УДК: 004.942
DOI 10.52575/2687-0932-2023-50-4-859-872

Системно-объектное моделирование смарт-контрактов

Жихарев А.Г., Киданов В.В., Корсунов Н.И.
Белгородский государственный национальный исследовательский университет,
Россия, 308015, г. Белгород, ул. Победы, 85
E-mail: 1104936@bsu.edu.ru

Аннотация. Научная статья посвящена системно-объектному моделированию смарт-контрактов на платформе Ethereum с использованием языка Solidity и инструмента Remix. В работе представлена имитационная модель создания и управления смарт-контрактом на основе метода системно-объектного моделирования. Модель включает описание ресурсов и иерархию операций и процессов, а также инструментария, связанных с созданием и взаимодействием с контрактами. Особое внимание уделяется использованию языка Solidity для написания смарт-контрактов и инструмента Remix для их разработки и тестирования. Необходимо отметить, что модель сфокусирована на функциональных аспектах, связанных с пользовательским взаимодействием, и не включает подробного рассмотрения блокчейновых функций, таких как работа виртуальной машины Ethereum (EVM) и включение транзакций в новый блок. Разработанная системно-объектная модель максимально унифицирована и послужит основой для дальнейшей разработки методологии проектирования смарт-контрактов с минимальным количеством уязвимостей исходного кода.

Ключевые слова: системно-объектная модель, смарт-контракт, уязвимость смарт-контракта, блокчейн

Для цитирования: Жихарев А.Г., Киданов В.В., Корсунов Н.И. 2023. Системно-объектное моделирование смарт-контрактов. Экономика. Информатика, 50(4): 859–872. DOI: 10.52575/2687-0932-2023-50-4-859-872

System-Object Modeling of Smart Contracts

Alexander G. Zhikharev, Vladislav V. Kidanov, Nikolay I. Korsunov
Belgorod State National Research University
85 Pobedy St, Belgorod, 308015, Russia
E-mail: 1104936@bsu.edu.ru

Abstract. The scientific article is devoted to the system-object modeling of smart contracts on the Ethereum platform using the Solidity language and the Remix tool. The paper presents a simulation model of the creation and management of a smart contract based on the method of system-object modeling. The model includes a description of resources and a hierarchy of operations and processes, as well as tools related to the creation and interaction with contracts. Particular attention is paid to the use of the Solidity language for writing smart contracts and the Remix tool for their development and testing. It should be noted that the model focuses on functional aspects related to user interaction and does not include a detailed consideration of blockchain functions, such as the operation of the Ethereum Virtual Machine (EVM) and the inclusion of transactions in a new block. The developed system-object model is maximally unified and will serve as a basis for further development of the methodology for designing smart contracts with a minimum number of source code vulnerabilities.

Keywords: system-object model, smart contract, smart contract vulnerability, blockchain

For citation: Zhikharev A.G., Kidanov V.V., Korsunov N.I. 2023. System-Object Modeling of Smart Contracts. Economics. Information technologies, 50(4): 859–872 (in Russian). DOI: 10.52575/2687-0932-2023-50-4-859-872

Введение

В современном мире блокчейн-технологии [Накамото, 2008] привлекают все большее внимание в сфере информационных технологий. Это связано с тем, что технология блокчейн позволяет гарантировать целостность информации, которая представлена и хранится с использованием данной технологии. В общем смысле технология блокчейн представляет собой Систему распределенного реестра [Накамото, 2008]. (Последовательность блоков или единиц цифровой информации, последовательно хранящихся в общедоступной базе данных, которая служит основой для криптовалют). Одной из ключевых составляющих блокчейн-сети являются смарт-контракты [Жихарев, Киданов, Фефелов, 2023] (далее СК), которые представляют собой программы, способные автоматизировать и гарантировать исполнение различных соглашений и условий между участниками блокчейн-сети. Рентабельность, экономия времени, безопасность, прозрачность и точность – вот лишь некоторые из преимуществ [Жихарев, Киданов, Фефелов, 2023]. В результате этих преимуществ рынок смарт-контрактов, вероятно, будет расширяться. Их использование позволяет автоматизировать и обеспечить надежность выполнения соглашений и транзакций в цифровом пространстве. Однако создание и управление смарт-контрактами требует комплексного подхода, а метод функционального моделирования [Жихарев, Белов, Рачинский, 2019] упростит понимание и ускорит их разработку.

В данной научной статье представлено системно-объектное моделирование смарт-контрактов в фреймворке «Remix» (интегрированной среды разработки (Integrated Development Environment), сокр. IDE) [Амир Латиф и др., 2020] на платформе Ethereum. Предложенный метод позволяет создавать графические модели различных процессов и операций, включая описание ресурсов, информации, инструментария и связей между ними.

Технология системно-объектного моделирования [Жихарев, Белов, Рачинский, 2019] может стать удобным инструментом для проектирования и верификации защищенных смарт-контрактов [Жихарев, Киданов, Фефелов, 2023], что обусловлено многими факторами, такими как: безопасность (ошибки в коде при разработке СК), доверие (неверное функционирование, атаки на СК и их уязвимость – затрудняют широкое применение технологии и снижает доверие потенциальных клиентов), экономическая рентабельность (автоматизация и прозрачность данных противостоят уязвимости, финансовым и репутационным рискам), развитие инновационной технологии (СК играют ключевую роль в эволюции развития блокчейн-технологии). Разработка простого и надежного инструмента для проектирования гарантированно защищенных смарт-контрактов будет способствовать развитию блокчейн-экосистемы и созданию более надежных и безопасных приложений.

В данной статье авторами был совершен важный шаг для ускорения процессов понимания и функционального моделирования смарт-контрактов, что является актуальной проблемой в области блокчейн-технологий. Разработанная модель может быть использована в дальнейших исследованиях и практических приложениях для оптимизации создания и управления СК в сети блокчейн.

Модель Смарт-Контракта

Системно-объектный подход моделирования представляет собой методологию анализа и проектирования систем. В рамках этого подхода система рассматривается как совокупность взаимосвязанных объектов, каждый из которых выполняет определенные функции и имеет свои характеристики.

