

Automatizando el paso de Solidity a e³value: un puente tecnológico para facilitar la comprensión de contratos inteligentes

Cristian Gómez-Macías¹, Juan Manuel Vara Mesa¹, Francisco Javier Pérez-Blanco¹, David Granada Mejía¹, Rubén De La Fuente Lillo¹

Grupo de Investigación CommIT, Universidad Rey Juan Carlos
Móstoles, Comunidad de Madrid, España
{cristian.gomez, juanmanuel.vara, francisco.perez, david.granada}@urjc.es
r.delafuenteli@alumnos.urjc.es

Resumen Este trabajo presenta un puente tecnológico unidireccional que automatiza la transformación de contratos inteligentes escritos en Solidity en modelos e³value. Con el fin de facilitar la comprensión e interpretación de los contratos, la solución eleva el nivel de abstracción al traducir la lógica técnica del contrato en una representación visual centrada en los intercambios de valor que regula el contrato, y se complementa con un asistente visual que recoge información adicional. Un caso de estudio sirve para mostrar que el enfoque genera modelos correctos y completos, útiles para interpretar el comportamiento del contrato, contribuyendo así a mejorar la comunicación entre perfiles técnicos y no técnicos y a apoyar tareas de análisis y validación.

Keywords: Smart Contract · e³value · Model Driven Engineering · Business Modeling

1. Introducción

Es evidente que la combinación de blockchain y contratos inteligentes puede posibilitar la ejecución de acuerdos y operaciones sin intermediarios, lo que tendría un impacto positivo directo en la eficiencia, la transparencia y la trazabilidad en múltiples sectores [11]. Sin embargo, este potencial no se traduce en los niveles de adopción esperados. En parte, esto puede deberse a que cuando la lógica empresarial se codifica en un lenguaje como Solidity, resulta casi imposible para perfiles no técnicos revisar o validar esa lógica [12,14].

Esta situación amplifica una brecha habitual en los proyectos basados en blockchain y contratos inteligentes: el diseño y la supervisión del servicio suelen recaer en analistas, gestores o responsables de procesos, mientras que la implementación y el despliegue dependen de especialistas técnicos. Como consecuencia, puede aparecer un desajuste entre la intención empresarial y su materialización en el contrato, lo que dificulta la incorporación de la tecnología blockchain en organizaciones tradicionales, precisamente en escenarios donde la claridad y la trazabilidad resultan fundamentales [10].

Para mitigar esta dificultad, en este trabajo se ha desarrollado un puente tecnológico unidireccional que permite transformar los contratos escritos en Solidity en modelos e³value [8]. Estos modelos permiten representar redes de intercambio de valor a un alto nivel de abstracción, haciendo así accesible la lógica de los contratos a los perfiles no técnicos y facilitando la comunicación entre estos y los desarrolladores.

El resto del trabajo se estructura como sigue: la sección 2 presenta brevemente los conceptos en los que se basa esta propuesta. La sección 3 describe la propuesta con detalle, incluidas las correspondencias y el flujo de transformación desde el AST (Abstract Syntax Tree) del contrato hasta la generación del modelo visual. La sección 4 presenta la validación de la propuesta y discute los principales resultados. La sección 5 revisa los trabajos relacionados y, por último, la sección 6 resume las principales contribuciones y las líneas de trabajo futuro.

2. Contexto

El objetivo de este trabajo es desarrollar un puente tecnológico que facilita el paso de contratos inteligentes codificados con Solidity a modelos de negocio expresados en la notación e³value. En esta sección se presentan brevemente ambos conceptos y las tecnologías sobre las que se sustenta la propuesta.

2.1. Notación e³value

e³value es una notación para la elaboración de modelos de negocio que se centra en representar los intercambios de valor que tienen lugar entre los actores que interactúan en la provisión de un servicio o similar [8]. La Figura 1 muestra un modelo e³value que representa los intercambios de valor que se dan en un servicio de distribución digital de contenido multimedia o *streaming*. Los principales elementos del modelo son:

- **Actor:** los participantes que interactúan, intercambiando objetos de valor. Si fuera una asociación conjunta de varios de ellos, se representa con un elemento **Market Segment**, como los *Listeners* o los *Advertisers* del ejemplo.
- **Value Object:** aquello que se intercambia entre los actores y que tiene valor desde la perspectiva del negocio. Debido a su naturaleza, puede ser representado de diferentes formas: un servicio, tokens, un derecho, un bien tangible, etc. En el ejemplo, los suscriptores o *Listeners* abonan una cuota de suscripción (*subscription fee*) y a cambio reciben acceso a la oferta musical de la plataforma (*music access*).
- **Value Interface:** el conjunto de puertos que encapsulan lo que un actor ofrece/recibe en un intercambio de valor. Estas interfaces permiten expresar que los intercambios ocurren como parte de una “oferta” o “contraprestación”.
- **Value Port:** cada punto de entrada o salida a través del cual se entrega o recibe un objeto de valor. En el ejemplo cada actor tiene un único *value interface* y cada uno incluye un *value port* de entrada y uno de salida.

