SmaC: Soportando el Modelado de Contratos inteligentes

Cristian Gómez, Juan M.Vara, Francisco J. Pérez-Blanco, Esperanza Marcos

Grupo de Investigación Kybele, Universidad Rey Juan Carlos
Móstoles, Madrid (28933)
{cristian.gomez,juanmanuel.vara,francisco.perez,esperanza.marcos}@urjc.es

Resumen A pesar del interés que despierta la tecnología blockchain y los contratos inteligentes, su complejidad supone un problema que ralentiza su adopción. Con la intención de contribuir a minimizar este problema, en la última edición de estas jornadas presentábamos una propuesta metodológica y tecnológica para el uso de modelos en el ámbito de los contratos inteligentes. El objetivo de este trabajo es presentar los primeros resultados de esa propuesta.

Keywords: Model Driven Engineering · Smart Contract · Solidity

1. Contexto y Motivación

Podemos contemplar una blockchain como un sistema *peer-to-peer* en el que los nodos que la integran validan y registran transacciones en una base de datos inmutable y distribuida (el popular *ledger*) de forma consensuada [1]. Más allá de las criptomonedas, una de las aplicaciones más importantes de la tecnología blockchain es poder alojar y ejecutar contratos inteligentes.

Un contrato inteligente es un programa informático que ejecuta una serie de acciones sobre la base de datos distribuida que proporciona la blockchain, cuando se cumplen las pre-condiciones registradas en el contrato. De estas definiciones podemos intuir cuáles son las principales ventajas de este tipo de contratos: no requieren de un tercero (un intermediario) que vele por el cumplimiento de esas pre-condiciones y la ejecución de las acciones, es inmutable y es público, cualquiera en cualquier momento puede consultar su especificación que, al tratarse de código software, deja además poco espacio a la interpretación, en contraposición con los contratos tradicionales [1].

A pesar de que en la literatura encontramos casos de éxito sobre su aplicación en varios sectores, como el de la alimentación, la administración pública o la educación [10], hay un consenso generalizado respecto a la complejidad del ecosistema blockchain en general y la dificultad de establecer una translación entre las reglas de negocio y los contratos inteligentes en particular [8].

No obstante, y aunque existen varias propuestas para facilitar la interacción con estos contratos una vez que se han construido, no hay muchos trabajos que traten de aislar a potenciales usuarios de negocio de la complejidad que entraña construirlos o, simplemente utilizarlos [9].

Una de las aproximaciones adoptadas por algunos de estos trabajos es precisamente el uso de modelos. Por ejemplo, Lorikeet [11] o Caterpillar [6] son motores de ejecución de BPMN (Business Process Model Notation) capaces de traducir (parcialmente) el comportamiento recogido en un modelo BPMN a varios contratos inteligentes.

Mientras que ADICO-Solidity [2] y SmaCoNat [9] proporcionan soluciones basadas en un DSL (Domain Specific Language) más cercano al lenguaje natural que Solidity para la especificación de un *smart contract*. Sin embargo, aún se puede mejorar en aspectos como la identificación de buenas prácticas y la construcción de herramientas para reducir la barrera de entrada al uso de *smart contracts*. Otra reseña destacable es que dichas especificaciones de *smart contracts* realizadas con esta herramienta no son compilables a bytecode, por lo que es necesario el uso de una segunda herramienta que realice este propósito [8].

2. Estado actual de la propuesta

En este contexto, en un trabajo previo introducíamos una propuesta dirigida a facilitar el desarrollo y uso de contratos inteligentes aplicando técnicas de Ingeniería Dirigida por Modelos [4]. El propósito de este trabajo es introducir uno de los primeros resultados de esa propuesta, `SmaC` [1], un entorno dirigido por modelos para el desarrollo de contratos inteligentes.

En la actualidad `SmaC` es un DSL textual desarrollado con Xtext[2] que permite codificar el contrato inteligente en Solidity [3] y disponer del modelo EMF correspondiente, tal y como ilustra la Figura 1: la parte izquierda muestra la codificación con `SmaC` de un contrato inteligente tal y como se incluye en la documentación de Solidity[4] y la parte derecha el modelo EMF correspondiente.