Данный подход был выбран и взят за основу как наилучший в связи с тем, что он уделяет особое внимание объектам в системе и их взаимодействию. Объекты представляются абстрактными сущностями, которые имеют состояние, поведение и связи с другими объектами.

Основные принципы системно-объектного подхода включают:

- идентификацию и моделирование объектов в системе;
- определение и описание свойств и атрибутов объектов;

- анализ и определение взаимосвязей между объектами;
- определение функциональных возможностей каждого объекта;
- разделение системы на модули или компоненты на основе объектов.

Данный подход позволит лучше понять процесс разработки и структуру Смарт-Контракта на языке Solidity [Хегед, 2018], а также взаимодействие компонентов системы, что позволит облегчить разработку программного обеспечения за счет разделения системы на более мелкие модули, реализующие конкретные функции.

Этапы создания Смарт-Контракта:

0. Концепция. Определяется цель разработки Смарт-контракта для Ethereum, определение основных задач и условий будущего договора.

1. Разработка. Кодирование Смарт-контракта с использованием языка программирования (Solidity от Ethereum).

2. Компиляция. Перед развертыванием Смарт-контрактов их необходимо скомпилировать. Процесс преобразования кода вашего контракта в файл JSON, чтобы он мог быть прочитан другими веб-приложениями. (Пример: Смарт-контракт для Ethereum, после написания на Solidity компилируются в байт-код EVM [Дингман и др., 2019] (или виртуальной машины Ethereum), что делает их совместимыми со всеми сетями EVM).

3. Развертывание. Развертывание СК или фактическое размещение в выбранной сети. (При развертывании СК выполняется и совершается транзакция с использованием реальной криптовалюты). После этого этапа развернутый контракт запускается и все закодированные в нем функции заработают после выполнения установленных условий.

4. Тестирование. Тщательное тестирование перед размещением СК в сети (ведь Смарт-контракты неизменяемы после размещения и все ошибки, обнаруженные после развертывания, отредактировать невозможно).

5. Выполнение. После выполнения, развертывания и запуска СК в сети проверяется его работоспособность. Это включает в себя проверку кошельков (чтобы убедиться, что нужные балансы появляются в нужное время), решение задач по обслуживанию и проблем с хранением.

Применение системно-объектного подхода моделирования способствует созданию гибкой и масштабируемой системы, которая легко поддается изменениям и расширению. Он широко используется в таких областях, как разработка программного обеспечения, системный анализ, проектирование информационных систем и управление проектами.

Пример Смарт-Контракта:

```
    // SPDX-License-Identifier: MIT
pragma solidity ^0.8.15;
contract Wallet {
    address public owner;
mapping (address => uint) public payments;
constructor() {
    owner = msg.sender;
}
    function payForItem() public  payable {
      payments[msg.sender] = msg.value;
      }
    function withdrawal() public {
      address payable _ to = payable(owner);
            address _ thisContract = address(this);
    _ to.transfer(_thisContract.balance);
      }
    }
```

Листинг 1. Код Смарт-Контракта
Listing 1. Smart Contract, Code

В данном смарт-контракте присутствуют только две функции:

1. Функция "payForItem()". Это публичная функция без возвращаемого значения (void), которая принимает платеж от отправителя и записывает сумму платежа в словарь "payments" по адресу отправителя.

2. Функция "withdrawal()". Это публичная функция без возвращаемого значения (void), которая переводит все деньги, находящиеся на счете контракта, на адрес владельца контракта.

Важно, что в представленном смарт-контракте не используются классы [Чен и др., 2021], так как он изначально написан на языке Solidity, который не поддерживает классы в традиционном понимании. Однако с точки зрения контракта рассматривается его структура. В данном случае контракт "Wallet" является основным и единственным контрактом в данной модели.

Первым шагом моделирования системы в системно-объектном подходе является ее графическое представление, т.е. разработка контекстной диаграммы, которая позволяет определить границы системы и ее взаимодействие с другими системами или акторами.

Контекстная диаграмма приводит работу по разработке Смарт-Контракта (далее СК) в общий вид, рисунок 1.

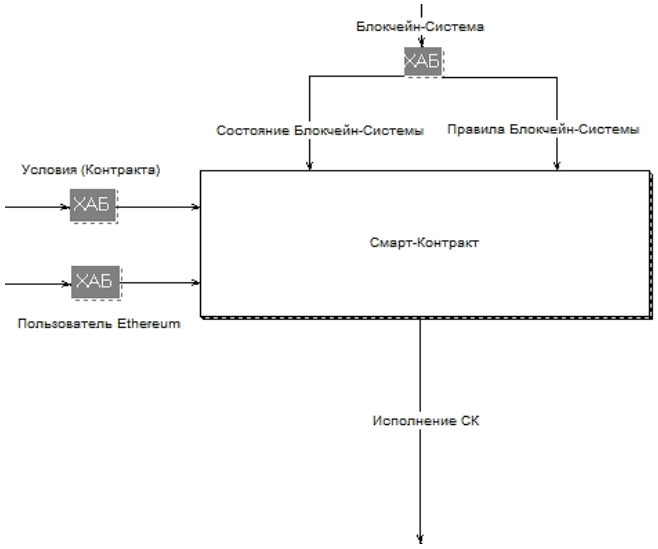

Рис. 1. Контекстная диаграмма взаимосвязей СК с внешней средой
Fig. 1. Contextual diagram of the relationship of the UK with the external environment

Информация по Данным.

Входные данные:

– текущее состояние Блокчейн-Системы;

– данные о контрагентах (Когнитивные данные лица, управляющего смарт-контрактом).

Выходные данные:

– исполнение Смарт-контракта

(Новое состояние Блокчейн системы – «завершенный Смарт-Контракт». (Блокчейн система приобретает новое состояние, когда исполняется Смарт-Контракт).

Информация по Управлению.

Общий вид – *Управление*:

– Правила Блокчейн Системы. (т.е. «Протоколы» которых нет, или свод правил сети Ethereum, как в нашем случае).

Механизм:

– автоматизированная Блокчейн-Система (техническое средство поддержки технологического бизнес-процесса по разработке и исполнению Смарт-Контракта.