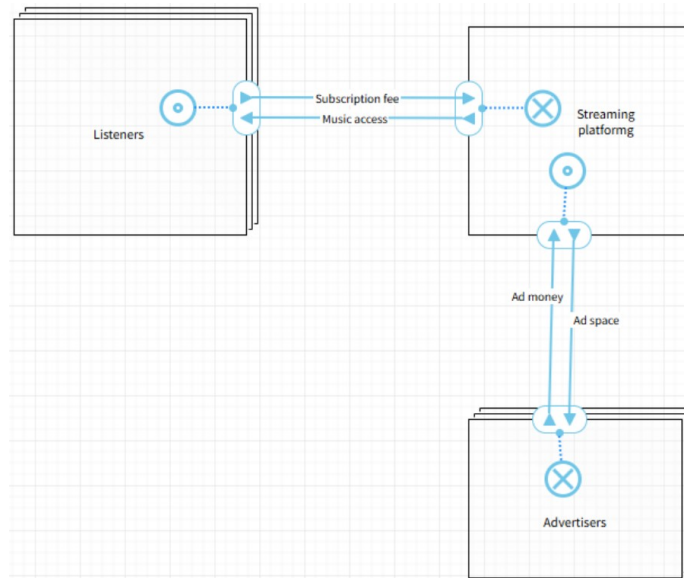


Figura 1. Modelo e³value para un servicio de streaming.

- **Value Exchange:** la relación que conecta dos puertos y representa el intercambio de un objeto de valor entre actores. Se visualiza como una línea con el nombre del objeto de valor que circula por esa conexión.
- **Start/End Stimulus:** permiten delimitar el inicio y fin de un escenario de valor, aportando contexto sobre qué eventos disparan el intercambio.

Por su naturaleza, el modelo e³value prioriza la perspectiva *económica* del sistema a implementar. Esta naturaleza es especialmente útil cuando se pretende interpretar un contrato inteligente desde el punto de vista de negocio: el contrato implementa reglas y transacciones, y el modelo e³value permite ofrecer una lectura centrada en el intercambio de valor entre las partes que participan del contrato: qué intercambian los actores, con quién y bajo qué relaciones de dependencia, etc. Esto facilita analizar la red de valor y la viabilidad de esta, lo que hace que la notación resulte especialmente útil para alinear la lógica económica del modelo de negocio con los acuerdos y compromisos entre partes. Una vista del metamodelo de la notación e³value puede consultarse en el repositorio¹. En definitiva, el uso de esta notación facilita la comunicación y el análisis en fases tempranas por parte de los profesionales de negocio.

2.2. Contratos inteligentes

Un contrato inteligente no es más que un programa alojado en una red blockchain, que se ejecuta automáticamente cuando se dan las condiciones preestable-

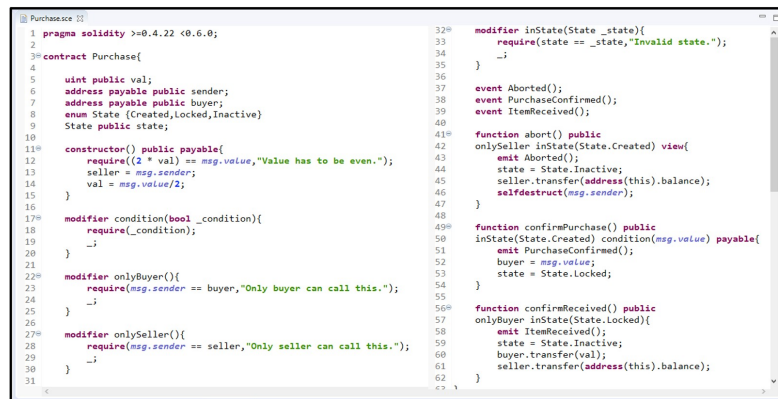
¹ <https://github.com/CommITURJC/SoliditytoE3ValueConverter/blob/main/metamodelo/readme.md>

cidas en el mismo [4]. A pesar de las ventajas que ofrecen desde el punto de vista operacional, su complejidad complica la comprensión no sólo para perfiles no técnicos, sino incluso para desarrolladores, lo que ha repercutido negativamente a sus niveles de adopción [2,16].

Solidity es el lenguaje de programación más utilizado para la especificación de contratos [3]. A continuación, se enumeran los principales componentes del lenguaje en relación con la propuesta que se presenta:

- Las variables de tipo **address** y las construcciones relacionadas con roles se corresponden con los posibles actores que interactúan con/en el contrato.
- Las **funciones** representan puntos de interacción con el contrato y se consideran candidatas para modelar interfaces de valor, ya que articulan operaciones observables desde el exterior.
- Los **eventos** (event / emit) permiten registrar acciones significativas y, en muchos contratos, sirven como traza de operaciones que tienen interpretación económica (p.e., emisión de tokens, transferencias, recompensas).
- Las **operaciones de transferencia** (por ejemplo, llamadas tipo transfer, transferFrom, mint, etc.) permiten detectar intercambios de valor, incluso cuando no existe un evento explícito asociado al intercambio.

