НАУЧНО-ИЗДАТЕЛЬСКИЙ ЦЕНТР «АКАДЕМИЯ ЕСТЕСТВОЗНАНИЯ»

SCIENTIFIC PUBLISHING CENTER «ACADEMY OF NATURAL HISTORY»

НАУЧНОЕ ОБОЗРЕНИЕ • ТЕХНИЧЕСКИЕ НАУКИ № 6

SCIENTIFIC REVIEW • TECHNICAL SCIENCES 2021

Журнал «Научное обозрение. Технические науки» зарегистрирован Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство ПИ № ФС77-57440
ISSN 2500-0799

Двухлетний импакт-фактор РИНЦ – 0,270
Пятилетний импакт-фактор РИНЦ – 0,242

Учредитель, издательство и редакция: ООО НИЦ «Академия Естествознания», Почтовый адрес: 105037, г. Москва, а/я 47 Адрес редакции и издателя: 410056, Саратовская область, г. Саратов, ул. им. Чапаева В.И., д. 56

Founder, publisher and edition: LLC SPC Academy of Natural History, Post address: 105037, Moscow, p.o. box 47 Editorial and publisher address: 410056, Saratov region, Saratov, V.I. Chapaev Street, 56

Подписано в печать 30.12.2021
Дата выхода номера 31.01.2022
Формат 60×90 1/8

Типография ООО НИЦ «Академия Естествознания», 410035, Саратовская область, г. Саратов, ул. Мамонтовой, д. 5

Signed in print 30.12.2021
Release date 31.01.2022
Format 60×90 8.1

Typography LLC SPC «Academy Of Natural History» 410035, Russia, Saratov region, Saratov, 5 Mamontovoi str.

Технический редактор Байгузова Л.М. Корректор Галенкина Е.С., Дудкина Н.А.

Тираж 1000 экз.
Распространение по свободной цене
Заказ НО 2021/6

Журнал «НАУЧНОЕ ОБОЗРЕНИЕ» выходил с 1894 по 1903 год в издательстве П.П. Сойкина. Главным редактором журнала был Михаил Михайлович Филиппов. В журнале публиковались работы Ленина, Плеханова, Циолковского, Менделеева, Бехтерева, Лесгафта и др.

Journal «Scientific Review» published from 1894 to 1903. P.P. Soykin was the publisher. Mikhail Filippov was the Editor in Chief. The journal published works of Lenin, Plekhanov, Tsiolkovsky, Mendeleev, Bekhterev, Lesgaft etc.

М.М. Филиппов (M.M. Philippov)

С 2014 года издание журнала возобновлено Академией Естествознания

From 2014 edition of the journal resumed by Academy of Natural History

НАУЧНОЕ ОБОЗРЕНИЕ • ТЕХНИЧЕСКИЕ НАУКИ

SCIENTIFIC REVIEW • TECHNICAL SCIENCES

www.science-education.ru

2021 г.

В журнале представлены научные обзоры, статьи проблемного и научно-практического характера

The issue contains scientific reviews, problem and practical scientific articles

СОДЕРЖАНИЕ

CONTENTS

СТАТЬИ

УДК 621.315.55:537.632.3

МАГНИТНОЕ ПОЛЕ ТОНКИХ ПОЛОСОК ТОКА

Глущенко А.Г., Глущенко Е.П., Ванькова А.Е.

Поволжский государственный университет телекоммуникаций и информатики, Самара, e-mail: gag646@yandex.ru

Эффективное уменьшение размеров электронных систем коммуникации возможно при использовании интегральной технологии. Это определило необходимость разработки эффективных методов расчета схем, изготовляемых методами интегральной технологии. Рассматриваются особенности создания магнитного поля широкими полосками проводников с равномерной плотностью тока, используемых при построении интегральных схем устройств управления параметрами электронных систем телекоммуникации. Известные результаты расчета магнитных полей, создаваемых токами, ограничиваются системами тонких проводников. Это ограничивает область применимости результатов расчета областью слаботочной электроники, когда плотность тока не превышает допустимых значений, определяемых потерями на эффект Джоуля–Ленца. Увеличение уровня мощности интегральных устройств с обеспечением допустимого уровня тепловых потерь определяет необходимость увеличения ширины полосок проводящих элементов интегральных схем и роста уровня мощности создаваемого токами интегральных элементов магнитных полей. С использованием закона Био–Савара–Лапласа получено аналитическое решение для расчета магнитных полей, создаваемых вблизи широких полосок, с равномерным распределением тока по прямому проводнику и для широких кольцевых полосок. Полученные результаты позволяют конструировать высокотехнологичные интегральные схемы высокой мощности с высокой теплостойкостью и механической надежностью.

Ключевые слова: магнитное поле, полоска проводника с током

MAGNETIC FIELD OF THIN CURRENT STRIPS

Glushchenko A.G., Glushchenko E.P., Vankova A.E.

Volga State University of Telecommunications and Informatics, Samara, e-mail: gag646@yandex ru

An effective reduction in the size of electronic communication systems is possible with the use of integral technology. This determined the need to develop effective methods for calculating circuits manufactured by the methods of integrated technology. The features of creating a magnetic field by wide strips of conductors with a uniform current density used in the construction of integrated circuits of devices for controlling the parameters of electronic telecommunication systems are considered. The known results of calculating the magnetic fields generated by currents are limited to thin conductor systems. This limits the area of applicability of the calculation results to the area of low-current electronics, when the current density does not exceed the permissible values determined by the losses due to the Joule–Lenz effect. An increase in the power level of integrated devices with the provision of an acceptable level of heat losses determines the need to increase the width of the strips of the conducting elements of the integrated circuits and to increase the power level generated by the currents of the integrated elements of magnetic fields. Using the Bio–Savart–Laplace law, an analytical solution is obtained for calculating magnetic fields generated near wide strips with a uniform current distribution along a straight conductor and for wide ring strips. The results obtained make it possible to design high-tech integrated circuits of high power with high heat resistance and mechanical reliability.

Keywords: magnetic field, conductor strip with current

Внедрение микросхем в интегральном исполнении привело к необходимости разработки методов расчета магнитных полей, создаваемых элементами микросхем, с учетом увеличения степени взаимовлияния интегральных элементов в планарном исполнении, которое имеет обычно приемлемые значения для слаботочных сигналов, но должно учитываться при росте уровня мощности новых интегральных элементов микро– и наноразмеров в различных диапазонах частот [1, 2].

Расчет поля проводится аналитически только для тонких проводников или численными методами для более сложных конфигураций [3]. Этот недостаток теории требует создания новых физико-математических моделей [3, 4], которые позволят получить аналитические решения для схем планарной технологии. Мощность микроэлектронных электронных устройств, выполняемых обычно в интегральном исполнении, определяется сечением токопроводящих элементов из-за роста джоулевых потерь энергии при увеличении токов в схемах обработки информации и ограничена уровнями десятков микроватт, поэтому дальнейшее увеличение уровня мощности передаваемых сигналов в интегральных схемах требует развития методов аналитического расчета интегральных структур, в частности методов расчета электромагнитных полей в этих структурах [5]. Это позволяет создать новые устройства управления, в том числе в интегральном исполнении. Использование тонких пленок проводников тока дает возможность достаточно просто создать управляемое магнитное поле в интегральных схемах.

