollo, disminuyendo la probabilidad de fallos luego del despliegue.

Validación de la propuesta. Para validar la propuesta se detallarán diversos casos de estudios. Se definirán casos ficticios simples para pruebas iniciales y otros encontrados en la literatura que reflejen situaciones reales.

3 RESULTADOS PRELIMINARES

Se ha comenzado la revisión de literatura que incluyó el estudio del uso de UML/OCL y lenguajes y métodos formales para la especificación, verificación y validación de contratos inteligentes.

3.1 Caso de estudio

Se ha observado que la temática es un área activa de investigación, particularmente en torno a la búsqueda de métodos formales y semiformales que aseguren su correcta especificación, implementación, verificación y validación antes del despliegue. Se presenta a continuación a modo de resumen diferentes líneas de investigación que abordan la temática.

Numerosos trabajos, como el de Bartolini et al. [10] proponen el uso de UML y OCL como lenguajes accesibles para ingenieros de software que no poseen experiencia formal en Blockchain. Estos enfoques permiten representar estructuras y restricciones de los contratos de forma visual y declarativa, facilitando la comprensión y reutilización de modelos. Sin embargo,

UML/OCL no posee una semántica formal completa que permita la verificación automática de propiedades.
Según Hsain et al. [11], los lenguajes específicos de dominio han sido empleados para generar contratos inteligentes a partir de abstracciones de alto nivel, en especial para aplicaciones financieras. Por ejemplo, Veschetti et al. también presentan un nuevo lenguaje de modelado, denominado SmartML, enfocado en resolver vulnerabilidades de seguridad [12].
Lenguajes formales como Alloy e Isabelle/HOL han sido aplicados para especificar contratos con precisión matemática. Alloy ha sido utilizado por Grishchenko et al. [13] para detectar errores en contratos Ethereum modelando su semántica y verificando propiedades mediante análisis exhaustivo. Isabelle/HOL ha sido propuesto para verificar contratos inteligentes en Solidity [14].
Se ha especificado un caso de estudio inicial, con el objetivo de definir un método para el modelado de contratos inteligentes con UML y OCL. El mismo caso será utilizado para posteriormente establecer la transformación de UML/OCL a Alloy, y luego llevar a cabo la Verificación y Validación con el analizador Alloy.

El Sistema de Certificación Digital de Títulos Universitarios en Blockchain UNSL-Cert [15] asegura que no se dupliquen los certificados de los estudiantes mediante una regla que prohíbe el registro de un título con el mismo nombre de carrera para un estudiante que ya tiene uno. Esto garantiza la integridad y la unicidad de los registros de títulos universitarios en la plataforma. Este sistema implementa la regla que debe cumplirse en lenguaje Solidity y las pruebas fueron realizadas luego de ser desarrollado con test unitarios y pruebas de integración, no realiza una validación y/o verificación formal.
Hasta el momento se ha abordado la definición de la regla en lenguaje natural y su formalización en UML y OCL.

Regla en Lenguaje Natural

La regla de negocio más crítica para la validez del certificado es la unicidad del título por estudiante. En términos simples, esto significa que un estudiante no puede tener dos títulos registrados para la misma carrera. Este requisito es fundamental para mantener la integridad de la base de datos de títulos, ya que evita la duplicación de credenciales para un mismo individuo.

Formalización en OCL

OCL se usa para expresar restricciones de manera formal e inequívoca, lo cual elimina ambigüedades del lenguaje natural. En el modelo de clases, esta regla se aplica sobre la clase Student y su relación con la clase Certificate. Se muestra a continuación la Restricción en OCL.

Restricción OCL: Existe un único título por estudiante

```
context Student
inv: self.certificate ->collect(c |
c.universityDegree.degreeProgramName)
 ->isUnique(degreeName | degreeName)
```

context Student: Define que la regla se aplicará a todas las instancias de la clase Student.
self.certificate: Referencia a la colección de certificados asociados a un estudiante específico (self).
->collect(c | c.universityDegree.degreeProgramName): Esta es una operación de navegación que toma cada certificado (c) de la colección y extrae el nombre de la carrera (degreeProgramName) de su grado universitario (universityDegree). El resultado es una nueva colección que solo contiene los nombres de las carreras.
->isUnique(degreeName | degreeName): Este método de la biblioteca estándar de OCL verifica que todos los elementos en la colección de nombres de carreras sean únicos. Si el método retorna false, la invariante se rompe, lo que indica un intento de registrar un título duplicado.