Una vista de los anteriores componentes puede apreciarse en la siguiente Figura 2 que representa un smart contract de compra remota segura:



```
1 pragma solidity >=0.4.22 <0.6.0;
2
3 contract Purchase{
4
5     uint public val;
6     address payable public sender;
7     address payable public buyer;
8     enum State {Created,Locked,Inactive}
9     State public state;
10
11 constructor() public payable{
12     require(2 * val) == msg.value, "Value has to be even.";
13     seller = msg.sender;
14     val = msg.value/2;
15 }
16
17 modifier condition(bool _condition){
18     require(_condition);
19     _;
20 }
21
22 modifier onlyBuyer(){
23     require(msg.sender == buyer, "Only buyer can call this.");
24     _;
25 }
26
27 modifier onlySeller(){
28     require(msg.sender == seller, "Only seller can call this.");
29     _;
30 }
31
32 modifier inState(State _state){
33     require(state == _state, "Invalid state.");
34     _;
35 }
36
37 event Aborted();
38 event PurchaseConfirmed();
39 event ItemReceived();
40
41 function abort() public
42 onlySeller inState(State.Created) view{
43     emit Aborted();
44     state = State.Inactive;
45     seller.transfer(address(this).balance);
46     selfdestruct(msg.sender);
47 }
48
49 function confirmPurchase() public
50 inState(State.Created) condition(msg.value) payable{
51     emit PurchaseConfirmed();
52     buyer = msg.value;
53     state = State.Locked;
54 }
55
56 function confirmReceived() public
57 onlyBuyer inState(State.Locked){
58     emit ItemReceived();
59     state = State.Inactive;
60     buyer.transfer(val);
61     seller.transfer(address(this).balance);
62 }
```

Figura 2. Smart contract de compra remota segura.

Sin embargo, la lógica de negocio no siempre se explicita en el código. Por ejemplo, un evento puede ser relevante para el escenario de valor, pero el contrato no indica si debe interpretarse como estímulo de inicio o de fin. Del mismo modo, ciertos parámetros numéricos o estructuras de datos carecen de significado económico sin un contexto adicional. Por este motivo, además del análisis automático del contrato, la propuesta que se presenta contempla mecanismos

auxiliares para que el usuario ayude a recopilar la información necesaria. Para ello, se utiliza un asistente visual durante el proceso de transformación.

3. Desarrollo de la propuesta

Esta sección presenta el puente tecnológico que permite conectar contratos inteligentes y modelos e³value. En primer lugar, se describe el diseño de la solución. A continuación, se presenta el análisis de correspondencias entre los dos espacios técnicos que conecta el puente tecnológico, para pasar a comentar algunos detalles de implementación, y finalizar con una descripción del proceso de transformación mediante el uso de los componentes anteriormente descritos.

3.1. Vista general de la solución

La Figura 3 proporciona una vista general de la propuesta que se presenta en este trabajo. Como ilustra la Figura, esta solución toma como entrada un contrato inteligente codificado en lenguaje Solidity con cualquiera de los IDEs disponibles al efecto, como Remix² o SmaC [7].

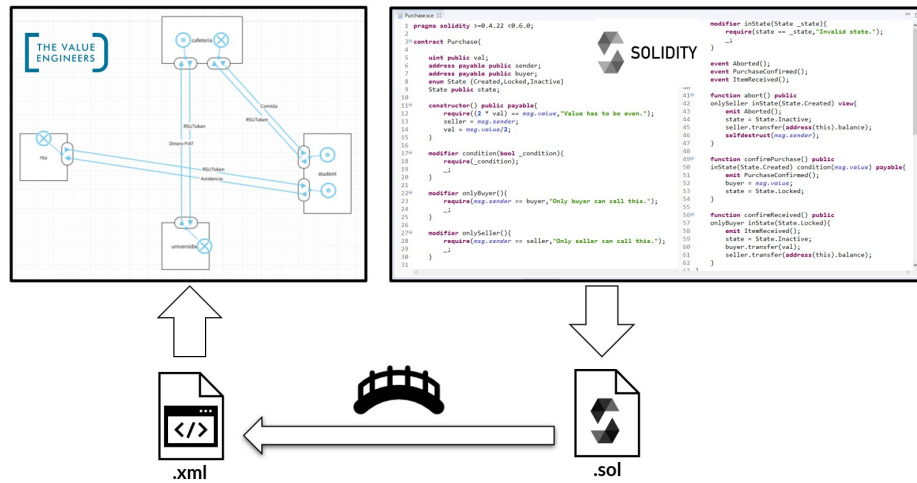


Figura 3. Vista general de la solución.