Цель исследования: вывод аналитических соотношений для расчета напря-

женности магнитного поля, создаваемого широкими полосками тока с равномерной плотностью тока по сечению проводников, для наиболее часто встречающихся в устройствах микро– и наноэлектроники прямолинейных и кольцевых конфигураций (рис. 1, 2).

Моделирование и основные соотношения

Для расчета магнитного поля, создаваемого элементом проводника с током, используется закон Био–Савара–Лапласа:

$$d\mathbf{H} = \frac{I[\mathbf{dl}, \mathbf{r}]}{4\pi r^3}.$$

Для прямолинейного проводника, показанного на рис. 1:

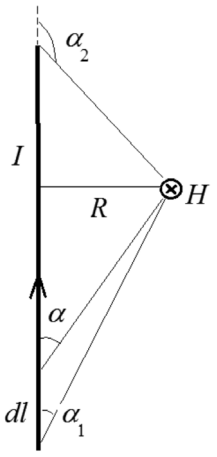

Рис. 1. К расчету магнитного поля, создаваемого тонким прямым проводником с током

напряженность магнитного поля, создаваемого в любой точке прямолинейным тонким проводником конечной длины, определяется соотношением [1, 2]:

$$H = \frac{I}{4\pi R}\left(\cos\alpha_1 - \cos\alpha_2\right),$$

где R – кратчайшее расстояние от точки наблюдения до линии действия тока, α_1 и α_2 – углы, под которыми видны концы проводника из точки наблюдения.

Результаты исследования и их обсуждение

Рассмотрим тонкую полоску токопроводящего элемента шириной $w = w_1 + w_2$, по которому идет ток I в направлении оси 0z (рис. 2).

Каждый элемент тока тонкой полоски шириной dl с линейной плотностью тока j в полоске создает компоненту напряженности магнитного поля:

$$dH = \frac{dI}{4\pi r(\beta)}\left(\cos\alpha_1 - \cos\alpha_2\right) = \frac{jdl}{4\pi r(\beta)}\left(\cos\alpha_1 - \cos\alpha_2\right),$$

где $j = \frac{I}{w}$ – линейная плотность тока, $dI = j \cdot dl$.

Из рис. 2 следует, что:

$$CD = dl \cdot \sin\beta,\ CD = r \cdot \sin(d\beta) \simeq r \cdot d\beta$$

Отсюда: $dl \simeq \frac{r \cdot d\beta}{\sin\beta}$.

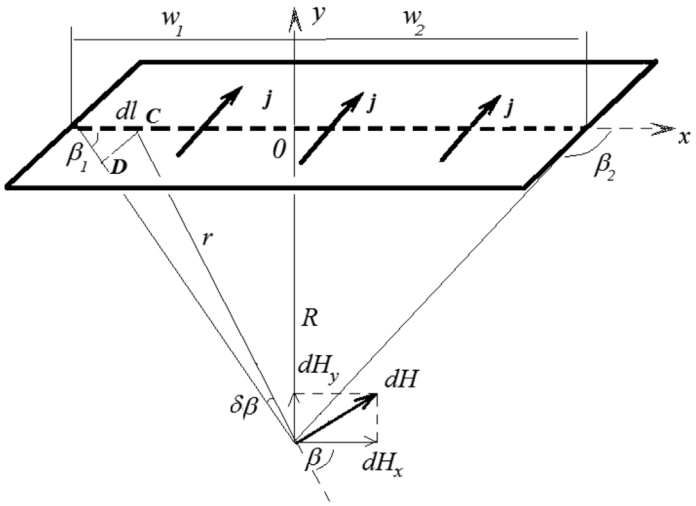

Рис. 2. Схема расчета магнитного поля, создаваемого широкой токопроводящей полоской

Проекции вектора напряженности магнитного поля на координатные оси x0y равны: $dH_x = dH \cdot \sin\beta$,

$$dH_y = dH \cdot \cos\beta .$$

Компонента поля H_x определяется из:

$$H_x = \int_0^w dH_x = \int_0^w \frac{jdl}{4\pi r} \cdot \sin\beta \cdot (\cos\alpha_1 - \cos\alpha_2) = \int_{\beta_1}^{\beta_2} \frac{j \cdot rd\beta}{4\pi r \cdot \sin\beta} \cdot \sin\beta \cdot (\cos\alpha_1 - \cos\alpha_2)$$

или

$$H_x = \int_{\beta_1}^{\beta_2} \frac{jd\beta}{4\pi} \cdot (\cos\alpha_1 - \cos\alpha_2) = \frac{j}{4\pi}(\cos\alpha_1 - \cos\alpha_2)(\beta_2 - \beta_1) =$$

$$= \frac{j}{4\pi}(\cos\alpha_1 - \cos\alpha_2)\left(\pi - arctg\frac{w_2}{R} - arctg\frac{w_1}{R}\right).$$

Окончательно компонента магнитного поля определяется соотношением:

$$H_x = \frac{I}{4\pi w}(\cos\alpha_1 - \cos\alpha_2)\left(\pi - arctg\frac{w_2}{R} - arctg\frac{w_1}{R}\right)$$

где $w = w_1 + w_2$ – ширина полосы.

Компонента магнитного поля H_y определяется из соотношения:

$$H_y = \int_0^w dH_y = \int_0^w \frac{jdl}{4\pi r} \cdot \cos\beta \cdot (\cos\alpha_1 - \cos\alpha_2) = \int_{\beta_1}^{\beta_2} \frac{j \cdot rd\beta}{4\pi r \cdot \sin\beta} \cdot \cos\beta \cdot (\cos\alpha_1 - \cos\alpha_2)$$

т.е.

$$H_y = \int_{\beta_1}^{\beta_2} \frac{jd\beta}{4\pi} \cdot ctg\beta \cdot (\cos\alpha_1 - \cos\alpha_2) = \frac{j}{4\pi}(\cos\alpha_1 - \cos\alpha_2)\int_{\beta_1}^{\beta_2} ctg\beta \cdot d\beta =$$

$$= \frac{j}{4\pi}(\cos\alpha_1 - \cos\alpha_2)\ln\left|\frac{\sin\beta_2}{\sin\beta_1}\right| = \frac{I}{4\pi w}(\cos\alpha_1 - \cos\alpha_2)\ln\left|\frac{\sin\beta_2}{\sin\beta_1}\right|$$

В частном случае, если полоски с током считать бесконечно длинными, соотношения принимают вид:

$$H_x = \frac{j}{2\pi}(\beta_2 - \beta_1) = \frac{j}{2\pi}\left(\pi - arctg\frac{w_2}{R} - arctg\frac{w_1}{R}\right) = \frac{I}{2\pi w}\left(\pi - arctg\frac{w_2}{R} - arctg\frac{w_1}{R}\right)$$

$$H_y = \frac{j}{2\pi}\ln\left|\frac{\sin\beta_2}{\sin\beta_1}\right| = \frac{I}{2\pi w}\ln\left|\frac{\sin\beta_2}{\sin\beta_1}\right|$$

Рассмотрим расчет магнитного поля на оси широкой, тонкой, кольцевой полоски с равномерной плотностью тока (рис. 3).

