УДК 65.012.123

В.Ю. Карпычев

ФУНКЦИОНАЛЬНАЯ МОДЕЛЬ СМАРТ-КОНТРАКТА НА ПЛАТФОРМЕ ETHEREUM

Нижегородский государственный технический университет им. Р.Е. Алексеева

Функциональное моделирование на основе стандарта IDEF0 – создание графических моделей любой предметной деятельности, включающих иерархическое описание процессов, операций, ресурсов (информации), инструментария, исполнителей, управления и связей между ними. В статье предложена функциональная модель создания и управления смарт-контрактами на платформе Ethereum.

Ключевые слова: функциональное моделирование, блокчейн-технологии, смарт-контракты, Эфириум.

Введение

В настоящее время большое внимание ученых и специалистов в различных областях привлекает блокчейн (БЧ) технология. Принимая во внимание множество определений данного термина, будем рассматривать в настоящей статье блокчейн как «многофункциональную и многоуровневую информационную технологию, предназначенную для надежного учета различных активов» [1]. Одним из актуальных вариантов применения этой технологии является *смарт-контракт* (smart-contract, «умный» контракт, далее − S-контракт). Под этим понятием в настоящей статье понимается специализированное программное обеспечение, содержащее элементы искусственного интеллекта (смарт) и предназначенное для поддержки экономической деятельности в части фиксации некоторых предметных правил, автоматического их исполнения при реализации набора детерминированных условий и хранения данных по исполненным договорам.

Рассмотрим основные функции S-контракта. Для описания его смарт-функции примем следующее определение: средства роботизированной автоматизации процессов (Robotic Process Automation) – «это программные «роботы», использующие правила бизнес-логики для выполнения четко заданных, повторяющихся функций и рабочих процессов точно так же, как это сделал бы человек» [2].

Блокчейн-технология S-контракта. Строго говоря, наличие смарт-функционала является необходимым, но недостаточным для такой программы. Полная архитектура S-контракта предполагает также использование технологии распределенного реестра (блокчейна) данных всех транзакций (исполнения обязательств), выполняемых программой. В настоящее время структура и функционал блокчейена рассмотрены достаточно подробно [3]. Цель настоящей статьи – представить работу S-контракта с точки зрения пользователя, сосредоточившись на выполнении S-функций. Для этого предпринята попытка интерпретировать S-технологию в виде функциональной модели методологии / стандарта SADT/IDEF0. В качестве теоретико-методологической основы создания функциональной модели S-контракта использованы работы [4,5]. Методология IDEF0 изложена во многих работах, в качестве первоисточника рекомендуем [6]. Стандарт IDEF0 принят в США [7] и рекомендован Госстандартом России для исследования структуры, параметров и характеристик производственных и организационно-экономических систем [8, 9].

Главными компонентами IDEF0-модели являются диаграммы, графически представляющие структуру функций предметной области, а также информации и объектов, связывающих эти функции. Функции и их интерфейсы представлены как блоки и дуги соответственно. При IDEF0-моделировании производится декомпозиция предметной контекстной (целе-

вой) функции на согласованные и непротиворечивые функции следующего уровня детализации. Предметная функция любого уровня, начиная с контекстной, преобразует входной поток сущностей – физических объектов или информации в выходной. Будем называть такие потоки *Входом* и *Выходом* функции. Для активации функций необходимы *Управление* − воздействие, определяющее условия выполнения функции, и *Механизмы* – средства, непосредственно реализующие функцию. Для поддержки моделирования используются CASE (Computer Aided Software Engineering) средства.

Характеристика предметной области

В качестве предметной области моделирования будем рассматривать S-технологию как средство поддержки классической договорной деятельности. В соответствии с проектом федерального закона № 419059-7 «О цифровых финансовых активах»: «*смарт-контракт* – договор в электронной форме, исполнение прав и обязательств по которому осуществляется путем совершения в автоматическом порядке цифровых транзакций в распределенном реестре цифровых транзакций в строго определенной таким договором последовательности и при наступлении определенных им обстоятельств».

При таком подходе заключение и исполнение S-контракта можно рассматривать как бизнес-процесс (БП) «*Управлять S-контрактом*», A_0.

IDEF0-модель

Построение IDEF0-модели БП начинается с контекстной диаграммы, A_0, описывающей в общем виде работу системы в предметной области рис. 1.