A partir del contrato, la solución genera un modelo e³value que puede editarse con cualquiera de los editores web existentes para ello. Antes de entrar en el detalle del puente tecnológico unidireccional que implementa esta solución, es necesario identificar las correspondencias (semánticas) entre los elementos de Solidity y los de la notación e³value. Este análisis no solo guiará el diseño e

² <https://remix.ethereum.org/>

implementación de las transformaciones, sino que además permite maximizar su eficacia, entendida como la preservación del mayor grado de información posible al pasar de un espacio técnico al otro.

3.2. Análisis de correspondencias

Frente a otras notaciones, como BPMN, más centradas en el flujo de actividades, e³value se centra en el flujo de intercambios de valor entre las entidades participantes en un determinado contexto, y por eso se adecúa mejor a la hora de buscar una abstracción con la que representar conceptualmente un contrato inteligente, en términos más comprensibles para cualquier perfil, y especialmente los no técnicos.

En otros trabajos [6,9], a la hora de construir el puente tecnológico inverso entre ambos dominios ya identificamos las principales correspondencias entre la notación e³value y los contratos Solidity. Estas correspondencias se resumen en la Tabla 1.

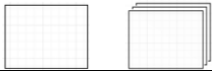



Contrato inteligente	Elemento e ³ value	Representación gráfica e ³ value
Tipo de dato address	Actor/Market Segment	
Smart contract	Value Interface	
Funciones	Value port(s) & Value exchange	
Criptoactivos, servicios, información, bien tangible, etc.	Value object	[Tokens, servicios, € ...]
Eventos	Start/End Stimulus	

Tabla 1. Correspondencias entre contratos Solidity y modelos e³value

En la notación e³value, las entidades **Actor** y **Market Segment** constituyen los participantes del intercambio de valor. En el contexto de los contratos inteligentes, estos participantes pueden representarse mediante **direcciones** (tipo de dato address) en Solidity, que identifica a los sujetos capaces de invocar funciones e interactuar con el contrato.

Desde esta perspectiva, los componentes **Value Port** y sus **Value Exchange** asociados pueden equipararse a los mecanismos funcionales del contrato encargados de gestionar la transferencia del **Value Object** entre las partes: esto es, las operaciones que permiten la entrega o la recepción del activo o prestación acordados en el intercambio.

El **Value Object** designa, por tanto, el elemento negociado dentro del contrato inteligente y puede ser representado de diferentes formas según su naturaleza, ya sea como un token, un servicio o simple información. A su vez, el elemento

Value Interface, que organiza y habilita el intercambio al agrupar Value Port y Value Exchange, puede corresponderse con el propio contrato inteligente, entendido como el artefacto que encapsula la lógica y las funciones necesarias para ejecutar la circulación del objeto de valor entre los participantes.

Por último, los elementos **Start** y **End Stimulus**, representan las notificaciones cuando se comienza o se finaliza un escenario de valor, correspondiéndose estas notificaciones con los eventos definidos en el contrato inteligente y que se ejecutan cuando se ejecuta la lógica del contrato.

El puente tecnológico presentado en este trabajo se sustenta sobre estas correspondencias.

3.3. Desarrollo del puente tecnológico

Esta sección describe brevemente algunos detalles del puente tecnológico construido en este trabajo, que permite generar un modelo e³value a partir de un contrato inteligente.

En el contexto de este trabajo, el contrato Solidity se trata como un artefacto del que se obtiene un modelo origen, a partir del cual se genera un modelo de mayor nivel de abstracción e³value interpretable por negocio. La solución construida implementa el proceso de transformación por etapas representado en la Figura 4.

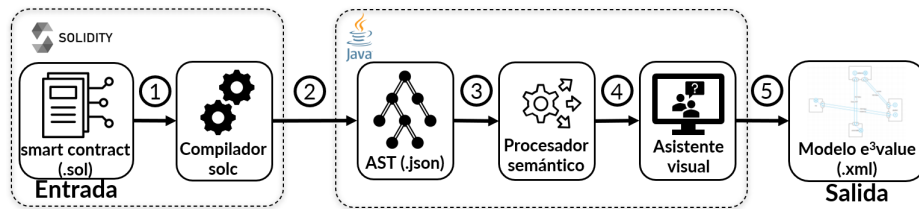


Figura 4. Proceso de transformación

En este proceso se distinguen cinco pasos o etapas. En la primera etapa se toma como entrada el contrato Solidity con extensión `.sol` ①. A continuación, dicho contrato se procesa con el compilador `solc` ②, para obtener su AST en formato JSON (JavaScript Object Notation), que es utilizado como representación estructurada del código sobre la que se aplican reglas de extracción ③. El resultado se utiliza como entrada de un procesador semántico ④ que construye un modelo intermedio para describir el escenario de valor en forma de actores, eventos, operaciones y objetos de valor. En esta misma etapa se integra el asistente visual realizado con la biblioteca gráfica de Java Swing³ que permite recopilar la información que no puede inferirse de manera fiable solo a partir del AST, como la clasificación de estímulos de inicio y final y la identificación explícita