Компоненты Блокчейн-платформы Ethereum:

• Вычислительные средства Ethereum (компьютеры пользователей) – EVM (Эфириум Вертуал Машина) является средой исполнения Смарт-Контракта в блокчейн Эфириума. (EVM – предоставляет исполнительную среду для выполнения кода СК написанных на Solidity (и в др. языках тоже)).

• Операторы (участники обеспечивают функционирование и поддержку инфраструктуры Эфириума. Операторы выполняют роль узлов сети [Conte de Leon et al., 2017], которые поддерживают работу сети и выполняют различные функции (например, проверка транзакций, создание блоков, хранение данных, обрабатывают транзакции и операции с Эфиром в соответствии с правилами Сети Эфириум (или «протоколом»)).

• P2P-сеть, реализованная в интернете – сеть, в которой участники (узлы) взаимодействуют напрямую друг с другом без посредников (каждый узел – равноправен в статусе и в функционале [Гао и др., 2019]. Например, направляет и принимает деньги)).

• Транзакции [Хан, Сарвар, Авайс, 2022] – сетевые сообщения, содержащие сведения об отправителе и получателе, значение и полезные данные.

• Клиенты (Eth)ereum [Дхулаввагол, Бхаджантри, Тотад, 2020] – Это ПО (которое помогает пользователем взаимодействовать с сетью Эфириум), самый известный «Remix IDE».

• Алгоритмы – алгоритм консенсуса. Ethereum [Хао др., 2018] заимствует модель консенсуса Bitcoin под названием Nakamoto Consensus. Она использует последовательные блоки с единичными подписями, где с помощью алгоритма PoW (proof of work – доказательство (выполнения) работы) [Хао др., 2018] определяется самая длинная цепочка (то есть текущее состояние); алгоритм безопасности PoW под названием Ethash, но в будущем на смену ему придет механизм консенсуса PoS (proof of stake – доказательство доли владения).

Декомпозицию процесса разработки Смарт-Контракта, с его последующим исполнением (рисунок 2), сформулируем в следующие предметные цели:

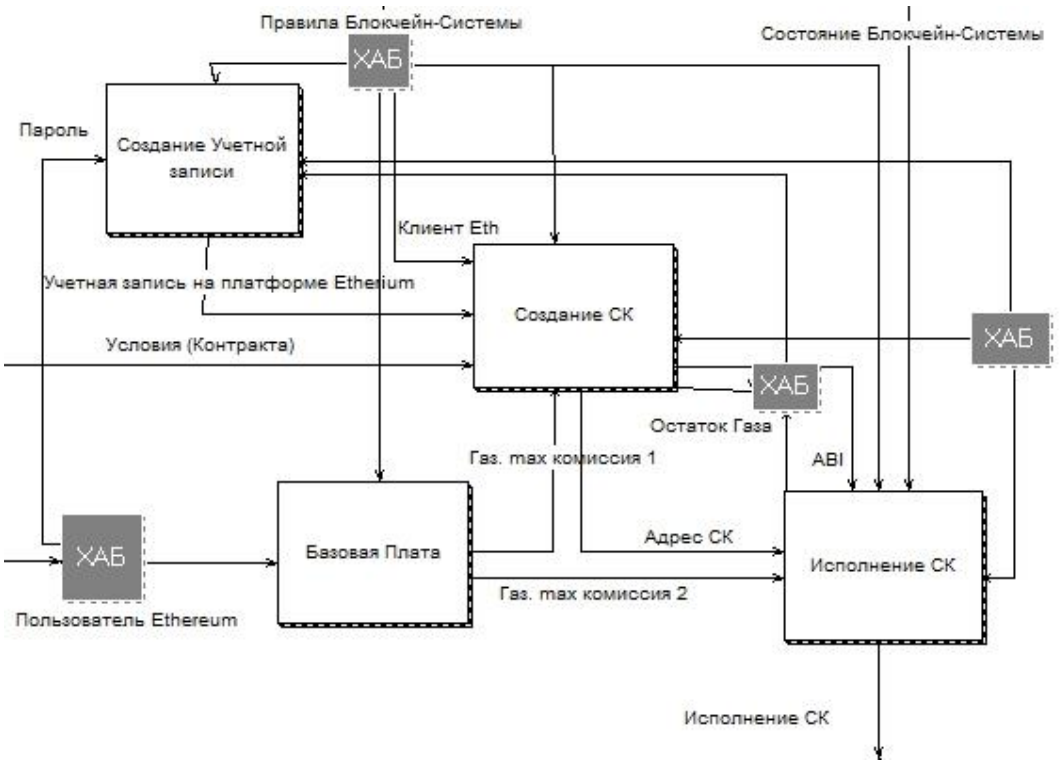

Рис. 2. Схема разработки и исполнения Смарт-Контракта
(без детализации Управления и Механизмов)
Fig. 2. Smart Contract development and execution scheme
(without detailed Controls and Mechanisms)

1) «Создание Смарт-Контракта», X2 – пользователь (разработчик) взаимодействует с СК через методы, что позволяет осуществлять управление контрактом и его функциональностью;

2) «Исполнение Смарт-Контракта», X3 – Пользователь может взаимодействовать с СК для выполнения определенных операций.

Разрабатывался Смарт-Контракт на Публичной Блокчейн Сети «Ethereum».

Для разработки Смарт-Контракта на языке Solidity и реализации полного пользовательского функционала на блокчейн-платформе Ethereum требуется Создание учетной записи внешнего владельца (EOA), X1.

«Базовая плата» (или «Плата за Газ» [Семельсбергер, Боруп, Грин, 2006]) X4, моделирующая процедура «уплаты комиссии» за транзакцию пользователем Сети Ethereum. Данная транзакция является обязательной, так как каждый блок имеет резервную цену, известную как «базовая плата». Это минимальная плата «за газ», которую пользователь должен заплатить, чтобы включить транзакцию в следующий блок.

Процесс «Создание учетной записи X1».

X1 – создание учетной записи в электронном кошельке MetaMask [Айер и др., 2018] (программное обеспечение, содержащее закрытые ключи. Используется для доступа к учетным записям Ethereum и управления ими, а также для взаимодействия с смарт-контрактами. Несмотря на название, кошельки никогда не хранят настоящие монеты или токены). Электронный кошелек – объект, содержащий адрес, баланс, одноразовое число и, при необходимости, хранилище и код. Учетная запись может принадлежать контракту или иметь внешнего владельца (англ. externally owned account, или EOA) [Касиредди, 2017].