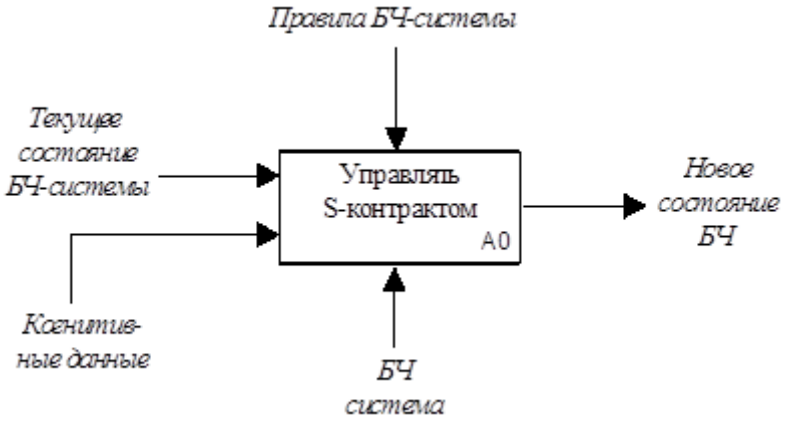

Рис. 1. Контекстная диаграмма БП «Управлять S-контрактом», A_0

В качестве *Входов* будем рассматривать *Текущее состояние БЧ-системы* и *Когнитивные данные* лица, управляющего S-контрактом. *Выходом* диаграммы является *Новое состояние БЧ-системы* (исполненное условие или завершенный S-договор).

Управление сформулируем в общем виде: *Правила БЧ-системы. Механизм:* техническим средством поддержки БП A_0, является БЧ-система – автоматизированная система, основные компоненты которой – вычислительные средства Ethereum, операторы и P2P-сеть, реализованная в интернете, клиенты (Eth)ereum.

На рис. 2 представлен первый уровень декомпозиции БП A_0, который можно рассматривать как последовательное достижение двух предметных целей: «Создать S-контракт», A_2 и «Исполнять S-контракт», A_3.

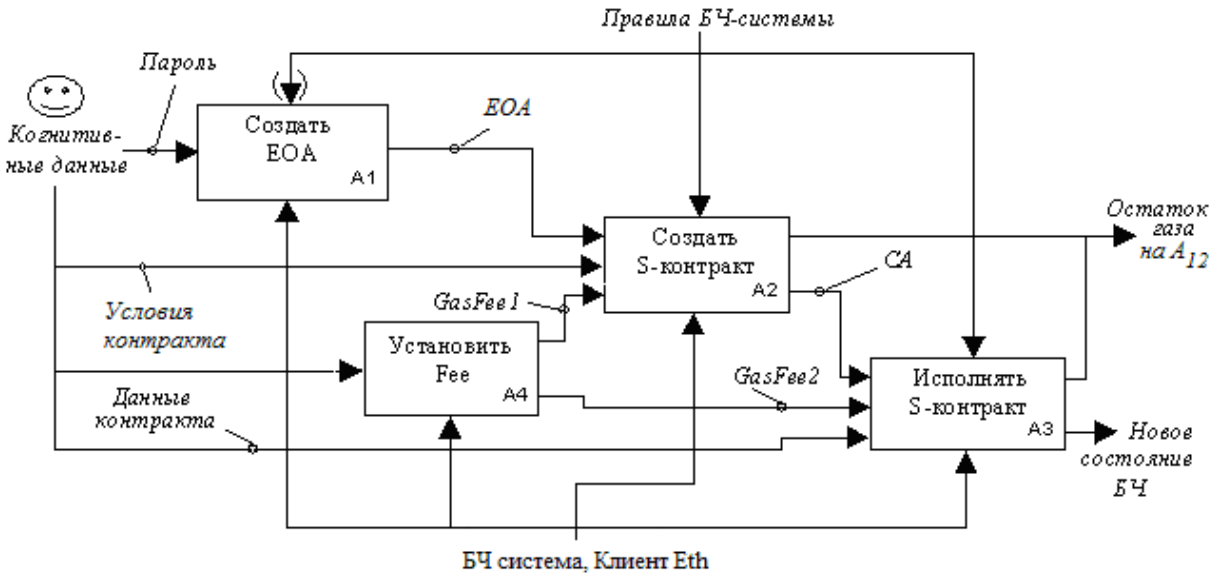

Рис. 2. Декомпозиция БП «Управлять S-контрактом», A_0

Для реализации пользовательского функционала в Ethereum (создания / использования S-контракта) необходима внешняя учетная запись (Externally owned account, EOA). Создание EOA моделируется процессом «Создать EOA», A_1. Также на этой диаграмме представлена функция «Установить Fee», A_4, моделирующая процедуры оплаты активности пользователя в Ethereum.