Магнитное поле в плоскости полоска определяется для равномерной плотности тока соотношением:

$$H = \int_{r_1}^{r_2} dH = \int_{r_1}^{r_2} \frac{jdr}{2r} = \frac{j}{2}\ln\frac{r_2}{r_1} = \frac{I}{2(r_2 - r_1)}\ln\frac{r_2}{r_1} .$$

Полученные соотношения могут быть использованы и для расчета магнитного поля при неравномерном распределении плотности тока в поперечном сечении. В частности:

$$H = \int_{r_1}^{r_2} dH = \int_{r_1}^{r_2} \frac{j(r)dr}{2r}.$$

Например, при линейной функции распределения:

$$j(r) = Cr \text{ при } r_1 \le r \le r_2$$

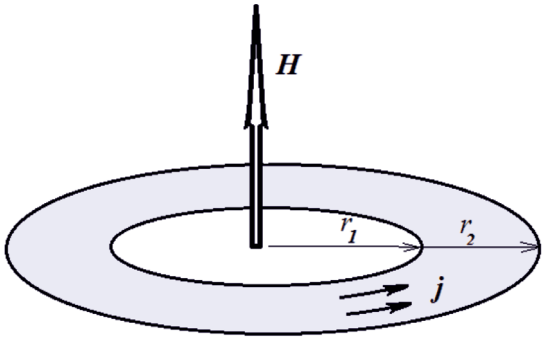

Рис. 3. Кольцевая полоска проводника с шириной полосы $r_2 - r_1$

имеем:

$$H = \int_{r_1}^{r_2} dH = \int_{r_1}^{r_2} \frac{Crdr}{2r} = \int_{r_1}^{r_2} \frac{Cdr}{2} = \frac{C}{2}(r_2 - r_1)$$

здесь постоянная С определяется из условия заданного значения тока, пропускаемого через полоску проводника:

$$I = \int_{r_1}^{r_2} jdr = \int_{r_1}^{r_2} Crdr = \frac{Cr^2}{2}\Big|_{r_1}^{r_2} = \frac{C\left(r_2^2 - r_1^2\right)}{2}.$$

Таким образом,

$$C = \frac{2I}{r_2^2 - r_1^2}.$$

И магнитное поле в центре кольца определяется соотношением:

$$H = \frac{I}{r_1 + r_2}.$$

Рассмотрим напряженность магнитного поля, создаваемого на оси широкой кольцевой полоски с током на расстоянии h от ее плоскости. Результирующее поле в силу симметрии структуры будет иметь только x составляющую, компонента

$$H_y = 0$$

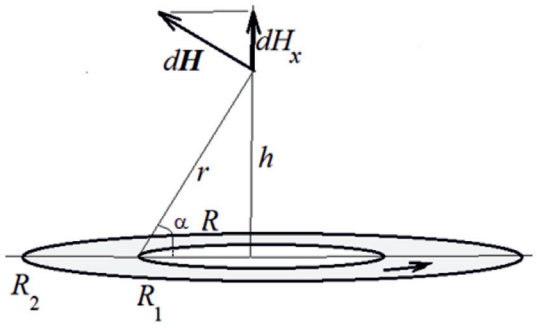

Рис. 4. Кольцевая полоска проводника с шириной $R_2 - R_1$

Компонента проекции магнитного поля на ось 0x равна:

$$dH_x = dH \cdot \sin\alpha,$$

где угол α определяется из прямоугольного треугольника (рис. 4):

$$\sin\alpha = \frac{h}{r} = \frac{h}{\sqrt{R^2 + h^2}},$$

Тогда компонента магнитного поля H_x может быть найдена из соотношения:

$$H_x = \int_{r_1}^{r_2} dH_x = \int_{r_1}^{r_2} dH \cdot \sin\alpha =$$

$$= \int_{R_1}^{R_2} \frac{jdR \cdot h}{2r \cdot r} = \int_{R_1}^{R_2} \frac{jdR \cdot h}{2\left(R^2 + h^2\right)}.$$

Интегрирование приводит к выражению для компоненты магнитного поля, перпендикулярной плоскости кольца:

$$H_x = \frac{j}{2}\left(arctg\frac{R_2}{h} - arctg\frac{R_1}{h}\right).$$

Или, выражая через силу тока в проводнике, ищем эту компоненту в виде:

$$H_x = \frac{I}{2(r_2 - r_1)}\left(arctg\frac{R_2}{h} - arctg\frac{R_1}{h}\right).$$

Заключение

Рассмотрена задача расчета магнитного поля, возбуждаемого широкими полосками проводника тока прямолинейной и кру-

говой формы. Получены аналитические решения для нескольких конфигураций, на основе которых возможно создание планарных элементов микро- и наноразмеров для различных диапазонов частот. Конечная ширина полосок с током влияет на распределение поля, что необходимо учитывать при проектировании функциональных элементов электроники. Увеличение ширины полосок тока в схемах в интегральном исполнении позволяет существенно увеличить допустимый уровень сигналов за счет увеличение уровня рассеяния тепловой энергии.

Список литературы

1. Савельев И.В. Курс общей физики. Т. 2. СПб.: Лань, 2021. 500 с.

2. Фризен В.Э., Черных И.В., Бычков С.А., Тарасов Ф.Е. Методы расчета электрических и магнитных полей. Екатеринбург: УрФУ, 2014. 176 с.

3. Wang S., Yong H., Zhou Y. Modified FFT-based method for the calculations of the thin superconductors with transport current. AIP Advances. 2021. V. 11. P. 035103.

4. Сапогин В.Г., Прокопенко Н.Н., Манжула В.Г. О расчёте коэффициента увеличения планарной индуктивности спирального типа // Фундаментальные исследования. 2013. № 11–6. С. 1150–1153.

5. Glushchenko A.A., Glushchenko A.G., Glushchenko E.P. Analytical solution of the problem of calculating a magnetic field in the center of a correct polygon. ISciense. 2019. no. 12 (56). P. 6–12.

УДК 004.89

О СОДЕРЖАТЕЛЬНОЙ ИНФОРМАЦИИ И РЕШЕНИИ ЗАДАЧ

Попов С.В.

ООО «Научно-внедренческая фирма БП+», Москва, e-mail: s-v-popov@yandex.ru

В связи с использованием искусственного интеллекта (ИИ) для моделирования когнитивной деятельности человека интерес представляет зависимость сложности решения задач от спецификации предметной области, в которой задачи формулируются. Встречаются просто формулируемые задачи со сложным решением, и наоборот. В связи с этим желательно иметь универсальный показатель сложности задачи, который определял бы сложность ее решения. Исходим из того, что всякая задача, так или иначе, предполагает разбиение множества объектов предметной области на классы, определяющие подзадачи. В статье предполагается, что все отношения эквивалентности над множеством объектов образуют частично упорядоченное множество. Чем сильнее эквивалентность, тем больше неопределенности устраняется при решении и тем компактнее само решение. Так как невозможно ввести числовой параметр, измеряющий силу эквивалентности, то предлагается представить порождение классов как результат анализа состава объектов. Тогда можно построить бинарное дерево с единственным корнем, каждый путь которого описывает в точности один объект. Если в этом дереве ввести склейку узлов, представляющих один класс, то, чем сильнее эквивалентность, тем больше склеек в дереве. И все пути, ведущие из корня в один узел, определяют один класс, связанный с уникальной процедурой, которую необходимо выполнить, чтобы найти общее решение задачи. После этого вводят параметр *неопределенности* бинарного дерева, полагая, что дуги, ведущие из одного узла в его непосредственные потомки, равновероятные. Такая неопределенность связана с неопределенностью того, какие процедуры будут использоваться при решении задачи. При известном частотном распределении объектов относительно классов можно подсчитать неопределенность как характеристику предметной области. Показывается, что эта неопределенность может служить единственным параметром, определяющим сложность предметной области, который помогает оценивать сложность процедуры, решающей исходную задачу.

