SmaCly: un lenguaje de bloques para trabajar con 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 y curva de aprendizaje 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 uno de los componentes tecnológicos que hemos desarrollado para implementar esta propuesta: un lenguaje de bloques para la especificación y representación gráfica de contratos inteligentes.

Keywords: Model Driven Engineering · Smart Contract · Solidity · Blockly

1. Contexto y Motivación

Una *blockchain* es básicamente una red *peer-to-peer* que proporciona un *ledger* o registro distribuido del que todos los nodos que la conforman disponen de una copia. Estos nodos se encargan de verificar y registrar transacciones mediante técnicas criptográficas que ayudan a proporcionar altos niveles de seguridad e integridad de la información registrada [11]. Además de servir de base tecnológica sobre la que operan los populares criptoactivos, una de las funcionalidades más destacadas de estas redes es la de alojar y ejecutar *smart contracts* o contratos inteligentes.

Un *smart contract* es un programa informático alojado en una *blockchain* cuyo funcionamiento es relativamente básico: si las pre-condiciones asociadas a las cláusulas del contrato se cumplen, se ejecuta automáticamente la lógica recogida en el *cuerpo* del contrato. De este modo, una de las principales ventajas derivadas del uso de contratos inteligentes en procesos cotidianos es la posibilidad de prescindir de terceros que actúan como intermediarios, con el consiguiente ahorro de tiempos, monetario, etc. [4]. Además, al ser código software, un contrato inteligente está completamente exento de posibles interpretaciones (habituales en los contratos tradicionales). La accesibilidad por parte de todos los nodos para acceder a este tipo de contratos favorece la legibilidad de los mismos y la transparencia de las transacciones que se llevan a cabo en el ecosistema[2].

A pesar de estas ventajas, existen aún grandes barreras de entrada, como su curva de aprendizaje o la dificultad de realizar una traslación de reglas de negocio a contratos inteligentes, que de algún modo limitan la adopción de esta tecnología [1].

La IDM (Ingeniería Dirigida por Modelos) es una de las aproximaciones que se ha utilizado hasta la fecha para lidiar con este tipo de problemas. Existen varias propuestas como [7] o [10] que permiten extrapolar parcialmente las ideas recogidas en un modelo BPMN (Business Process Model Notation) a un *smart contract*. Otras propuestas, como `ADICO-Solidity` [3] y `SmaCoNat` [9], ofrecen soluciones basadas en el uso de un DSL más cercano al lenguaje natural que Solidity (un lenguaje habitual para su codificación) para la especificación de contratos inteligentes. Sin embargo, estas soluciones dejan aún mucho espacio de mejora para solventar el problema de adopción de los *smart contracts* como solución tecnológica [8].

2. SmaCly

En este contexto, en trabajos previos introducíamos una propuesta dirigida a facilitar el desarrollo y uso de contratos inteligentes aplicando técnicas de IDM y uno de los primeros resultados de esa propuesta, `SmaC` [1], un entorno dirigido por modelos para el desarrollo de contratos inteligentes [5]. En este trabajo, presentamos `SmaCly` [2], un componente más de esta propuesta que proporciona un lenguaje de bloques para soportar la especificación gráfica de contratos inteligentes.

En la actualidad, `SmaCly` es una sintaxis gráfica desarrollada con Blockly[3], una librería JavaScript de Google para la definición de lenguajes de bloques, similar a Scratch. Así, `SmaCly` permite especificar gráficamente un contrato inteligente en Solidity [4] mediante el uso de bloques (útil para desarrolladores nóveles) y ese contrato puede convertirse automáticamente en un modelo `SmaC` y viceversa: una transformación Xtend permite exportar modelos `SmaC` a un formato .xml legible por `SmaCly` y por tanto importarlos en la herramienta. La Figura 1 ilustra esta funcionalidad.

Más allá de las ventajas que se derivan de la posibilidad de disponer de estos puentes tecnológicos entre el espacio de los modelos y el de las visualizaciones gráficas basadas en bloques, `SmaCly` soporta una serie de funcionalidades adicionales que facilitan la codificación de contratos en Solidity:

- Ofrece un control de anexión de bloques, lo que garantiza que se cumpla con un patrón estructural y habilita un control de errores a la hora de definir el contrato.
- Incorpora cuadros de diálogo para ayudar al usuario a interconectar los diferentes bloques que representan elementos Solidity.