Управление и *Механизмы* на этой диаграмме не декомпозированы (не детализированы).

Процесс «Создать EOA», A_1

EOA используется для создания транзакций[1], хранения Эфира и управления (взаимодействия) с S-контрактом. Каждая EOA имеет определенное состояние и 20-байтовый адрес (идентификатор), используемый для ее идентификации. Таким образом, создание EOA заключается в формировании получения адреса в адресном пространстве Ethereum. *Входом* функции *A_1*. является *пароль*, который в дальнейшем используется для разблокировки счета. *Выход* функции A_1 – EOA, содержащая публичный и приватный ключи, баланс Eth. Содержание процесса *A_1*. раскрывается на его декомпозиции, рис. 3.

Процесс *A_1* включает четыре функции.

Функция «Получить адрес», A_{11} выполняется средствами клиента Ethereum (кошелька) и состоит в формировании запроса и получении идентификатора (адреса).

Функция «Пополнить счет» (A_{12}). Счет должен быть пополнен некоторой суммой Эфириума (Eth), необходимой для оплаты исполнения транзакций. Существуют различные варианты пополнения счета. На диаграмме представлен вариант пополнения с Обменника валют (внесистемный *Механизм* для A_{12}). *Входами* функции являются волеизъявление пользователя ☺ и идентификатор счета. Также на вход функции может поступать *Остаток Газа на A_{12}* от выполнения предшествующих транзакций с EOA.

При детальном моделировании (в статье не проводится) функция A_{12} также может быть декомпозирована для иллюстрации процедуры пополнения.

Функция A_{13} – техническая.

Функция «Разблокировать счет», A_{14}. Для использования OAE необходимо разблокировать. В качестве *Входов* A_{14} указываются идентификатор счета, пароль и продолжитель-

[1] Транзакция в Ethereum – это пакет данных, предназначенный для развертывания нового контракта (*Транзакция T^0*), перемещения эфира из одного счета на другой счет или в контракт (не рассматривается), или вызова метода контракта (*Транзакция T^n*) [8].

ность разблокировки в секундах, *t* (для кошелька MyEtherWallet). *Выход* A_{14} – санкция (*C*) на проведение операций со счетом.

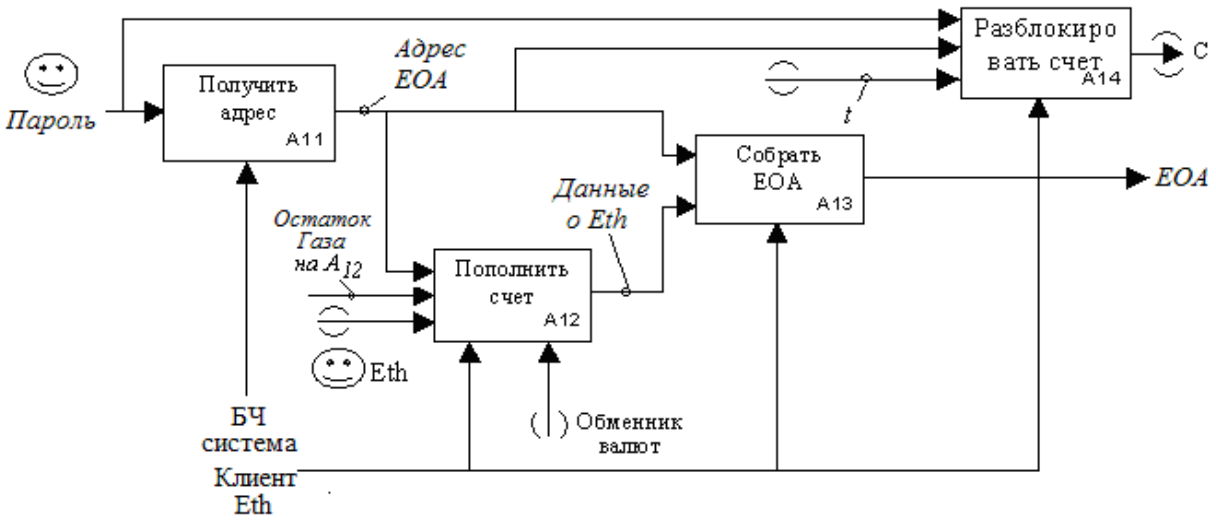

Рис. 3. Декомпозиция процесса «Создать EOA», A_1

Управление на декомпозиции A_1 представляет стандартные правила работы с Ethereum и его клиентом. Поэтому на рис. 3 оно не показано для улучшения чтения диаграммы. *Механизмы* декомпозиции A_1 – Ethereum и его клиент (кошелек).