Ключевые слова: задача, эквивалентность, энтропия, предметная область, перебор, вычисление, решение, бинарное дерево

ABOUT MEANINGFUL INFORMATION AND THE SOLVING PROBLEMS

Popov S.V.

LLC «Nauchno-vnedrencheskaya firma BP+», Moscow, e-mail: s-v-popov@yandex.ru

In connection with the use of artificial intelligence (AI) for modeling human cognitive activity, it is of interest to study the dependence of the complexity of problem solving on the specification of the subject area in which the tasks are formulated. There are simply formulated problems with a complex solution, and vice versa. In this regard, it is desirable to have a universal indicator of the complexity of the problem, which would determine the complexity of its solution. We proceed from the fact that every task, one way or another, involves splitting a set of objects of the subject area into classes that define subtasks. The article assumes that all equivalence relations over a set of objects form a partially ordered set. The stronger the equivalence, the more uncertainty is eliminated in the solution, and the more compact the solution itself. Since it is impossible to enter a numerical parameter measuring the strength of equivalence, it is proposed to present the generation of classes as a result of the analysis of the composition of objects. Then you can build a binary tree with a single root, each path of which describes exactly one object. If a gluing of nodes representing one class is introduced in this tree, then the stronger the equivalence, the more glues there are in the tree. And all paths leading to one node define one class associated with a unique procedure that must be performed to find a common solution to the problem. After that, the uncertainty parameter of the binary tree is introduced, assuming that the arcs leading from one node to its immediate descendants are equally probable. Such uncertainty is related to the uncertainty of which procedures will be used in solving the problem. With a known frequency distribution, the distribution of objects relative to classes can be calculated uncertainty as a characteristic of the subject area. It is shown that this uncertainty can serve as the only parameter determining the complexity of the subject area, which helps to assess the complexity of the procedure solving the original problem.

Keywords: problem, equivalence, entropy, subject area, iteration, calculation, solution, binary tree

В связи с широким проникновением искусственного интеллекта (ИИ) в различные сферы для моделирования когнитивной деятельности человека интерес представляет исследование сложности решения задач в зависимости от вида предметной области (ПО), в которой они формулируются. Чтобы установить такую связь, в первую очередь следует определить параметры, от которых зависит сложность ПО. Естественно, что это определяется формулируемой задачей, поскольку в одной ПО можно сформулировать несколько задач, и сложность их решения не всегда определяется сложностью формулировки. Встречаются просто формулируемые задачи со сложным решением, и наоборот. В связи с этим желательно иметь универсальный показатель сложности задачи в конкретной ПО, который определял бы сложность ее решения. Как представляется, такой показатель должен быть связан с содержательной сложностью задачи независимо от краткости или громоздкости ее формулировки. В настоящей статье предлагается подход, позволяющий ввести показатель

сложности задачи с опорой на ее содержательную формулировку.

Полагаем, что $\boldsymbol{O}$ есть множество объектов ПО. Каждый из них однозначно характеризуется конечным набором признаков. Например, это могут быть множество логических матриц с фиксированным значением n строк и m столбцов или множество образов, заданных своими элементами, где каждый элемент принадлежит конечному алфавиту. Одновременно с множеством $\boldsymbol{O}$ рассматривается множество $\boldsymbol{O}^*$ так называемых *недоопределенных* объектов, каждый из которых содержит лишь часть наборов признаков, характеризующих объекты. Например, если в качестве объектов рассматриваются матрицы с n строками, то в качестве недоопределенных объектов можно рассматривать матрицы с $n_1 < n$ строками. Если объекты – это некоторые образы, то недоопределенные объекты – суть их собственные части. Понятно, что каждый недоопределенный объект можно доопределить, в общем случае, разными способами до объекта ПО. Будем полагать, что если недоопределенный объект o^* доопределяется до объекта o, то это происходит в результате конкатенации o^* с некоторым недоопределенным объектом o_1^*, что будем обозначать $o = o^* o_1^*$. Введем пустой недоопределенный объект λ, который можно доопределить до любого объекта. Однако для объекта или недоопределенного объекта o верно равенство $o = o\lambda$. Поэтому полагаем, что справедливо включение $\boldsymbol{O} \subset \boldsymbol{O}^*$. И в дальнейшем будем употреблять общий термин *объект*, выделяя *недоопределенные объекты* по мере необходимости.

Полагаем, что решаемая задача Z над ПО с множеством $\boldsymbol{O}$ объектов состоит в установлении некоторого свойства P над $\boldsymbol{O}$. Поэтому задача Z определяет на множестве $\boldsymbol{O}^*$ отношение эквивалентности объектов $o_1, o_2 \in \boldsymbol{O}^* (o_1 \approx o_2 \Leftrightarrow \forall o^* \in \boldsymbol{O}^* (o_1 o, o_2 o \in \boldsymbol{O} \Rightarrow \mathrm{P}(o_1 o) = \mathrm{P}(o_2 o)))$. Однако два недоопределенных объекта $o_1, o_2 \in \boldsymbol{O}^*$ называются *разделяемыми*, если выполняется соотношение: $\exists o^* \in \boldsymbol{O}^* (o_1 o^*, o_2 o^* \in \boldsymbol{O} \Rightarrow \mathrm{P}(o_1 o^*) \neq \mathrm{P}(o_2 o^*))$. В этом случае o^* называется *разделяющим объектом*.

Каждый класс эквивалентности определяет свою собственную подзадачу, которую необходимо решить, чтобы получить искомое решение общей задачи. В логике это называется *разбором случаев*. Например, задачу над множеством целых чисел можно представить как две задачи: одна предполагает рассматривать четные числа в качестве новой ПО, другая – нечетные. Другой пример: решение числовых головоломок. В процессе решения происходит разбор случаев, каждый характеризуется уникальной комбинацией замен букв цифрами и требует своего решения. Поэтому исходная задача представляется совокупностью подзадач, решение которых приведет к окончательному результату.

Все отношения эквивалентности, которые можно ввести над множеством объектов ПО, образуют частично упорядоченное множество, по включению. Чем слабее эквивалентность, тем уже ее классы эквивалентности, т.е. тем слабее условия, при которых объекты эквивалентны. Самая сильная подразумевает эквивалентность всех объектов ПО. Самая слабая характеризуется классами, состоящими из одного объекта. В этом случае бессмысленно говорить о какой-либо связи объектов, задача не имеет содержательного смысла.

Для решения нетривиальной задачи над ПО обязательно наличие априорной информации о факторизации множества объектов. Это устраняет неопределенность, неизбежную при поиске решения и связанную с объединением в классы семантически совпадающих объектов. Чем сильнее отношение эквивалентности, т.е. чем шире ее классы, тем больше неопределенности устраняется при решении и тем компактнее решение. Например, при решении геометрической задачи, чем большим числом доказанных соотношений между фигурами располагаем, тем шире классы эквивалентности фигур на чертеже. И, как следствие, тем компактнее решение, так как всякое неизвестное, но обязательное для решения соотношение надо доказывать.