Информация по Данным:

Входные данные:

– ПАРОЛЬ.

Выходные данные:

– Учетная Запись в блокчейн-платформе Ethereum.

Данные Учетной записи пользователя (разработчика Смарт-Контракта) содержат:

А) публичный ключ;

Б) приватный ключ;

В) баланс Eth.

Внутренняя транзакция (или «сообщение») [Касиредди, 2017]. Транзакция, отправленная с учетной записи контракта на другую учетную запись или EOA. (T0 – Транзакция по созданию контракта. Специальная транзакция с «нулевым адресом» в качестве получателя. Используется для регистрации контракта и записи его в блокчейн Ethereum. Tn – данные, переданные в блокчейн Ethereum, подписанные исходной учетной записью, нацелены на определенный адрес. Транзакция содержит метаданные, такие как лимит газа для этой транзакции).

Транзакция Ethereum – это действие по изменению состояния, которое инициирует пользователь, через свою учетную запись (пример – отправка зафиксированных данных на определенный адрес).

Открытый блокчейн работает как децентрализованные системы, ключевым аспектом которых является контроль приватных ключей [Антонопулос, Вуд, 2018], управляющих их средствами и доступом к «электронным кошелькам» или «учетной записи» (чтобы убрать путаницу: один приватный ключ – равен одной «учетной записи»). Данный процесс включает в себя 4 функции, рисунок 3.

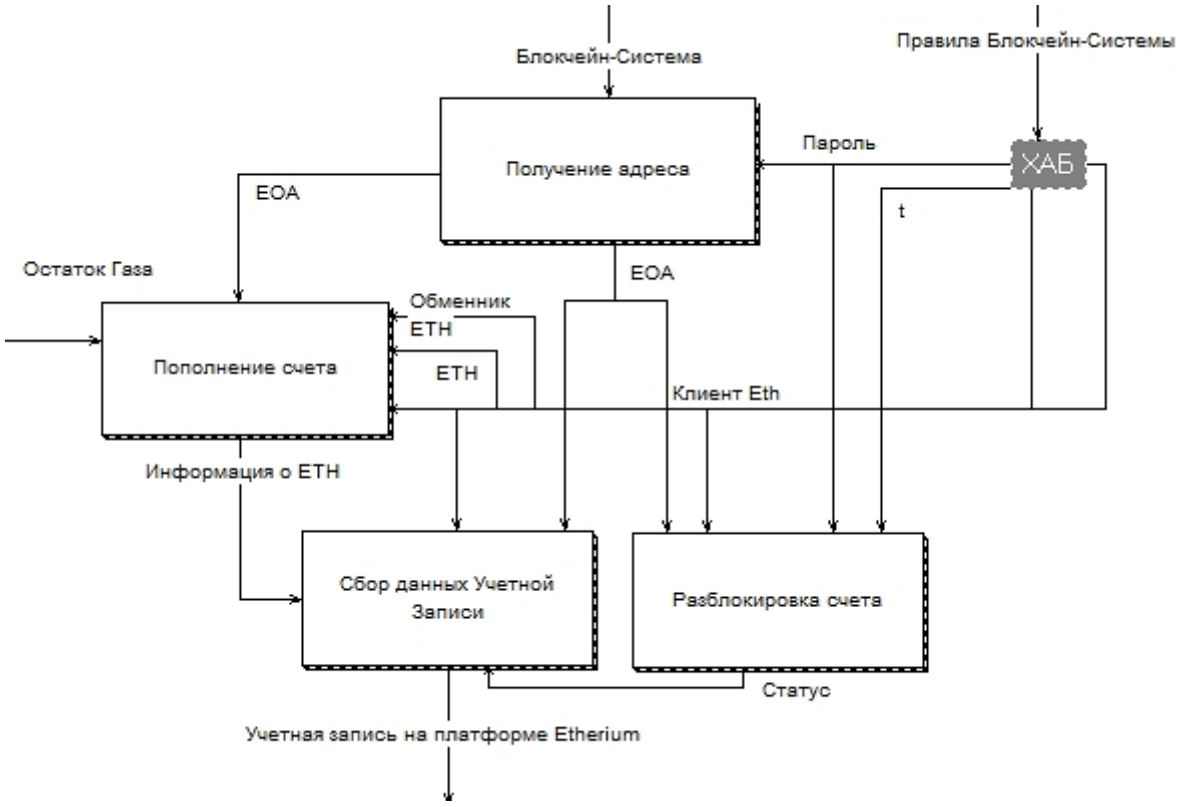

Рис. 3. Процесс «Создание кошелька», X1 (без детализации Управления)
Fig. 3. The process of «Creating a wallet», X1 (without detailed Controls and Mechanisms)

Создание «электронного кошелька» заключается в формировании запроса и получение адреса EOA (идентификатора).

Наш электронный кошелек MetaMask. При его активации генерируется мнемоническая резервная копия, состоящая из 12 английских слов. На странице учетной записи отображается следующая информация: название аккаунта, адрес Ethereum, название сети.

Неопытным разработчиками рекомендуем, первостепенно, переключить кошелек на тестовую сеть, чтобы не использовать реальные средства, а лишь ETH из «тестнета» [Вуд и др., 2014].

На диаграмме представлен механизм «Обменник валют», по средствам которого осуществляется пополнение счета. N-ная сумма ETH необходима для осуществления транзакции по исполнению СК.

С целью упрощения восприятия и увеличения читабельности *Управление* (соглашение Пользователя, правила работы с платформой Эфириум и его Клиентами) не показано.

Механизмы:

– платформа Ethereum;

– клиент Eth.

Функция «Пополнение счета» детально не моделируется.

Информация по Данным:

Входные данные:

– учетная запись (на которую распространяется Согласие Пользователя);

– количество Эфира;

– «Остаток Газа» от будущих транзакций.

Выходные данные:

– информация о количестве ETH.

Также для создание электронного кошелька необходимо «Разблокировать счет».