³ <https://docs.oracle.com/javase/8/docs/api/javaw/swing/package-summary.html>

de los objetos de valor. Finalmente, se genera la salida del proceso ⑤: un modelo e3value exportado en un formato XML compatible con mxGraph⁴. Esta librería permite construir, visualizar y modificar diagramas interactivos basados en grafos e incorpora funcionalidades para el renderizado en un sitio web, lo que permite el *layout* automático del modelo de forma que puede ser visualizado y editado en herramientas con altos niveles de usabilidad, como draw.io⁵ o e3web⁶

Desde el punto de vista de la arquitectura software, la herramienta se organiza en capas que separan la interfaz, el análisis del contrato, la gestión del modelo semántico y la generación del diagrama para los editores web, lo que facilita su evolución e integración con otros puentes tecnológicos, soporte de notaciones adicionales, etc. La solución se encuentra disponible en su propio repositorio de código abierto⁷ donde también puede verse en acción.

4. Caso de estudio

Con el fin de comprobar la utilidad de la solución, se utilizó un caso de estudio que se emplea en actividades docentes para impartir contenidos sobre desarrollo de smart contracts con Solidity en cursos de grado y posgrado. El objetivo del caso de estudio en términos de validación de la propuesta era analizar hasta qué punto la solución era capaz de generar modelos de valor correctos y completos a partir de un contrato inteligente.

4.1. Smart Contract RSU Token

El dominio del caso de estudio es un hipotético programa de *universidad saludable*, cuyo propósito es promover hábitos de vida saludables entre los estudiantes mediante incentivos digitales en forma de tokens: *RSUToken*. Desde el punto de vista técnico, *RSUToken* implementa un sistema de incentivos sobre la red Ethereum siguiendo el estándar para tokens fungibles *ERC-20* (Ethereum Request for Comments), e integra mecanismos habituales en contratos reales, como el control de acceso basado en roles mediante los contratos *AccessControl* y funcionalidades asociadas a la quema de tokens mediante *ERC20Burnable* como puede apreciarse en la vista parcial del contrato que muestra la Figura 5.

A nivel funcional, el contrato representa un ciclo de valor completo que incluye la emisión del incentivo, la circulación del token, su canje por productos y el retorno institucional del valor, lo que permite observar cómo las acciones codificadas se corresponden con intercambios económicamente interpretables.

El escenario contempla cuatro participantes relevantes desde la perspectiva de negocio. El *Estudiante* actúa como usuario final, cuya participación en las actividades se quiere incentivar: recibe tokens como recompensa por participar en

⁴ <https://jgraph.github.io/mxgraph/>

⁵ <https://app.diagrams.net/>

⁶ <https://e3web.thevalueengineers.nl/>

⁷ <https://github.com/CommITURJC/SoliditytoE3ValueConverter>

```
1 // SPDX-License-Identifier: GPL-3.0
2 pragma solidity ^0.8.20;
3
4 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
5 import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol";
6 import "@openzeppelin/contracts/access/AccessControl.sol";
7
8 contract RSUToken is ERC20, ERC20Burnable, AccessControl {
9     bytes32 public constant CAFETERIA_ROLE = keccak256("CAFETERIA_ROLE");
10    bytes32 public constant RSU_ROLE = keccak256("RSU_ROLE");
11    address public universidad;
12
13    event AttendanceAwarded(address indexed student, uint256 amount);
14    event FoodPurchased(address indexed student, address indexed cafeteria, uint256 amount);
15    event TokensReturned(address indexed cafeteria, address indexed universidad, uint256 amount);
16
17    constructor() ERC20("RSU Healthy Token", "RSU") {
18        _grantRole(DEFAULT_ADMIN_ROLE, msg.sender);
19        _grantRole(RSU_ROLE, msg.sender);
20        universidad = msg.sender;
21    }
22
23    function awardAttendance(address student, uint256 amount) external {
24        onlyRole(RSU_ROLE)
25    {
26        _mint(student, amount);
27        emit AttendanceAwarded(student, amount);
28    }
29
30    function purchaseFood(address student, address cafeteria, uint256 amount) external {
31        require(student == msg.sender, "Solo el alumno puede pagar");
32        require(hasRole(CAFETERIA_ROLE, cafeteria), "RSUToken: destino no es cafeteria autorizada");
33        _transfer(student, cafeteria, amount);
34        emit FoodPurchased(student, cafeteria, amount);
35    }
36}
```

Figura 5. Vista parcial del contrato inteligente RSU.

actividades saludables y los utiliza como medio de pago en el entorno universitario. La *Cafetería* representa la entidad que acepta tokens como contraprestación por productos o servicios. Posteriormente, un mecanismo de reembolso le permite monetizar los tokens. La Oficina de Universidad Saludable o *RSU* es la unidad gestora encargada de validar la participación de los estudiantes, coordinar campañas y ejecutar funciones específicas del contrato. Finalmente, la *Universidad* actúa como emisor y garante del valor del token, gestionando la tesorería del sistema y la reabsorción de los tokens asociados al reembolso a las cafeterías.