Пусть π есть всюду определенная арифметическая программа, вычисляющая предикат P. Если на ее вход поступает объект $o \in \boldsymbol{O}$, то программа завершает работу в точности в одном из двух финальных состояний: c_0 либо c_1, в зависимости от значения $P(o)$, соответственно 0 или 1. Если на вход программы поступает недоопределенный объект o^*, то он определяет одно вычисление $\pi(o^*)$, заканчивающееся в некотором, в общем случае, не финальном состоянии c^*. Поскольку π всюду определенная, то если на ее вход поступает любой объект, то ее вычисление завершается в некотором состоянии. Однако каждое вычисление программы π вызывается некоторым объектом o^*. Это вычисление может быть продолжено за счет различных вычислений, которые получаются, если объект o^* доопределять различными способами.

Следующим образом введем отношение π-*эквивалентности*: для объектов $o_1, o_2 \in \boldsymbol{O}^* (o_1 \approx \pi o_2 \Leftrightarrow \forall o^* \in \boldsymbol{O}^* (o_1 o^*, o_2 o^* \in O \Rightarrow$ вычисления $\pi(o_1 o^*)$ и $\pi(o_2 o^*)$ заканчива-

ются в одном состоянии). Очевидно, что для эквивалентных объектов из $\boldsymbol{O}$ финальные состояния совпадают. Отметим, что состояния, в которых завершаются вычисления $\pi(o_1)$ и $\pi(o_2)$, могут быть различными. Но при всяком одинаковом продолжении o^* объектов o_1 и o_2 до полностью определенных объектов эти продолжения заканчиваются всегда в одинаковом состоянии, т.е. объекты o_1o^* и o_2o^* одновременно либо обладают, либо не облают свойством P. В результате получаем следование: $\forall o_1o_2 \in \boldsymbol{O}^*(o_1 \approx \pi o_2 \Rightarrow o_1 \approx o_2)$.

Покажем, что верно и обратное следование. Пусть два недоопределенных объекта o_1, $o_2 \in \boldsymbol{O}^*$ находятся в отношении $o_1 \approx o_2$, и вычисления $\pi(o_1)$ и $\pi(o_2)$ заканчиваются в состояниях соответственно c_1 и c_2. Допустим, что имеется такое продолжение o^*, что o_1o^*, $o_2o^* \in \boldsymbol{O}$, но из этих двух состояний c_1 и c_2 можно продолжить вычисления в разные финальные состояния. Следовательно, имеется одно продолжение для эквивалентных объектов, которое переводит два состояния c_1 и c_2 в два разных финальных состояния. Следовательно, продолжение o^* – разделяющее, поэтому объекты o_1, o_2 не эквивалентные. Это противоречие. Следовательно, верно соотношение: $\forall o_1o_2 \in \boldsymbol{O}^*(o_1 \approx o_2 \Rightarrow o_1 \approx \pi o_2)$.

Полагаем, что каждый объект из множества $\boldsymbol{O}$ представляется бинарным вектором, а недоопределенные объекты представляются векторами над базисом <0, 1, _>. Здесь символом _ обозначены неизвестные или несущественные признаки. Поэтому все объекты ПО можно представить в виде бинарного дерева T с единственным корнем, в котором каждый узел имеет двух непосредственных потомков, один порождается дугой с меткой 0, другой – 1. Каждый путь из корня описывает в точности один, в общем случае, недоопределенный объект. Пути из корня, заканчивающиеся в висячих узлах, определяют объекты из множества $\boldsymbol{O}$.

Теперь, если в бинарном дереве ввести склейку узлов, представляющих один класс, то сила эквивалентности определяет число склеек. Чем сильнее эквивалентность, тем больше склеек в дереве. В результате висячие узлы любого поддерева, содержащего корень, представляют классы, объекты которых определяются путями, ведущими в один узел. А все пути, ведущие в один узел, определяют один класс. Последний определяет подзадачу, решение которой необходимо в рамках общего решения задачи. Однако невозможно ввести числовой параметр, который измерял бы силу априорно известной эквивалентности объектов ПО, поскольку эквивалентности образуют частично упорядоченное множество по включению. Тем не менее, силу эквивалентности можно охарактеризовать количеством склеек в бинарном дереве: чем больше склеек, тем эквивалентность сильнее.

Поскольку один узел определяет класс эквивалентности объектов, то сечение $\boldsymbol{C}$ дерева T определяет классы эквивалентности объектов, характеризующихся фиксированными наборами параметров. Тогда множество $\boldsymbol{C}$ определяет вычисления программы π, оканчивающиеся, по меньшей мере, в $|\boldsymbol{C}|$ различных состояниях. Для кодирования этих состояний в программе π требуется память не меньшая, чем $log_2|\boldsymbol{C}|$ бит [1–4]. Однако все объекты из $\boldsymbol{C}$ не могут быть закодированы c меньшей, чем $log_2|\boldsymbol{C}|$, сложностью. Следовательно, в программе π имеется вычисление длины не меньше $|\boldsymbol{C}|$. Так удалось оценить память арифметической программы в зависимости от максимального сечения бинарного дерева, представляющего объекты ПО.

Введем параметр *неопределенности* бинарного дерева, полагая, что дуги, ведущие из одного узла в непосредственные потомки, равновероятные. Неопределенность в таком случае связана с неопределенностью того, какие подзадачи будут решаться при решении общей задачи. Известно число подзадач, известно частотное распределение, пусть p_1, p_2, …, p_q, объектов ПО, с которым они попадают в q классов, т.е. используются в качестве аргументов подзадач. Следовательно, можно подсчитать неопределенность H совокупности этих классов как характеристику исходной задачи над ПО [5].

Покажем, что так введенная неопределенность может служить параметром, определяющим сложность задачи. Для этого свяжем неопределенность задачи как неопределенность разбиения объектов ПО со сложностью вычислений программы, решающей исходную задачу. Здесь исходим из следующих соображений. Предполагается, что каждый класс определяет совокупность вычислений программы р, решающей исходную задачу, т.е. вычисляющей предикат P. Все вычисления программы представляются так называемой *разверткой*, т.е. бинарным деревом с единственным корнем, в котором вычислительные узлы не вызывают размножения вычислений, а логические представляются узлами с двумя потомками.

Связь между бинарным деревом разбора объектов ПО и разверткой устанавливается исходя из следующих соображений. Если в бинарном дереве разбора ПО имеются два узла, пусть n_0 и n_1, определяющие разные классы объектов и, соответственно, различные подзадачи, то им в развертке $R\pi$

программы π соответствуют два различных множества вычислений. Если допустить, что это не так, то приходим к заключению, что эквивалентность, определенную на множестве объектов ПО, можно усилить, объединив классы, соответствующие узлам n_0 и n_1. В результате каждому поддереву T бинарного дерева разбора объектов ПО в развертке Rp соответствует бинарное дерево вычислений с корнем, такое, что каждый узел $n \in T$ определяет множество $\boldsymbol{M}_n$ вычислений. И разные узлы дерева T определяют разные множества вычислений. Тогда, если H есть неопределенность дерева T, то соответствующее бинарное поддерево развертки $R\pi$ имеет глубину не меньше, чем H, а число различных вычислений не меньше 2^H. Так можно оценить как размер памяти программы π, так и число вычислений. Исходя из базиса арифметической программы оценивается сложность вычисления. Так, если базис состоит из логического оператора = и арифметических $s(x)$ – прибавления единицы и $0(x)$ – обнуления переменной, то средняя длина вычисления, определяемого поддеревом разбора ПО с неопределенностью H, не менее 2^H.