Информация по Данным:

Входные данные:

– запись ЕОА;

– ПАРОЛЬ;

– t (время, потраченное на разблокировку, в сек).

Входные данные:

– статус (информация об успешной разблокировке счёта или соответствующее уведомление об ошибке, если разблокировка не удалась по какой-либо причине).

В дальнейшем детали и специфика функции могут быть определены разработчиком смарт-контракта в соответствии с конкретными требованиями и сценариями использования (например, условие «ограничения» операции при определенном количестве неправильно введенных паролей).

Состояние данных актуализируется на этапе «Сбора данных учетной записи» (техническая функция), рисунок 4.

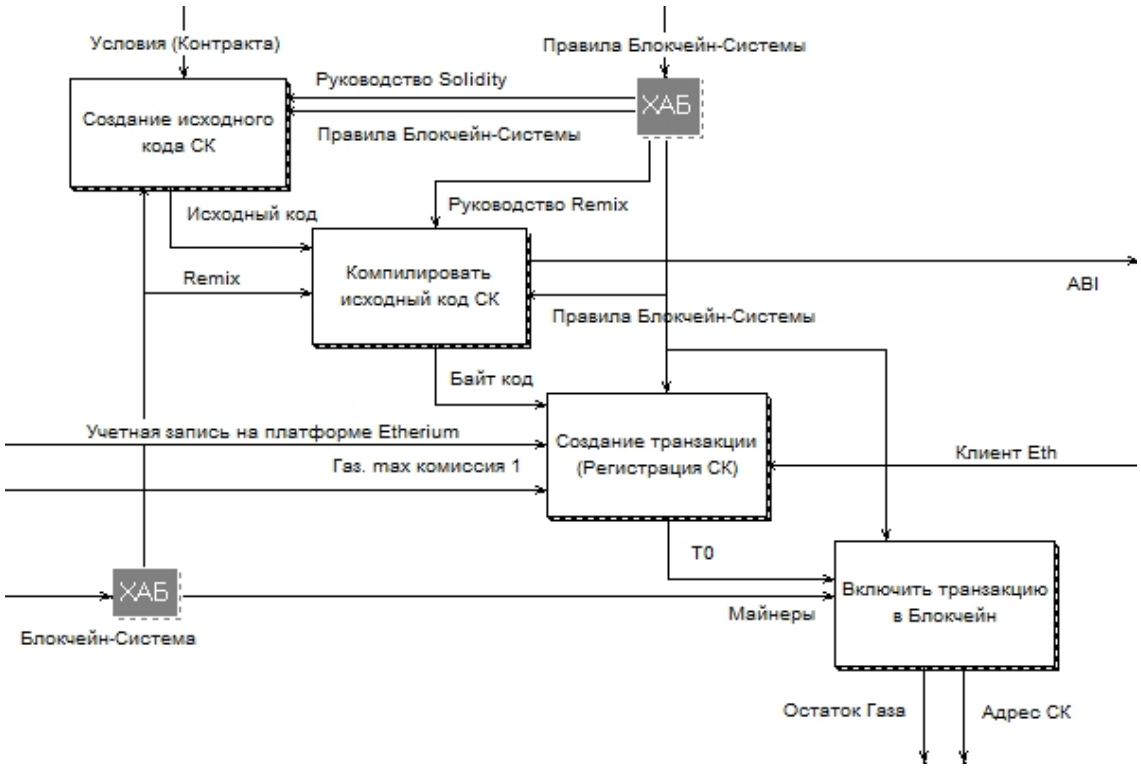

Рис. 4. Детализация процесса «Создание СК», X1

Fig. 4. Detailing of the process of «Creating a SC», X1

На данном этапе Пользователь разрабатывает программный код СК, который будет выполнять желаемые функции и исполнять логику контракта. Процесс детализации «Создание СК» на рисунке 4.

Функция «Создание исходного кода Смарт-Контракта». $X_{2.1}$ учитывает предмет и условия договора, которые прописываются на языке Solidity (самом известном языке для написания контракта и генерации байт-кода для EVM) по средствам интерфейса Remix IDE.

Информация по Данным:

Входные данные:

– условия контракта.

Входные данные:

– исходный код.

Управление – руководство Solidity.

Механизм – Remix IDE.

Для преобразования *Исходного кода Смарт-Контракта* (*Вход* $X_{2.2}$) в байт-код, который может быть выполнен машиной EVM в сети блокчейн, необходим компилятор Solidity. Компилятор поставляется в виде отдельного исполняемого файла и входит в состав различных фреймворков и интегрированных сред разработки (IDE), в нашем случае использовался Remix.

Информация по Данным:

Входные данные:

– исходный код.

Выходные данные:

– IBE, (двоичный интерфейс приложений);

– байт-код.

Управление – Руководство Solidity.

Механизм – Remix IDE.

ABI называют интерфейсом между двумя программными модулями или между операционной системой и пользовательской программой. ABI определяет способ доступа к структурам данных и функциям в машинном коде. ABI – это основной метод кодирования и декодирования данных в машинный код и обратно.

В Ethereum ABI используется для кодирования вызовов внутри контрактов с расчетом на EVM и для считывания данных из транзакций. Целью ABI-интерфейса является определение функций, которые можно вызывать из контракта, и описание того, какие аргументы принимает каждая из этих функций и какой результат она возвращает.

Следующим этапом является «регистрация СК» (функция «deploy» – развертывание) в блокчейне Ethereum. (Для проверки контракта использовался тестнет Ropsten). Регистрация контракта в блокчейне подразумевает создание специальной транзакции с конечным нулевым адресом (0x000…). Этот адрес сообщает блокчейну Ethereum о намерении зарегистрировать контракт Пользователем. В Remix данный шаг автоматизирован, отправку транзакции в MetaMask.

Транзакция «создания» не содержит эфир, а содержит 258 байт данных (скомпилированный контракт) [Сингх, Хосен, Юн, 2021]. Транзакции требуют платы и должны быть включены в подтвержденный блок. Gas – это вычисления, необходимые для обработки транзакции валидатором. Пользователи обязаны платить комиссию за вычисления.