Más allá de las opciones que supone la posibilidad de disponer de modelos de contratos, `SmaC` soporta una serie de funcionalidades que facilitan la codificación de contratos con Solidity:

- Impone un patrón estructural a la hora de codificar el contrato que obliga a especificar los componentes del contrato en un determinado orden. El resultado es un código más legible y comprensible. Esta necesidad era señalada por varios de los trabajos existentes en el área [9].
- Exige cierto control de gas (coste/tarifa de realizar una operación) previo a las acciones que se ejecuten dentro de bucles, de manera que se evitan brechas de seguridad como bucles infinitos.
- Proporciona un conjunto de tipos de datos predefinidos que mapea varios conceptos identificados como relevantes hasta el momento (como `User` o `Company`) que han sido directamente integrados en la gramática definida.

[1] https://github.com/KybeleGroup/SmaC
[2] https://www.eclipse.org/Xtext/
[3] https://solidity-es.readthedocs.io/es/latest
[4] https://docs.soliditylang.org/en/v0.5.13/solidity-by-example.html#safe-remote-purchase

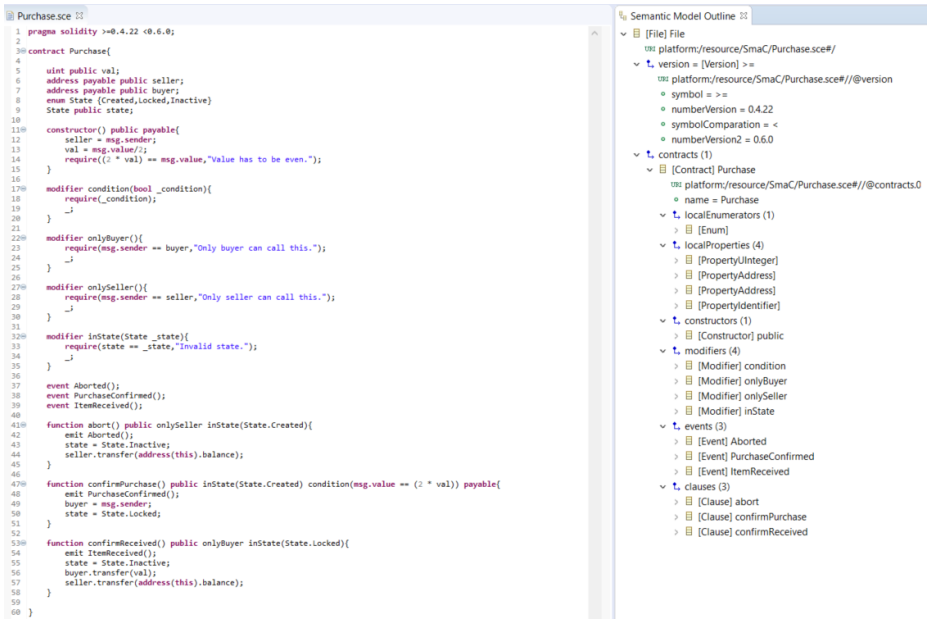

Figura 1. Contrato inteligente codificado con `SmaC` y modelo EMF correspondiente.

Ambos tipos,tienen una relación de correspondencia directa con los conceptos Actor y MarketSegment de e^{3}Value, una notación notación gráfica para representar y analizar redes de intercambio de valor [5]. Esto simplifica la trasposición de los contratos a modelos e^{3}Value y viceversa.

· Incorpora varias facilidades habituales en entornos de programación, como autocompletado de código, validación o *quick fixes*.