Информационные модели. Считаем, что объект управления описывается уравнением $\boldsymbol{y} = f(\boldsymbol{x})$, где $\boldsymbol{x}$ – это входные параметры, $\boldsymbol{y}$ – демонстрируемое поведение. Тогда его информационной моделью могут служить, например, таблицы базы данных, описывающие его поведение, или программа, вычисляющая функцию $f(\boldsymbol{x})$, и т.п.

***Пример* 1**. Пусть информационная модель отражает совместимость комплектующих при сборке изделия с заданным функционалом. Если она адекватна, то удастся избежать сборки неправильно функционирующего изделия.

***Пример* 2**. Пусть информационная модель отражает совместимость химических компонентов при синтезе вещества с заданными качествами. Если она не адекватна, то возможен синтез вещества с незапланированными качествами.

Сформулируем некоторые аналогии между понятиями теории информации и введенными представлениями. *Передатчику* информации соответствует *наблюдаемая часть модели*, а *сигналу* – поведение этой части, выраженное в тех или иных терминах. *Канал* связи соединяет наблюдаемую часть модели с частью, на которую осуществляется воздействие. Приемником информации служит еще не наблюдаемая часть модели, на которую воздействует активная, функционирующая часть. И в зависимости от сигнала приемник переходит в какое-либо состояние.

***Пример* 3**. Пусть M есть логическая матрица размерности $[n \times m]$, $M = M_1 + M_2$, M_1 имеет размерность $[n_1 \times m]$, $M_2 - [n_2 \times m]$, $n = n + n_2$. *Воздействиями* на модель является множество Σ всех означиваний n_1 строк матрицы M_1. Эта подматрица есть передатчик, который преобразует воздействия в сигналы. Каналом является горизонталь, разделяющая матрицы M_1 и M_2. Приемником является матрица M_2. Каждое воздействие $\sigma \in \Sigma$ превращает в истину часть $H\sigma$ столбцов матрицы M_1. Множество истинных столбцов $H\sigma$ есть сигнал, передающийся от передатчика приемника. Приемник реагирует на такой сигнал однозначно, при последующем рассмотрении эти столбцы вычеркиваются из матрицы M_2. В результате выходные сигналы канала воздействуют на приемник, изменяя его состояние. Таким образом, воздействия Σ формируют выходные сигналы, а именно множество подматриц матрицы M_2.

Здесь под моделью понимается дискретная функция $\varphi: \boldsymbol{A} \rightarrow \boldsymbol{B}$, где $\boldsymbol{A}$ есть множество входных воздействий, $\boldsymbol{B}$ – выходных. Значения ее аргумента $\boldsymbol{x}$ представляют собой бесконечные бинарные слова. Каждое слово представляется конкатенацией конечной префиксной части, заканчивающейся самой правой единицей, и нулевым бесконечным суффиксом. Говорим, что двоичная последовательность, означивающая часть аргументов $\boldsymbol{x}$, является *частичным* $\boldsymbol{x}$-набором. Аналогично будем говорить о *частичном* $\boldsymbol{y}$-наборе выходных значений. Анализ поведения функции φ сводится к проверке равенства $\varphi(\sigma_x) = \sigma_y$ для всякой пары (σ_x, σ_y) соответственно (в том числе частичных) $\boldsymbol{x}$-набора и $\boldsymbol{y}$-набора.

Допустим, что при всяком допустимом означивании подмножества $\boldsymbol{x}' \subset \boldsymbol{x}$ входных параметров известна выходная реакция модели. В результате модель φ представляется объединением $\varphi_1, \varphi_2, \ldots, \varphi_q$ подмоделей, число которых не больше числа частичных $\boldsymbol{x}$-означиваний. Следовательно, анализ поведения модели в зависимости от частичных входных параметров $\boldsymbol{x}'$ сводится к двум задачам.

1. Выяснить тип поведения из известных $\varphi_1, \varphi_2, \ldots, \varphi_q$, демонстрируемый при известном входном векторе σ_x.

2. Исследовать каждую из подмоделей $\varphi_1, \varphi_2, \ldots, \varphi_q$.

Пусть событие A встречается с частотой p_A и $\gamma(A) = -\log_2 p_A$ есть показатель его неожиданности, зависящий от частоты p_A. Величину $\gamma(A)$ назовем *частной неопределенностью* H_A *события* A. Если имеются q различных значений $A_1, A_2, \ldots, A_q$ одной случайной величины, каждое из которых

встречается с частотой соответственно p_1, p_2, …, p_q, то *усредненная неопределенность* равна $H = -\Sigma_{i=1,q} p_i \gamma(A_i)$.

Если частичный набор σ_x' означивает часть аргументов $\boldsymbol{x}' \subset \boldsymbol{x}$, то аргументы $\boldsymbol{x}'$ назовем *неподвижными*, а оставшиеся – *изменяемыми*. При означиваниях изменяемых аргументов так, что результирующие наборы принадлежат области определения *Def* φ, какие-то компоненты $\boldsymbol{y}' \subset \boldsymbol{y}$ могут принимать значения, не меняющиеся при означивании оставшихся компонентов аргумента. В этом случае $\boldsymbol{x}$-набор σ_x' определяет функцию $\varphi|\sigma_x'(\boldsymbol{u})$:

1) переменные $\boldsymbol{u}$ суть изменяемые аргументы;

2) область определения *Def* $\varphi|\sigma_x'$ получается в результате ограничения области *Def* φ означиваниями, $\boldsymbol{x}'$-проекции которых совпадают с σ_x';

3) функция $\varphi|\sigma_x'(\boldsymbol{u})$ получается из функции φ при аргументах с фиксированным частичным $\boldsymbol{x}$-означиванием σ_x' игнорированием неподвижных $\boldsymbol{y}$-компонентов.

***Пример* 4**. Пусть $\sigma_x = 01011 = 26_{10}$ есть частичное x-означивание аргументов двоичной функции прибавления единицы: $\varphi(x)=x+1$. Полагаем, что младшие разряды располагаются слева. Тогда функция $\varphi|\sigma_x'(\boldsymbol{u}) = 2^5\boldsymbol{u}$, поскольку неподвижные компоненты значений функции суть компоненты с нулевого до четвертого включительно, а остальные – изменяемые, и отсутствует перенос в пятый разряд. Все значения функции $\varphi(x)$ имеют своим префиксом 11011, если ограничить область ее определения двоичными аргументами с префиксом 01011.

Говорим, что $\boldsymbol{x}$-наборы σ'_x и σ''_x *эквивалентные относительно функции* φ, если функции $\varphi|\sigma'_x'(\boldsymbol{u})$ и $\varphi|\sigma''_x'(\boldsymbol{u})$ совпадают.