4 CONCLUSIONES Y TRABAJOS FUTUROS

De acuerdo a los resultados preliminares, se puede concluir que sería posible realizar la verificación formal de contratos inteligentes como clave para garantizar su correcto funcionamiento en Blockchain. Es por esto que la presente propuesta se encuentra basada en modelado UML/OCL, lenguajes ampliamente conocidos y utilizados en la ingeniería de software y la verificación y validación con Alloy, que permite un análisis exhaustivo de los términos del contrato inteligente. Se constituye así un marco de trabajo para validar que los contratos reflejen fielmente los acuerdos establecidos, mejorando la seguridad y calidad del software desplegado en entornos descentralizados.
Como trabajo futuro se tiene previsto estudiar la transformación de UML/OCL a Alloy, la Verificación y Validación con el analizador Alloy, la generación de código de un contrato inteligente y la validación de la propuesta con diferentes casos de estudios. Además, se encuentra en desarrollo patrones de transformación reutilizables para diferentes tipos de cláusulas contractuales y la verificación y validación de contratos inteligentes de casos reales (p.ej., pagos, préstamos, tokens, envíos) con el fin de validar los mismos utilizando Alloy Analyzer [8] que emplea bounded model checking.

Referencias

[1] L. Luu et al., “Making Smart Contracts Smarter,” in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16), 2016, doi: 10.1145/2976749.29783098.

[2] R. Fekih, “Formal verification of smart contracts based on model checking: An overview,” 2023. Accessed: Jun. 21, 2024. [Online]. Available:

[3] D. C. Schmidt, “Model-driven engineering,” IEEE Computer, vol. 39, no. 2, pp. 25–31, 2006, doi: 10.1109/MC.2006.5810.

https://www.researchgate.net/publication/3794013613.

[4] OMG. Unified Modeling Language (UML), version 2.5.1. 2017., https://www.omg.org/spec/UML/

[5] OMG. Object Constraint Language (OCL), version 2.4. 2017. https://www.omg.org/spec/OCL/

[6] A. Van Deursen, P. Klint, and J. Visser, “Domain-specific languages: An annotated bibliography,” SIGPLAN Not., vol. 35, no. 6, pp. 26–36, 200011.

[7] A. Garis and A. Sanchez, “Verification and Validation of Domain Specification Languages using Alloy,” in Proceedings of the XXI Congreso Argentino de Ciencias de la Computación (CACIC 2015), 2015, pp. 589-5984.

[8] Alloy 6.2.0; https://alloytools.org/

[9] A. Cunha, A. Garis, and D. Riesco, “Translating between Alloy specifications and UML class diagrams annotated with OCL,” Softw Syst Model, vol. 14, pp. 5–25, 2015, doi: 10.1007/s10270-013-0353-52.

[10] C. Bartolini, C. Palomba, and F. Lomuscio, “Smart Contract Modeling using UML and OCL for Formal Verification,” in

2021 IEEE International Conference on Software Architecture (ICSA), 2021, pp. 139–1501.

[11] Y. Hsain, N. Laaz, and S. Mbarki, “Ethereum’s Smart Contracts Construction and Development using MDE Technologies: A Review,” Procedia Computer Science, vol. 181, pp. 250–257, 20216.

[12] D. Jackson, Software Abstractions: Logic, Language, and Analysis, 2nd ed. MIT Press, 20127.

[13] I. Grishchenko, M. Maffei, and C. Schneidewind, “A Semantic Framework for the Security Analysis of Ethereum Smart Contracts,”

POPL, 20185.

[14] D. Marmsoler, A. Ahmed, and A. D. Brucker, “Secure Smart Contracts with Isabelle/Solidity,” in Software Engineering and Formal Methods (SEFM 2024), A. Madeira and A. Knapp, Eds. Springer, 20259.

[15] F. Vergés, A. Garis, and C. Tissera, “Sistema de Certificación Digital de Títulos Universitarios en Blockchain (UNSL Cert),” 2024.