3. Conclusiones y Trabajo futuro

Más allá de las facilidades que como IDE (Integrated Development Environment) para Solidity proporciona `SmaC` , lo realmente destacable es que, una vez que disponemos del modelo del contrato, podemos utilizar tecnología basada en modelos para establecer correspondencias entre el contrato y cualquier otro tipo de modelo, como modelos de negocio (destacando los modelos e^{3}Value) o proceso; validar su especificación; simular su ejecución y/o estimar el coste de ejecutarlo; generar cualquier tipo de informe, etc.

Si bien algunos trabajos han explorado acercar el uso de contratos inteligentes a los profesionales de negocio, mediante la relación entre BPMN y los contratos inteligentes con este mismo objetivo, se ha comprobado que la brecha entre ambos es demasiado grande como para posibilitar una traducción directa [7].

Nosotros sin embargo, hemos descubierto que e^{3}value se ajusta mucho mejor a la naturaleza de un contrato inteligente. Así, en [3] mostramos cómo pueden

utilizarse los modelos de valor para facilitar el uso de contratos inteligentes en los servicios proporcionados por la administración electrónica.

A modo de conclusión, actualmente trabajamos en el desarrollo de una sintaxis gráfica basada en bloques (tipo Scratch) para `SmaC` , la construcción de puentes tecnológicos mediante principios, técnicas y soluciones de la Ingeniería Dirigida por Modelos que exploten las relaciones con otras notaciones de modelado, como e^{3}value, y la validación experimental de la propuesta.

Agradecimientos. Este trabajo ha sido financiado parcialmente por el Gobierno de la Comunidad de Madrid, mediante el proyecto FORTE-CM (S2018/TCS-4314) y por el Ministerio de Economía y Empresa Español, mediante el proyecto MADRID (TIN2017-88557-R).

Referencias

1. Bartoletti, M., Pompianu, L.: An empirical analysis of smart contracts: platforms, applications, and design patterns. In: International conference on financial cryptography and data security. pp. 494–509. Springer (2017)
2. Frantz, C.K., Nowostawski, M.: From institutions to code: Towards automated generation of smart contracts. In: 2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS* W). pp. 210–215. IEEE (2016)
3. Gómez, C., Blanco, F.J.P., Vara, J.M., de Castro, V., Marcos, E.: Design and development of smart contracts for e-government through value and business process modeling. In: 54th Hawaii International Conference on System Sciences, HICSS 2021, Kauai, Hawaii, USA, January 5, 2021. pp. 1–10. ScholarSpace (2021), http://hdl.handle.net/10125/70867
4. Gómez, C., Vara, J.M., Blanco, F.J.P., Marcos, E.: Una propuesta para soportar la especificación a alto nivel de contratos inteligentes. In: JISBD2019. SISTEDES (2019), http://hdl.handle.net/11705/JISBD/2019/027
5. Gordijn, J., Akkermans, H.: Designing and evaluating e-business models. IEEE intelligent Systems **16**(4), 11–17 (2001)
6. López-Pintado, O., García-Bañuelos, L., Dumas, M., Weber, I., Ponomarev, A.: Caterpillar: A business process execution engine on the ethereum blockchain. Software: Practice and Experience **49**(7), 1162–1193 (2019)
7. Mavridou, A., Laszka, A.: Designing secure ethereum smart contracts: A finite state machine based approach. In: International Conference on Financial Cryptography and Data Security. pp. 523–540. Springer (2018)
8. Mik, E.: Smart contracts: terminology, technical limitations and real world complexity. Law, Innovation and Technology **9**(2), 269–300 (2017)
9. Regnath, E., Steinhorst, S.: Smaconat: Smart contracts in natural language. In: 2018 Forum on Specification & Design Languages (FDL). pp. 5–16. IEEE (2018)
10. Reyna, A., Martín, C., Chen, J., Soler, E., Díaz, M.: On blockchain and its integration with iot. challenges and opportunities. Future generation computer systems **88**, 173–190 (2018)
11. Tran, A.B., Lu, Q., Weber, I.: Lorikeet: A model-driven engineering tool for blockchain-based business process execution and asset management. In: BPM (Dissertation/Demos/Industry). pp. 56–60 (2018)