Газ является ключевым компонентом Ethereum. С одной стороны, это буфер между (постоянно меняющейся) ценой эфира и наградой, которую майнеры получают за свою работу, с другой – это защита от DoS-атак [Вуд и др., 2014]. Чтобы предотвратить случайные или злонамеренные бесконечные циклы и прочие расточительные вычисления в сети, инициатор каждой транзакции обязан указать лимит на объем вычислений, за которые он готов заплатить. Таким образом, данный механизм делает невыгодной отправку многочисленных бесполезных транзакций, поскольку их цена является пропорциональной вычислительным ресурсам, трафику и объему потребляемого хранилища.

Когда виртуальной машине EVM нужно завершить транзакцию, ей вначале выдается запас газа в размере, указанном в транзакции в качестве лимита. Каждый выполняемый опкод имеет определенную стоимость, поэтому по мере продвижения (по этапам) выполнения кода программы запас газа уменьшается. Перед каждой операцией EVM проверяет, достаточно ли оставшегося газа для ее выполнения. Если ответ отрицательный, работа программы прекращается (прерывается), а транзакция откатывается.

Если EVM успешно достигнет конца выполнения, не превысив заданный лимит, потраченный газ выплачивается майнеру в качестве комиссии за транзакцию с предварительной конвертацией в эфир (в соответствии с ценой, указанной в транзакции):

*комиссия майнера = израсходованный газ * цена на газ*

Оставшийся газ возвращается отправителю, тоже с предварительной конвертацией в эфир в соответствии с ценой, указанной в транзакции:

оставшийся газ = лимит на газ израсходованный газ
*возмещенный эфир = оставшийся (неизрасходованный) газ * цена газа*
Информация по Данным:
Входные данные:
– байт-код;
– максимальная комиссия за газ.
Выходные данные:
– Т0 (транзакция «создания»).
Механизм – Блокчейн-Система, Клиент Eth.

Далее происходит «включение транзакции» в блокчейн сеть. При завершении транзакционного процесса (успешного или нет), майнерами генерируется квитанция транзакции (transaction receipt). (Эта квитанция содержит записи логов (англ. log entries, или logs), которые описывают действия, произошедшие в процессе выполнения транзакции. События – это высокоуровневые объекты языка Solidity, используемые для формирования данных логов).

* Квитанция (англ. receipt). Данные, возвращаемые клиентом Ethereum для представления результата отдельной транзакции, включая ее хеш, номер ее блока, объем расходованного газа (gas) и адрес контракта, если речь идет о его развертывании.

Отметим, адрес контракта также возможно использовать в транзакции в качестве получателя, посылая на него средства или вызывая его функции.

На этапе включения транзакции в блокчейн-сеть, майнеры [Вуд и др., 2014] выполняют функцию проверки и подтверждения транзакций, а также создают новые блоки в блокчейне. Они работают над решением сложных вычислительных задач, которые называются «доказательством работы» (Proof of Work), чтобы доказать свою производительность и защитить сеть от вредоносных действий. Их функции:

1. Проверка транзакций (их «правильность», подписи, количество доступных средств для перевода и соответствие правилам протокола блокчейна. Далее – транзакция добавляется в «Mempool» [Оливейра и др., 2021] (пул неподтвержденных транзакций)).

2. Создание блоков. (Создание блока включает сбор нескольких проверенных транзакций вместе и решение сложной вычислительной задачи для добавления блока в цепочку. Далее транзакция включается в блок и распространяется в блокчейн сети).

3. Децентрализованное управление. (Благодаря возможности принимать решения путем голосования за изменения с помощью протоколов, таких как "голосование по вилке" (fork voting), определить будущую направленность сети).

Исполнение СК

Смарт-контракт «WALLET» является программой для управления эфиром, исполняемая внутри виртуальной машины (EVM), которая создается в рамках специальной транзакции, (этап компиляции СК в байт-код). Когда СК появляется в сети блокчейн, он получает адрес Ethereum точно так же, как «кошелек». Каждый раз, когда кто-то отправляет транзакцию по этому адресу, контракт запускается в EVM (транзакции, отправляемые по адресу контракта, могут содержать эфир и/или данные. Эфир, если контракт содержит его, «депонируется» на баланс контракта. Если контракт содержит данные, они могут определять имя функции контракта и вызывать ее, посылая условия (аргументы) для функции.

Для вызова функции СК, обязательным условием по его выполнению является «разблокированный кошелек» Х1.4 и Транзакция Tn. Код Смарт-контракт будет активирован в сети блокчейн, когда сообщение (транзакция) с Учетной записи будет отправлена на Адрес смарт-контракта (для взаимодействия с СК необходимо знать его адрес и бинарный интерфейс приложения), рисунок 5.