***Пример* 5**. Для функции $\varphi(\boldsymbol{x}) = \boldsymbol{x} + 1$ означивания: $\sigma'_x = 01011$, $\sigma''_x = 11111$, $\sigma'''_x = 11011$ определяют: $\varphi|\sigma_x'(\boldsymbol{u}) = 2^5\boldsymbol{u}$, $\varphi|\sigma_x'''(\boldsymbol{u}) = 2^5\boldsymbol{u}$, $\varphi|\sigma_x''(\boldsymbol{u}) = 2^5\boldsymbol{u}+1$. Префиксы значений функции $\varphi(\boldsymbol{x})$ суть следующие: 11011, если префикс аргумента σ'_x; 0000, если префикс σ''_x; 00111, если префикс σ'''_x. При σ''_x возник перенос в пятый разряд, в первом и третьем случаях переноса нет. Поэтому функция $\varphi|\sigma_x''(\boldsymbol{u})$ отлична от $\varphi|\sigma_x'(\boldsymbol{u})$ и $\varphi|\sigma_x'''(\boldsymbol{u})$, наборы σ'_x и σ'''_x эквивалентны, а σ''_x не эквивалентен им.

Эквивалентность $\boldsymbol{x}$-наборов сводит исследование исходной модели к описанию совокупности подмоделей. Число m классов определяет m различных подмоделей. И, если каждая подмодель предполагает собственное вычисление, то имеется m различных вычислений. Следующий пример демонстрирует содержательные рассуждения и введенные определения.

***Пример* 6**. Пусть $\pi(\boldsymbol{x})$ есть двоичная арифметическая программа, представленная своей диаграммой с двумя типами узлов: логических и арифметических. Тогда ее вычисления задаются входными значениями. Под *вычислением* понимаем последовательность арифметических и логических операторов, выполняющихся при заданном входе до остановки. В результате выходная переменная $\boldsymbol{y}$ принимает определенное значение.

Полагаем, что означивания σ_1 и σ_2 части $\boldsymbol{x}'$, $|\boldsymbol{x}'| = l$, аргумента эквивалентные ($\sigma_1 \approx \sigma_2$), если определяемые ими вычисления завершаются в одном и том же состоянии. Поэтому вычисления разбиваются на классы, определяемые эквивалентными означиваниями. И каждое частичное означивание определяет подпрограмму, выполняемую после продолжения этого означивания. Пусть $\pi_1, \pi_2, \ldots, \pi_q$ суть выделяемые подпрограммы, множество вычислений, определяющих подпрограмму π_i, обладает долей p_i, $i = 1, 2, \ldots, q$ среди всех вычислений над частичными входами длины l. Тогда неопределенность вопроса: какое событие из $\pi_1, \pi_2, \ldots, \pi_q$ наступит при означивании σ_x части $\boldsymbol{x}'$ аргументов, $|\boldsymbol{x}'| = l$, зависит лишь от долей $p_1, p_2, \ldots, p_q$, и равна $H_1 = -\Sigma_{i=1,q} p_i log_2 p_i$.

Исследование программы как совокупности $\pi_1, \pi_2, \ldots, \pi_q$ подпрограмм, обладающих собственными множествами вычислений, соответственно $T_1, T_2, \ldots, T_q$, $i = 1, 2, \ldots, q$, обладает неопределенностью $H_2 = \Sigma_{i=1,q} p_i log_2 |T_i'|$.

Понятно, что исследование программы приводит к разным значениям неопределенности. Если $l = 0$, то неопределенность максимальная, поскольку не известны возможные вычисления. Когда известны все вычисления, то неопределенность равна нулю. В случае частичных означиваний, $0 < l < n$ неопределенность равна сумме $H_1 + H_2$.

Обобщим предыдущие рассуждения следующим образом.

Пусть *Tree* есть растущее вниз однокорневое бинарное дерево, каждый висячий узел которого обозначает некоторый класс эквивалентности объектов ПО, и его энтропия равна H. Полагаем, что введенная эквивалентность определяет семантику ПО, когда одинаковые объекты суть элементы одного класса эквивалентности. Бинарное дерево с минимальной длиной пути из корня в листья и энтропией H имеет глубину H и в точности 2^H висячих узлов.

Свяжем энтропию дерева *Tree* со сложностью вычислительной процедуры, ко-

торая не различает эквивалентные объекты, выдавая для них одинаковые ответы. Для этого допустим, что имеется некоторая вычислительная процедура $\pi(\boldsymbol{x})$, вычисляющая некоторую двоичную всюду определенную функцию $\boldsymbol{y} = f(\boldsymbol{x})$ от одного аргумента. Полагаем, что в процессе вычисления процедура π переходит из одного состояния в другое, и ее состояния представляются *внутренней памятью*, которая есть бинарный вектор неограниченной длины. При отсутствии входа процедура находится в фиксированном нулевом состоянии, когда память заполнена нулями. Каждому состоянию соответствует уникальный вид этого вектора с ненулевым ограниченным префиксом и бесконечным нулевым суффиксом. Каждый вход σ либо приводит к остановке процедуры в *финальном состоянии End*, либо процедура достигает промежуточного состояния *S*, из которого она продолжит вычисление при расширении входа новыми компонентами. Тем самым все входные последовательности разбиваются на классы эквивалентности: с каждым классом связано уникальное состояние процедуры, из которого она продолжает вычисление при расширении аргумента. И для разных классов продолжения вычисления различные.

Представим вычисления процедуры $\pi(\boldsymbol{x})$ в виде следующего бинарного дерева *Tree*(π) (развертки). Корнем дерева служит состояние процедуры при пустом входе. Полагаем, что в корень ведет дуга, помеченная пустой меткой. Если некоторый узел *n* развертки представляет состояние *S* вычисления, полученное в результате подачи на вход процедуры бинарной последовательности σ, то из него выходят две дуги, помеченные 0 и 1. Каждая из них ведет в узел соответственно n_0 и n_1, представляющие собой состояния, в которые переходит процедура π при входах соответственно σ0 и σ1. Одинаковые состояния развертки склеиваются.

Пусть на вход процедуры π поступают все бинарные последовательности длины *l*. Часть ее развертки *Tree*(π, *l*), представляющая собой начальное поддерево с длиной ветвей *l*, определяет эквивалентность на множестве этих входов: два входа эквивалентные, если соответствующие вычисления приводят к одному состоянию. Исходя из этого и полагая равновероятными дуги, ведущие из одного узла развертки и помеченные 0 и 1, подсчитаем энтропию *H* дерева *Tree*(p, *l*). Тогда минимальная глубина двоичного дерева с энтропией *H* равна *H*, а число его висячих узлов – 2^H.

Так как каждый висячий узел определяет одно состояние вычисления, то состояний внутренней памяти будет не менее 2^H. Следовательно, чтобы реализовать все вычисления, представленные деревом *Tree*(π, *l*), необходима внутренняя память, содержащая не меньше *H* компонентов. В противном случае не удастся получить все классы эквивалентности входов процедуры.

В данном случае эквивалентность входов процедуры π определялась исходя из состояний самой процедуры. Также исходя из этого строилась развертка и определялась энтропия частичной развертки. Однако при изучении конкретной ПО нет априорной информации о поведении вычислительной процедуры. Поэтому интерес представляет такое определение эквивалентности над объектами ПО, чтобы разные классы определяли различные вычисления. Тогда по мощности фактор множества можно оценить размер памяти вычислительной процедуры, а затем исходя из особенностей ее вычислительных правил оценить сложность вычислений.