Процесс «Создать S-контракт», A_2

Декомпозиция процесса «Создать S-контракт» представлена на рис 4.

Функция «Создать исходный код S-контракта», A_{21}. Договорные условия предметной области (*Вход*) программируются на компилируемом языке Solidity. *Выходами* является *Исходный код S-контракта.*

Управление – Руководство по объектно-ориентированному, предметно-ориентированному языку программирования *Solidity*. В качестве средства программирования (*Механизм*) можно использовать инструмент Remix Solidity IDE.

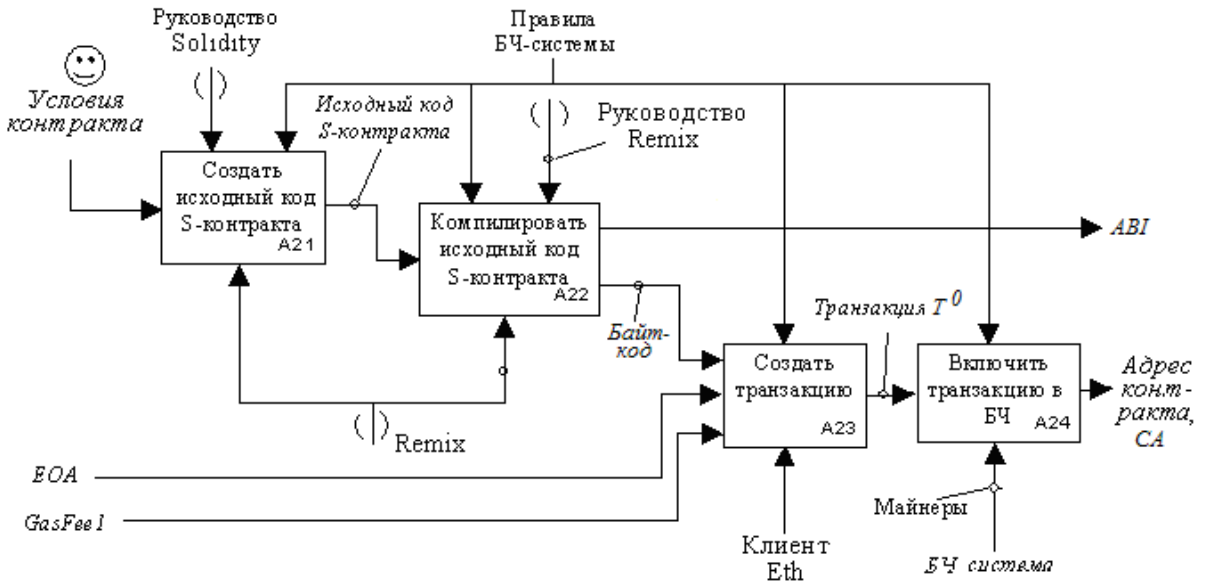

Рис. 4. Декомпозиция процесса «Создать S-контракт», A_2

Функция «Компилировать исходный код S-контракта», A_{22}. *Исходный код S-контракта* (*Вход* A_{22}) должен быть скомпилирован в *байт-код* – язык, используемый виртуальной машиной Ehereum (Ehereum Virtule Vachine, EVM). Байт-код S-контракта развертывается в блокчейн.

Для компиляции исходного кода на Solidity также можно использовать онлайн-компилятор Remix Solidity IDE (*Механизм* и *Управление* A_{22}).

На *Выходе* A_{22} также создается *бинарный интерфейс приложения* **(**Application Binary Interface, ABI**)**, *который необходим для указания вызываемой функции* контракта и возвращения данных в ожидаемом формате.

Функция «Создать транзакцию», A_{23}. Развертывание (deploy) S-контракта в БЧ осуществляется формированием и исполнением *Транзакции* «*0*» (создания). Для этого необходимы полный узел Ethereum или клиент (веб-интерфейс для обращения к полному узлу) и Эфир.

Создание S-контракта производится транзакцией без получателя. Такая транзакция передает байт-код контракта и Газ за исполнение кода создания контракта.