[1] https://github.com/KybeleGroup/SmaC
[2] https://github.com/KybeleResearch/SmaC/tree/main/SmaCly
[3] https://developers.google.com/blockly/
[4] https://solidity-es.readthedocs.io/es/latest

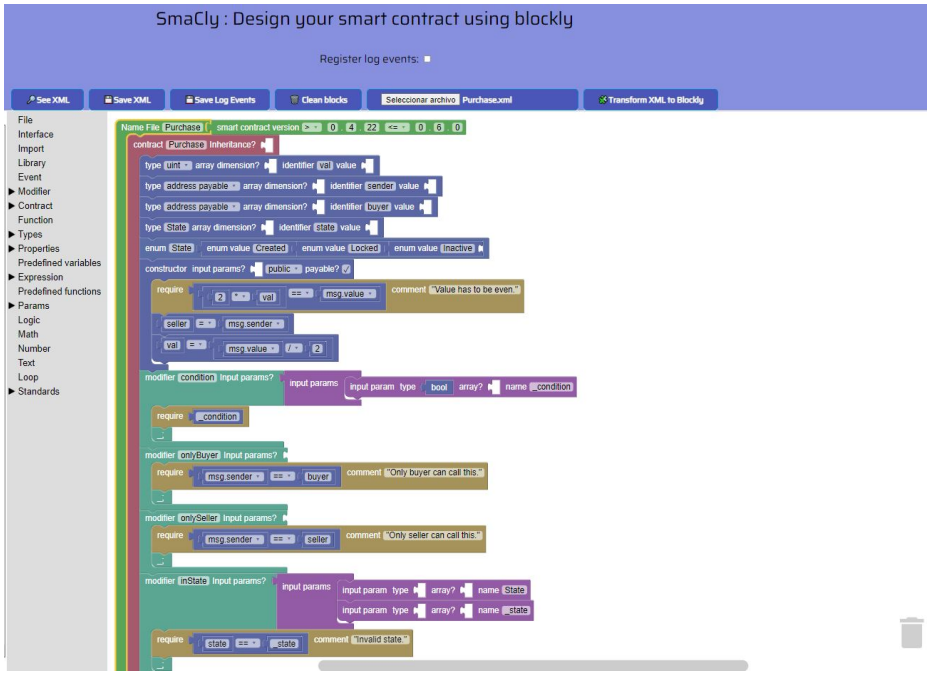

Figura 1. Visualización de un modelo `SmaC` con `SmaCly`

- A pesar de que `SmaCly` se encuentra integrado dentro de INNoVaServ, no es necesario utilizarla desde Eclipse, ya que al tratarse de una aplicación web, puede correr en cualquier navegador, lo que asegura la máxima portabilidad.
- Incorpora un conjunto de contratos predefinidos, siguiendo estándares ERC de Ethereum como: 20, 223, 721 y 771. El usuario puede partir de estos modelos y codificar (con bloques) la lógica de las funciones a ejecutar por el contrato.
- Soporta la descarga de los logs que recogen las acciones que va realizando el desarrollador durante la especificación del contrato. Estos logs pueden ser analizados (nuestra intención es hacerlo mediante minería de procesos [12]) para identificar dificultades, malas prácticas o errores habituales.

3. Próximos Pasos

Actualmente hemos incidido en el desarrollo de herramientas que, mediante el uso de modelos, faciliten el proceso de especificación y codificación de contratos inteligentes. Sin embargo, consideramos que es preciso acercarla a los profesionales de las áreas de negocio.

En nuestro afán por alcanzar este objetivo, hemos analizado que `e`^{3}`value` [6], un lenguaje de modelado para representar redes de valor que permite, por ejemplo, representar los intercambios de valor entre actores que se dan en la

provisión de servicios, se ajusta mucho mejor a la naturaleza de un contrato inteligente que otras notaciones como BPMN.

En [4] ilustrábamos esta relación de correspondencia, relacionando los elementos de la notación `e`^{3}`value` que representaba la provisión de un servicio electrónico por parte de la administración pública y los elementos de un contrato inteligente.

Actualmente trabajamos en la construcción de puentes tecnológicos que, mediante principios, técnicas y soluciones de IDM, exploten estas relaciones de correspondencia y en la validación experimental de toda la propuesta.

Agradecimientos. Este trabajo ha sido financiado parcialmente por el MilCIN a través del proyecto SerDigital (PID2020-117244RB-I00) y por el programa de contratación predoctoral para personal en formación en departamentos de la Universidad Rey Juan Carlos (C1PREDOC2020), gracias a la adjudicación de la plaza (PREDOC20-019).

Referencias

1. Alharby, M., Van Moorsel, A.: A systematic mapping study on current research topics in smart contracts. Available at SSRN 3876872 (2017)
2. 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)
3. 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)
4. 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)
5. Gómez, C., Vara, J.M., Blanco, F.J.P., Marcos, E.: Smac: Soportando el modelado de contratos inteligentes (2021), http://hdl.handle.net/11705/JISBD/2021/045
6. Gordijn, J.: E-business value modelling using the e3-value ontology. In: Value creation from e-business models, pp. 98–127. Elsevier (2004)
7. 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)
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. 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)
11. Underwood, S.: Blockchain beyond bitcoin. Communications of the ACM **59**(11), 15–17 (2016)
12. Van Der Aalst, W.: Process mining: Overview and opportunities. ACM Transactions on Management Information Systems (TMIS) **3**(2), 1–17 (2012)