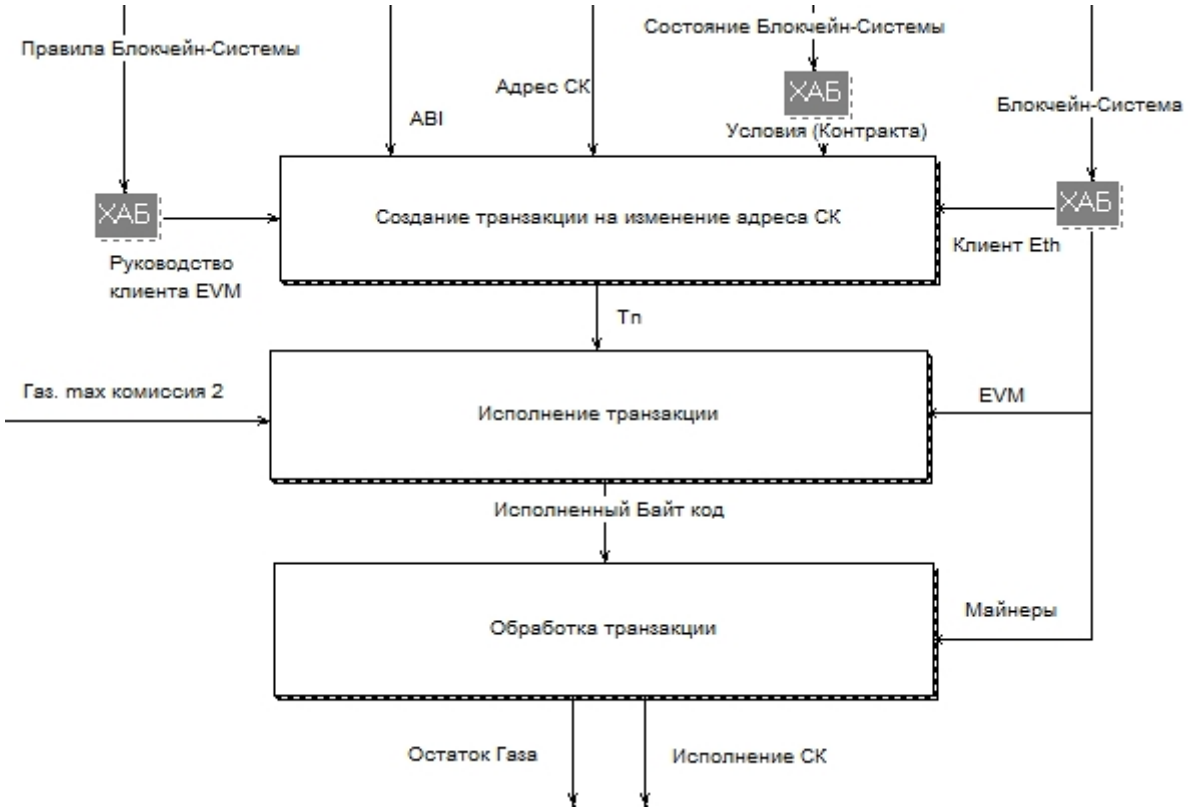

Рис. 5. Исполнение СК, X3
Fig. 5. Execution Smart contract, X3

Процедура изменения адреса СК в сети блокчейн $X_{3.1}$ аналогична $X_{2.3}$.
Отличия: наличие адреса СК и «сообщение» (внутренней транзакции, Tn).
Взаимодействие с СК происходит через электронный кошелек MetaMask.
Механизм – Клиент Eth.

Транзакция на этапе $X_{3.2}$ попадает в блокчейн Ethereum и включается в блок для дальнейшего исполнения. Виртуальная машина Ethereum (EVM) исполняет транзакцию путем выполнения байт-кода, указанного в транзакции. В случае изменения адреса смарт-контракта, EVM будет выполнять соответствующий код, указанный в транзакции.

Результаты исполнения транзакции на этапе $X_{3.3}$ «записываются» майнерами в блокчейн. Если адрес смарт-контракта изменился, новый адрес будет зафиксирован в блокчейне Ethereum. Это позволяет другим участникам сети узнать актуальный адрес смарт-контракта и взаимодействовать с ним. (Кроме того, результаты транзакции могут быть использованы для последующего аудита и проверки).

Заключение

В рамках исследования был разработан код смарт-контракта "Wallet". Контракт представляет собой простую систему кошелька, где пользователи могут делать платежи за товары или услуги, а владелец контракта осуществлять вывод средств.

Использование «UFO Modeller» для проведения системно-объектного моделирования позволило более полно и точно представить процесс разработки СК с целью его дальнейшей формализации.

Отметим, технология системно-объектного имитационного моделирования позволяет симулировать выполнение смарт-контракта, что позволит проверить его на адекватность генерируемых им транзакций [Жихарев, Киданов, Фефелов, 2023].

Дальнейшей актуальной задачей является анализ кода смарт-контрактов по средствам системно-объектного моделирования с точки зрения возникновения уязвимостей

программного кода. Такой подход позволит более детально изучить функциональность контрактов, а также выявить возможные уязвимости еще на этапе разработки, что позволит предотвращать потенциальные атаки или ошибки в смарт-контрактах.

Однако следует отметить, разрабатываемое решение на основе системно-объектного моделирования будет являться относительно новым и инновационно направленным. Дальнейшие исследования и разработки могут привести к улучшению инструментов и методик, стандартизации процессов разработки СК, а также расширению применения (методов / решений на основе) системно-объектного моделирования в области блокчейн-технологий.

Список литературы

Айер К. и др. 2018. Среда разработки ethereum. Создание игр с использованием смарт-контрактов Ethereum: промежуточные проекты для разработчиков Solidity. С. 19-36.

Амир Латиф Р.М. и др. 2020. Remix IDE: платформа на основе смарт-контрактов для сектора здравоохранения с использованием технологии блокчейн. Мультимедийные инструменты и приложения. С. 1-24.

Антонопулос А.М., Вуд Г. 2018.Освоение ethereum: построение смарт-контрактов и dapps. O'reilly Media.

Вуд Г. и др. 2014. Ethereum: безопасная децентрализованная книга обобщенных транзакций. Желтая книга проекта Ethereum. № 151. с. 1-32.

Гао Й. и др. 2019. Измерение и анализ топологии в сети ethereum p2p. Симпозиум IEEE 2019 по компьютерам и коммуникациям (ISCC). IEEE. С. 1-7.

Дингман У. и др. 2019. Дефекты и уязвимости в смарт-контрактах, классификация с использованием платформы NIST bugs framework. Международный журнал сетевых и распределенных вычислений.7(3): 121-132.

Дхулаввагол П.М., Бхаджантри В.Х., Тотад С.Г. 2020. Анализ производительности клиентов ethereum на блокчейне с учетом приложения для электронного голосования. Procedia Computer Science. Т. 167. С. 2506-2515.

Жихарев А.Г., Белов С.П., Рачинский С.А. 2019. Перспективы системно-объектного имитационного моделирования систем передачи информации. Экономика. Информатика. 46(3): 563-572.

Жихарев А.Г., Киданов В.В., Фефелов О.С. 2023. К вопросу о безопасности обработки информации с использованием смарт-контрактов. Научный результат. Информационные технологии. 8(1): 46-55.

Касиредди П. 2017.Как вообще работает Ethereum. Medium.

Накамото С. 2008. Биткоин: одноранговая система электронных платежей. Обзор децентрализованного бизнеса.

Оливейра В.С. и др. 2021. Анализ подтверждения транзакции в ethereum с использованием методов машинного обучения. Обзор оценки производительности ACM SIGMETRICS. 48(4): 12-15.

Семельсбергер Т.А., Боруп Р.Л., Грин Х.Л. 2006. Диметиловый эфир (DME) как альтернативное топливо. Журнал источников энергии. 156(2): 497-511.