Для оплаты развертывания S-контракта необходимо установить значения *GasLimit* и *GasPrice*. На рис. 4 в качестве одного из входов показана *GasFee 1* (*Максимальная комиссия за транзакцию, Max transaction fee = GasLimit* x *GasPrice*). При недостатке средств (*GasFee 1, 2*) транзакция не выполняется.

Механизмы A_{23}: БЧ-система и ее клиент.

Функция «Включить транзакцию в БЧ» (A_{24}). *Транзакция* «*0*» (создания) S-контракта поступает в БЧ-систему, проходит полную обработку Майнерами (*Механизм*). После исполнения транзакции будет получена «квитанция», содержащая *Адрес* созданного *S-контракта* (Contract Account, CA).

Адрес S-контракта определяется в момент его создания (получается из адреса создателя и количества транзакций, отправленных с этого адреса).

Процесс «Исполнять S-контракт», A_3

Для вызова функций S-контракта, которые меняют его состояние (например, задают значения переменных) должен быть разблокирован кошелек (функция A_{14}), создана и направлена в контракт *Транзакция* T^n. Транзакция на S-контракт – это вызов его метода. Транзакция, отправленная от EOA к CA, предназначена для активации кода CA (исполнения S-контракта). *Транзакции* обоих видов (T^0 и T^n) имеют одинаковые поля и отличаются их значениями, табл. 1 (по [5]).

Таблица 1

Элементы транзакций S-контракта

Поле транзакции	T^0	T^n	**Содержание**
Nonce	+	+	Уникальный идентификатор транзакции. Количество транзакций, которые были отправлены со счета.
GasPrice	+	+	Стоимость 1 единицы Gas в Wei. Устанавливается отправителем
GasLimit	+	+	Максимальное количество Gas, которое может быть потреблено за проведение данной транзакции. GasLimit задается и оплачивается заранее, прежде чем будут проведены вычисления
TO	0	+	Адрес контракта
Value	+	+	Количество Wei, передаваемых от отправителя к получателю. В транзакциях T^0 (создания контрактов) - начальный баланс вновь созданной учетной записи EOA
Signature (r, s, v)	+	+	Поля, используемые для создания цифровой подписи, которая идентифицирует отправителя транзакции
Init	Байт-код	-	Поле используется для инициализации вновь созданной учетной записи контракта
Data	-	*Data*	Входные данные (параметры) для вызова сообщения

Декомпозиция A_3 приведена на рис. 5.