La selección de este contrato como caso de estudio responde a varios criterios. En primer lugar, el hecho de ser utilizado con fines docentes facilita la reproducción del experimento y el control del escenario, sin penalizar su carácter realista. En segundo lugar, proporciona un escenario completo desde el punto de vista del intercambio de valor, ya que incluye incentivos, transacciones, canje y retorno. En tercer lugar, incorpora varios actores y roles definidos explícitamente, lo que favorece la detección de participantes a partir del análisis del AST. Además, el contrato incluye eventos que registran acciones relevantes, lo que permite evaluar la funcionalidad del asistente que solicita al usuario completar la interpretación semántica cuando esta no puede inferirse automáticamente, por ejemplo, al clasificar estímulos de inicio o fin. Por último, el contrato emplea elementos estándar y prácticas habituales en Solidity, como roles, modificadores, eventos y transferencias, lo que contribuye a que el análisis y la extracción de información sean más completos.

4.2. Modelo e³value generado

El contrato inteligente descrito en la sec. 4.1 sirve como punto de partida para comprobar el correcto funcionamiento de todas las fases o etapas del proceso de transformación implementado. Este proceso comienza con la generación y el análisis del AST, pasa por la construcción del modelo semántico intermedio, y termina con su conversión en un diagrama visual editable. El objetivo subyacente es comprobar si la solución propuesta cumple su objetivo principal: facilitar la comprensión de contratos inteligentes mediante representaciones orientadas al negocio.

Una vez completado el análisis semántico y respondidas las preguntas interactivas necesarias para construir el modelo, como la tipología del evento (inicial o final) o el objeto de valor que intercambia una de las partes, que no puede representarse en el propio contrato como un bien tangible, la herramienta genera automáticamente un archivo XML que contiene la descripción completa del modelo resultante. La interfaz muestra el contenido de dicho archivo y ofrece la opción de exportar el modelo o reiniciar el proceso. Si el usuario da por válido el resultado, deberá importar manualmente el modelo en el editor web deseado para obtener una representación visual del modelo e³value, como las que se muestran en la Figura 6.

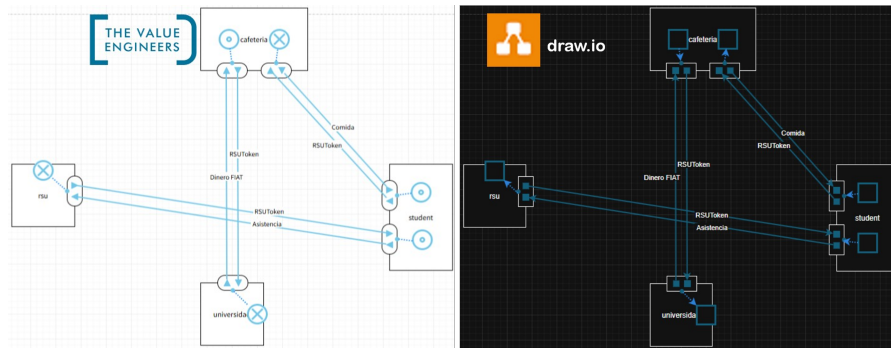


Figura 6. Modelo e³value resultante visualizado con e3web (izquierda) y draw.io (derecha)

Estas capturas confirman que el modelo generado es consumible por la librería gráfica mxGraph y aseguran la interoperabilidad de la solución con cualquier herramienta que haga uso de esta librería para el renderizado visual de modelos.

4.3. Análisis cualitativo del modelo e³value generado

Una vez que se ha comprobado que la solución es capaz de generar un modelo e³value editable a partir del contrato RSUToken, en esta sección se analiza cualitativamente la calidad del modelo desde dos dimensiones complementarias:

la **interpretabilidad** desde una perspectiva de negocio y la **fidelidad** respecto a las interacciones definidas en el contrato. El objetivo de este análisis no es verificar formalmente que el modelo refleja las propiedades del contrato, sino valorar en qué medida el modelo generado constituye una representación útil y comprensible del escenario implementado por el contrato.

Interpretabilidad desde la perspectiva de negocio. Uno de los objetivos principales de la propuesta es facilitar que perfiles de negocio puedan comprender el funcionamiento esencial de un contrato inteligente sin necesidad de entender el código que lo implementa. En este sentido, el modelo obtenido facilita la identificación de los tres intercambios de valor que se dan en el caso de estudio, y que, de alguna forma, constituyen un ciclo de valor:

- La recompensa de tokens al estudiante como incentivo por participar en actividades saludables.
- El uso posterior de esos tokens como medio de pago en la cafetería
- La devolución de tokens a la universidad como paso previo al reembolso a la cafetería, que cierra el ciclo de valor.

Estos intercambios se representan mediante actores conectados por intercambios de valor e interfaces que permiten identificar lo que cada participante entrega o recibe, y por estímulos que señalan el inicio o fin del intercambio. Así, la representación del contrato que ofrece el modelo posibilita una lectura orientada al intercambio económico que resulta mucho más accesible para profesionales que no conozcan Solidity y necesiten saber qué hace el contrato en cuestión.