Сингх С., Хосен А.С.М.С., Юн Б. 2021. Атаки на безопасность блокчейна, проблемы и решения для будущей распределенной сети интернета вещей. IEEE Access. Т. 9. С. 13938-13959.

Хан М.М.А., Сарвар Х.М.А., Авайс М. 2022. Анализ потребления газа транзакциями блокчейн Ethereum. Параллелизм и вычисления: практика и опыт. 34(4): e6679.

Хао Й. и др. 2018. Анализ производительности алгоритма консенсуса в частном блокчейне. Симпозиум IEEE Intelligent Vehicles Symposium 2018 (IV). IEEE. С. 280-285.

Хегед С.П. 2018. К анализу ландшафта сложности смарт-контрактов ethereum на основе solidity. Материалы 1-го Международного семинара по новым тенденциям в разработке программного обеспечения для блокчейна. С. 35-39.

Чен Дж. и др. 2021. Проверка дефектов: автоматическое обнаружение дефектов смарт-контрактов путем анализа байт-кода evm. IEEE Transactions on Software Engineering. 48(7): 2189-2207.

Conte de Leon D et al. 2017 "Blockchain: properties and misconceptions", Asia Pacific Journal of Innovation and Entrepreneurship, 11(3): 286-300.

References

Ayer K. et al. 2018. Ethereum Development Environment. Creating games using Ethereum smart Contracts: Interim projects for Solidity developers. pp. 19-36.

Amir Latif R.M. et al. 2020. Remix IDE: a platform based on smart contracts for the healthcare sector using blockchain technology. Multimedia tools and applications. pp. 1-24.

Antonopoulos A.M., Wood G. 2018.Mastering ethereum: Building Smart Contracts and dapps. O'reilly Media.

Wood G. et al. 2014. Ethereum: a secure decentralized book of generalized transactions. Yellow Book of the Ethereum project. No 151. p. 1-32.

Gao Y. et al. 2019. Measurement and analysis of topology in the ethereum p2p network. IEEE 2019 Symposium on Computers and Communications (ISCC). IEEE. pp. 1-7.

Dingman W. et al. 2019. Defects and vulnerabilities in smart contracts, classification using the NIST bugs framework // International Journal of Network and Distributed Computing. 7(3): 121-132.

Dhulavvagol P.M., Bhajantri V.H., Total S.G. 2020. Performance analysis of ethereum clients on the blockchain taking into account the application for electronic voting. Procedia Computer Science. Vol. 167. pp. 2506-2515.

Zhikharev A.G., Belov S.P., Rachinsky S.A. 2019. Prospects for system-object simulation modeling of information transmission systems. Economics. Information technologies. 46(3): 563-572.

Zhikharev A.G., Kidanov V.V., Fefelov O.S. 2023. On the issue of security of information processing using smart contracts. Research result. Information Technologies. 8(1): 46-55.

Kasireddi P. 2017.How does Ethereum work in general. Medium.

Nakamoto S. 2008. Bitcoin: Peer-to-peer electronic payment system. Review of decentralized business.

Oliveira V.S. et al. 2021. Analysis of transaction confirmation in ethereum using machine learning methods. Review of ACM SIGMETRICS performance evaluation. 48(4): 12-15.

Semelsberger T.A., Borup R.L., Green H.L. 2006. Dimethyl ether (DME) as an alternative fuel. Journal of Energy Sources. 156(2): 497-511.

Singh S., Hosen A.S.M.S., Yun B. 2021. Attacks on blockchain security, problems and solutions for the future distributed Internet of Things network. IEEE Access. Vol. 9. pp. 13938-13959.

Hanna., Sarvar H.M.A., And weiss M. 2022. Analysis of gas consumption by transactions of the Ethereum blockchain //Parallelism and computing: Practice and experience. 34(4): e6679.

Hao Y. et al. 2018. Performance analysis of the consensus algorithm in a private blockchain. IEEE Intelligent Vehicle Symposium 2018 (IV). IEEE. pp. 280-285.

Heged S.P. 2018. On the analysis of the complexity landscape of ethereum smart contracts based on solidity. Materials of the 1st International Seminar on New Trends in Software Development for Blockchain. pp. 35-39.

Chen J. et al. 2021. Defect checking: automatic detection of smart contract defects by analyzing evm bytecode. IEEE Transactions on Software Engineering. 48(7): 2189-2207.

Conte de Leon D et al. 2017. "Blockchain: properties and misconceptions", Asia Pacific Journal of Innovation and Entrepreneurship, 11(3): 286-300.

Конфликт интересов: о потенциальном конфликте интересов не сообщалось.
Conflict of interest: no potential conflict of interest related to this article was reported.

Поступила в редакцию 24.10.2023
Поступила после рецензирования 17.11.2023
Принята к публикации 01.12.2023

Received October 24, 2023
Revised November 17, 2023
Accepted December 01, 2023

ИНФОРМАЦИЯ ОБ АВТОРАХ

Жихарев Александр Геннадиевич, доктор технических наук, доцент, заведующий кафедрой автоматизированных систем и технологий, Белгородский государственный национальный исследовательский университет, г. Белгород, Россия

INFORMATION ABOUT THE AUTHORS

Alexander G. Zhikharev, Doctor of Technical Sciences, Associate Professor, Head of the Department of Automated Systems and Technologies, Belgorod State National Research University, Belgorod, Russia

Киданов Владислав Викторович, аспирант кафедры прикладной информатики и информационных технологий, Белгородский государственный национальный исследовательский университет, г. Белгород, Россия

Корсунов Николай Иванович, доктор технических наук, профессор, заслуженный деятель науки Российской Федерации, профессор кафедры математического и программного обеспечения информационных систем, Белгородский государственный национальный исследовательский университет, г. Белгород, Россия

Vladislav V. Kidanov, post-graduate student of the Department of Applied Informatics and Information Technologies, Belgorod State National Research University, Belgorod, Russia

Nikolay I. Korsunov, Doctor of Technical Sciences, Professor, Honored Scientist of the Russian Federation, Professor of the Department of Mathematical and Software Support of Information Systems, Belgorod State National Research University, Belgorod, Russia