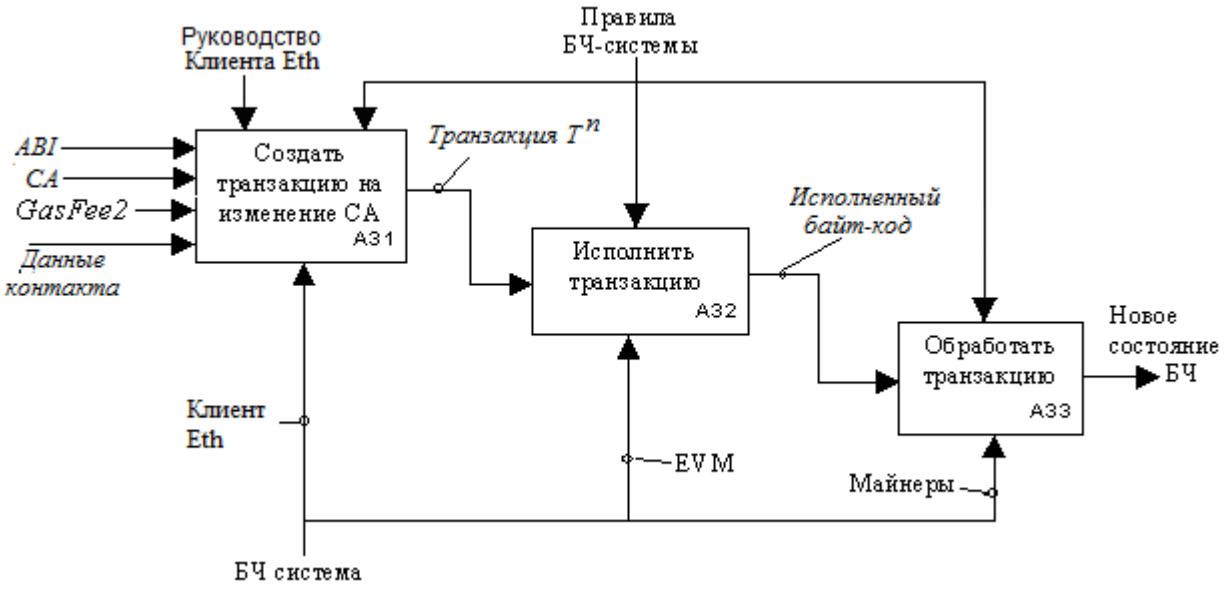

Рис. 5. Декомпозиция процесса «Взаимодействовать с S-контрактом», A_3

Функция A_{31} «Создать транзакцию на изменение CA» аналогична функции A_{23} на создание S-контракта. Отличиями A_{31} являются наличие адреса S-контракта (Вход *CA*) и значение *nonce* ≠ 0. Входы функции A_{31} приведены на рис. 5. Для взаимодействия с контрактом необходим его *бинарный интерфейс приложения* (Abstract Binary Interface, ABI), который можно получить, скомпилировав исходный код S-контракта, A_{22}.

Для взаимодействия с созданным S-контрактом используется Eth-клиент (*Механизм*), например, MyEtherWallet.

Функции A_{32} - A_{33} – штатные системные функции Ethereum – выполняются с использованием *Механизмов* и под *Управлением* Ethereum.

Функция A_{32} «Исполнить транзакцию» моделирует в общем виде работу виртуальной машины EVM по исполнению байт-кода S-контракта.

Функция A_{33} моделирует в работу множества майнеров по включению транзакции в блокчейн.

Заключение

В представленной модели отражены только основные функции, выполняемые пользователем при создании и взаимодействии с S-контрактом. За пределами детального рассмотрения (декомпозиции) остались «чисто» блок-чейновые функции, например, работа EVM, включение транзакции в новый блок и др. Также необходимо указать на отражение в модели ряда авторских представлений о логике работы S-контракта, в явном виде не описанной в литературе. Важной является вариативность модели в зависимости от используемого Eth-клиента.

Разработанную функциональную модель S-контракта следует в целом рассматривать как первую версию, требующую в соответствии с принципами IDEF0/SADT моделирования итеративной доработки.

Библиографический список

1. **Свон, М.** Схема новой экономики / М. Свон. – М.: Олимп-Бизнес, 2016, – 224 с.
2. **Берк, П.** Когда стоит применять программных роботов / П. Берк // ITWeek. – URL: https://www.itweek.ru/ai/article/detail.php?ID=189893 (дата обращения 18.04.2019).
3. **Карпычев, В.Ю.** Функциональное моделирование (IDEF0) как метод исследования блокчейн-технологии / В.Ю. Карпычев // Труды НГТУ. – 2018. – № 4. – С. 22-32.
4. **Прасти, Н.** Блокчейн. Разработка приложений / Н. Прасти. – СПб.: БХВ-Петербург, 2018. – 256 с.
5. **Bashir, I.** Mastering BlockChain / I. Bashir, Packt, 2017, – 531 p.
6. **Марка, Д.А.** Методология структурного анализа и проектирования SADT (Structured Analysis & Design Technique) / Д.А. Марка, К. МакГоуэн. – М.: МетаТехнология, 1993, – 240 с.
7. Integration DEFinition for function modeling (IDEF0). Draft Federal Information Processing Standards Publication 183, 1993 December 21. – URL: http:idef.com/wp-content/uploads/2016/02/idef0.pdf (дата обращения 15.04.2019).
8. РД IDEF0-2000. Методология функционального моделирования IDEF0. – М.: Госстандарт России, 2000, – 75 с.
9. Р 50.1.028-2001. Рекомендации по стандартизации. Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования. – М.: Госстандарт России, 2003, – 50 с.

Дата поступления
в редакцию: 17.09.2018

V.Y. Karpychev

A FUNCTIONAL MODEL OF SMART CONTRACT ON THE ETHEREUM PLATFORM

Nizhny Novgorod state technical university n.a. R.E. Alekseev

Purpose: Purpose: Development of the client part of a smart contract model using the IDEF0 notation.
Methodology: The proposed model of a smart contract is created with the help of SADT / IDEF0, a well-known methodology of structural analysis and design.
Value: The proposed model can be used in the development of original smart contracts, as well as in their fundamental research and training.
Research implications: Further research consists of the following: extending the model, presenting the structural and parametric characteristics of a smart contract, identifying possible vulnerabilities and limitations of the model, and proposing ways to overcome them.

Key words: functional modeling, blockchain technologies, smart contracts, Ethereum.