Correspondencia con las interacciones especificadas en el contrato. El modelo reproduce de manera coherente las principales interacciones codificadas en el contrato RSUToken.

- La emisión de tokens, asociada a la función `awardAttendance` del contrato, se representa como un flujo de valor, cuyo estímulo de inicio es la asistencia del estudiante a la actividad, recibiendo a cambio el incentivo desde la RSU.
- El uso de tokens como medio de pago en la cafetería, codificado en el contrato con la función `purchaseFood`, aparece como una transacción entre estudiante y cafetería, junto con el evento de inicio de estímulo en el estudiante.
- El reembolso de los tokens a la cafetería, codificado en la función `redeem`, se transforma en una relación inversa con estímulo de inicio en el actor de la cafetería.

Todos los eventos han sido correctamente identificados como estímulos, y los objetos de valor modelados representan fielmente el valor económico que se intercambia en cada interacción. Asimismo, las relaciones entre actores son precisas. Aunque los roles, como `CAFETERIA_ROLE` o `RSU_ROLE`, no siempre definen explícitamente a qué entidades representan, la herramienta los asocia correctamente a actores en el modelo e³value a partir del uso que se hace de ellos en el contrato en funciones clave.

Limitaciones. El modelo generado mapea y representa correctamente todos los elementos de valor y las interacciones del contrato, pero existen algunos aspectos que, evidentemente, no son representados. Conviene explicitar que no se trata de limitaciones de la solución, sino limitaciones inherentes a la naturaleza de la notación e³value.

Esta pérdida de información es una consecuencia natural del nivel de abstracción al que trabaja e³value, que está diseñado para representar intercambios de valor y no para capturar lógica de control ni detalles técnicos. Por ejemplo, los privilegios administrativos no se reflejan en el modelo como un actor o mecanismo explícito de control, y validaciones internas, como expresiones `require`, utilizadas para restringir quién puede invocar una función o bajo qué condiciones puede invocarse, tampoco aparecen representadas, ya que no constituyen un intercambio de valor observable entre entidades. Del mismo modo, elementos técnicos como el uso de bibliotecas de control de acceso, la posibilidad de fraccionar el token o los efectos internos de operaciones como `mint` o `burn` sobre balances no se trasladan al modelo cuando no añaden información directa sobre el escenario de valor.

5. Trabajos relacionados

Por razones de espacio, este apartado ofrece un breve resumen del estado del arte en relación con las contribuciones de este trabajo.

El deficitario nivel de adopción de los contratos inteligentes en contextos reales ha impulsado en los últimos años la aparición de numerosas propuestas orientadas a acortar la distancia entre la especificación de modelos de negocio y su ejecución o implementación con tecnologías blockchain. Una parte relevante de estas propuestas se ha centrado en obtener contratos inteligentes a partir de modelos de proceso, siendo especialmente relevantes los que se centran en el uso de BPMN, como [12,14].

Existen también trabajos que, en cierto modo, también elevan el nivel de abstracción al que se puede contemplar un contrato inteligente. En particular, `sol2uml` es una herramienta que genera diagramas UML y diagramas de almacenamiento a partir de contratos Solidity [1]. Así, la herramienta facilita la comprensión técnica y estática del contrato, enfocándose en cómo está organizado internamente, pero evidentemente constituye un salto muy pequeño en cuanto a elevar el nivel de abstracción.

Por otro lado, en un trabajo reciente se presentaba una herramienta precisamente pensada para ayudar a la gente a entender qué hace realmente un smart contract [15]. La herramienta simula la ejecución del contrato con varios usuarios interactuando con él en diferentes situaciones y luego muestra el resultado con dibujos que ilustran qué funciones se activan, cómo fluye la criptomoneda o cómo cambian las variables internas del contrato. Así, mientras `PrettiSmart` aclara cómo se comportará el contrato en tiempo de ejecución mediante simulaciones, la propuesta que se presenta aclara qué modelo de negocio implementa

el contrato, proporcionando trazabilidad entre el código y la lógica económica que subyace.

Finalmente, hay trabajos que exploran directamente la conexión entre la notación e³value y los contratos inteligentes [13]. En particular, en [5] se propone una extensión no oficial de la notación e³value que incorpora elementos blockchain para facilitar el modelado de aplicaciones blockchain desde el punto de vista del diseño del modelo de negocio. Como resultado, los modelos elaborados con esta extensión se alejan del estándar de la notación. En cualquier caso, mientras que la propuesta que se presenta en este trabajo se centra en facilitar la comprensión de contratos reales ya escritos, la propuesta de Curty & Fill amplía la notación para concebir y analizar modelos de negocio blockchain desde cero. Dicho de otro modo, una traduce de código a negocio y la otra amplía la capacidad del negocio para modelar ecosistemas blockchain complejos.

6. Conclusiones

La propuesta presentada en este trabajo tiene por objetivo reducir la brecha provocada por la complejidad técnica de los contratos inteligentes y hacerlos más entendibles para todos los perfiles, especialmente para los profesionales de negocio. Para ello, se ha desarrollado un puente tecnológico unidireccional que transforma automáticamente contratos Solidity en modelos e³value, que facilitan enormemente la tarea de interpretar los intercambios de valor y la lógica económica implementada por el contrato, y por tanto, habilitan la validación y comunicación entre perfiles técnicos y no técnicos.

El caso de estudio desarrollado confirma la capacidad de la propuesta para generar modelos coherentes y completos, mapeando y registrando correctamente los elementos esenciales del ciclo de valor e incorporando información adicional mediante la interacción con el usuario, para resolver ambigüedades no inferibles desde el código. Aunque el nivel de abstracción propio de e³value implica inevitablemente la omisión de detalles técnicos en el proceso de transición, esta simplificación resulta adecuada para el objetivo de ofrecer una vista conceptual del contrato que mejore la comprensión del mismo.

La línea de trabajo futura más inmediata está obviamente relacionada con la validación de la solución, para la que se trabaja ya en dos direcciones. Por un lado, el desarrollo de un estudio que incorpore un cierto número de contratos y permita analizar cuantitativa y cualitativamente el proceso de transformación implementado por la solución, la cantidad y naturaleza de la información adicional que se necesita en cada caso, etc. Por otro lado, diseñar y ejecutar un experimento con usuarios reales que permite medir de forma sistemática la mejora que supone el uso de modelos e³value en términos de comprensión del contrato, y su impacto en tareas de validación del contrato.

Agradecimientos Este trabajo ha sido parcialmente financiado por la Comunidad Autónoma de Madrid a través del proyecto DESAFÍO-CM (**TEC-2024/COM-235**) y la UE mediante el proyecto SIC-SPAIN 4.0 (Safer Internet Centre Spain 4.0).

Referencias

1. Addison, N.: sol2uml: Solidity contract visualisation tool. <https://github.com/naddison36/sol2uml> (2023), a visualisation tool for generating UML class diagrams and storage layout diagrams from Solidity smart contracts
2. Anišić, H., Dragan, D., Gajić, D.B., Petrović, V.B.: An overview of smart contracts for non-programmers. In: 2024 11th International Conference on Electrical, Electronic and Computing Engineering (IcETRAN). pp. 1–6. IEEE (2024)
3. Bartoletti, M., Benetollo, L., Bugliesi, M., Crafa, S., Dal Sasso, G., Pettinau, R., Pinna, A., Piras, M., Rossi, S., Salis, S., et al.: Smart contract languages: A comparative analysis. *Future Generation Computer Systems* **164**, 107563 (2025)
4. Buterin, V.: A next-generation smart contract and decentralized application platform. *Ethereum White Paper* **3**(37), 2–1 (2014)
5. Curty, S., Fill, H.G.: A domain-specific e3value extension for analyzing blockchain-based value networks. In: IFIP Working Conference on The Practice of Enterprise Modeling. pp. 74–90. Springer (2023)
6. Gómez-Macías, C., Vara, J.M., Perez-Blanco, F.J., Granada, D., Villarrubia, C.: Soportando la producción de contratos inteligentes a partir de modelos de negocio. *SISTEDES* (2024), <https://hdl.handle.net/11705/JISBD/2024/121>
7. Gómez-Macías, C., Pérez-Blanco, F.J., Granada, D., Vara, J.M.: Integrating smart contracts into the modeling paradigm to harness the potential of models. *Software and Systems Modeling* pp. 1–20 (2025)
8. Gordijn, J., Akkermans, H.: Designing and evaluating e-business models. *IEEE intelligent Systems* **16**(4), 11–17 (2001)
9. Gómez-Macías, C., Pérez-Blanco, F.J., Vara, J.M., De Castro, V., Marcos, E.: Design and development of smart contracts for e-government through value and business process modeling. *Proceedings of the 54th Hawaii International Conference on System Sciences* pp. 20569–2078 (2021)
10. van Hoek, R.: Unblocking the chain—findings from an executive workshop on blockchain in the supply chain. *Supply Chain Management: An International Journal* **25**(2), 255–261 (2020)
11. Kushwaha, S.S., Joshi, S., Singh, D., Kaur, M., Lee, H.N.: Ethereum smart contract analysis tools: A systematic review. *Ieee Access* **10**, 57037–57062 (2022)
12. 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)
13. Poels, G., Kaya, F., Verdonck, M., Gordijn, J.: Early identification of potential distributed ledger technology business cases using e 3 value models. In: *International Conference on Conceptual Modeling*. pp. 70–80. Springer (2019)
14. 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)
15. Wen, X., Nguyen, T.D., Zhang, L., Sun, J., Wang, Y.: Prettismart: Visual interpretation of smart contracts via simulation. *IEEE Transactions on Visualization and Computer Graphics* (2025)
16. Zou, W., Lo, D., Kochhar, P.S., Le, X.B.D., Xia, X., Feng, Y., Chen, Z., Xu, B.: Smart contract development: Challenges and opportunities. *IEEE transactions on software engineering* **47**(10), 2084–2106 (2019)