Исходя из этого рассмотрим вопрос, как выглядит неопределенность модели в зависимости от различных определений эквивалентности ее состояний при частичном означивании аргумента. В качестве примера рассмотрим формулу разложения логических функций (по переменной *x*) $f(x) = x \wedge f(1) \vee \overline{x} \wedge f(0)$, полагая, что логическая функция задана матрицей, представляющей ее КНФ. Тогда разложение по нескольким переменным можно представить бинарным деревом, в котором узлы суть логические матрицы, и при его построении матрицы, определяющие эквивалентные (в каком-либо смысле) функции, склеиваются. Эквивалентности в данном случае могут быть различные. Это могут быть: логическая эквивалентность, совпадение матриц с точностью до перестановки строк и столбцов, логическая эквивалентность, доказываемая за ограниченное число преобразований, буквальное совпадение матриц и т.д. Понятно, что все эквивалентности, не противоречащие формуле разложения, формируют частично упорядоченное множество с минимальным элементом – логической эквивалентностью (самой сильной) и максимальным – буквальным совпадением матриц (самой слабой). При различных эквивалентностях получаются различные бинарные деревья. При более сильной эквивалентности будет больше склеиваний, нежели при слабой.

Пусть при одном порядке переменных разложения исходной функции и при различных эквивалентностях E_1 и E_2, $E_1 \leq E_2$ получаются два бинарных дерева T_1 и T_2.

И при одной последовательности $\boldsymbol{y}$ разложения переменных получаются два сечения S_1 и S_2 бинарных деревьев T_1и T_2. Тогда энтропии H_1 и H_2, определяемые сечениями соответственно S_1 и S_2, находятся в отношении: $H_1 \le H_2$. Следовательно, более сильная эквивалентность определяет меньшую энтропию, что вполне объяснимо: более сильная эквивалентность за счет более частой склейки узлов в бинарном дереве определяет меньшую энтропию. Содержательно это трактуется следующим образом: эквивалентность, подразумеваемая в разложении функции, устраняет часть неопределенности за счет дополнительных свойств рассматриваемых объектов. Поэтому неопределенность бинарного дерева меньше. Понятно, что так введенная энтропия касается только вида бинарного дерева и не учитывает сложности вычисления эквивалентностей. А поэтому ничего нельзя сказать о реальной сложности вычислений, не располагая знаниями о сложности вычисления эквивалентности.

Список литературы

1. Шеннон К. Работы по теории информации и кибернетике. М.: ИЛ, 1963. 830 с.

2. Кудряшов Б.Д. Теория информации. СПб.: Изд-во Питер, 2018. 312 с.

3. Леонтьев В.К., Гордеев Э Н. Комбинаторные аспекты теории информации. М.: МФТИ, 2019. 320 с.

4. Фурсов В.А. Лекции по теории информации: учеб. пособие / Под ред. Н.А. Кузнецова. Самара: Изд-во Самар. гос. аэрокосм. ун-та, 2016. 148 с.

5. Попов С.В. Логическое моделирование. М.: Тровант, 2016. 255 с.

УДК 004.738

ЗАЩИТА УСТРОЙСТВ ИНТЕРНЕТА ВЕЩЕЙ (IOT) С ПОМОЩЬЮ БЛОКЧЕЙН-ФРЕЙМВОРКА HYPERLEDGER FABRIC

Гаранин Н.А., Белов Ю.С.

ФГБОУ ВО «Московский государственный технический университет имени Н.Э. Баумана», филиал, Калуга, e-mail: fn1-kf@mail.ru

Данная статья посвящена вопросу защиты IoT и оценки блокчейн-фреймворка Hyperledger Fabric. Интернет вещей (IoT) стал неотъемлемой частью жизни человека. С ростом популярности данной технологии возрастает её уязвимость и риск эксплуатации. Проблемы безопасности привели к возникновению дополнительных проблем, таких как совместимость и вычислительная мощность устройств интернета вещей. Для решения вопросов безопасности используется технология блокчейн, из-за ее безопасного дизайна и децентрализации. В качестве решения безопасности IoT рассмотрен фреймворк Hyperledger, а также основные компоненты, обеспечивающие работу Hyperledger Fabric. В этом исследовании проводится оценка структур Blockchain и Hyperledger, защищающих устройства интернета вещей. Основное внимание уделяется рассмотрению структуры взаимодействия данного фреймворка с интернетом вещей и безопасностью в целом. Транзакционные отношения между участниками сети должны оставаться конфиденциальными и невидимыми для всех. Это стало возможным благодаря функции «каналы» в структуре Hyperledger. Hyperledger является разрешенным блокчейном, который имеет конфиденциальность и целостность, встроенные в дизайн. Данная технология организует полную изоляцию транзакций и личных данных. При совместном использовании хэшей в качестве доказательств транзакций личные данные могут быть переданы участникам «коллекции» или конкретной организации на основе конфиденциальности данных.

Ключевые слова: блокчейн, интернет вещей, Hyperledger Fabric, безопасность

INTERNET OF THINGS (IOT) DEVICE PROTECTION USING THE HYPERLEDGER FABRIC BLOCKCHAIN FRAMEWORK

Garanin N.A., Belov Yu.S.

Bauman Moscow State Technical University, Kaluga branch, Kaluga, e-mail: fn1-kf@mail.ru

This article is devoted to the issue of IoT protection and evaluation of the Hyperledger Fabric blockchain framework. The Internet of Things (IoT) has become an integral part of human life. With the growing popularity of this technology, its vulnerability and the risk of exploitation increases. Security issues have led to additional issues, such as compatibility and computing power of Internet of Things devices. Blockchain technology is used to solve security issues, because of its secure design and decentralization. As an IoT security solution, the Hyperledger framework is considered, as well as the main components that ensure the operation of Hyperledger Fabric. This study evaluates the Blockchain and Hyperledger structures that protect IoT devices. The main attention is paid to the structure of the interaction of this framework with the Internet of Things and security in general. Transactional relationships between network participants should remain confidential and invisible to everyone. This was made possible thanks to the «channels» function in the Hyperledger structure. Hyperledger, is an authorized blockchain that has privacy and integrity built into the design. This technology organizes the complete isolation of transactions and personal data. When hashes are shared as proof of transactions, personal data can be transferred to the participants of the «collection» or a specific organization based on data confidentiality.

Keywords: blockchain, Internet of Things, Hyperledger Fabric, security

Интернет вещей (IoT) относится к широкому спектру интеллектуальных объектов, объединенных в одну большую сеть и связанных интернетом. Данные объекты состоят из датчиков и микроконтроллеров, которые собирают и обрабатывают информацию из окружающей среды и обеспечивают работу их функций с помощью встроенного программного обеспечения. Объектами интернета вещей можно считать носимые устройства, умный дом, умный город, системы здравоохранения, транспорт, промышленность. Из-за роста охвата интернета вещей возникает проблема взаимосвязанности устройств и, как следствие, образуются проблемы с безопасностью и защитой конфиденциальности. Ученые прогнозируют, что в 2025 г. в IoT будет включать более 40 млрд устройств, которые будут генерировать более 75 млрд зеттабайт данных.

Цель исследования: рассмотреть способы решения вопросов безопасности для объектов интернета вещей, взаимодействующих со структурами Blockchain и Hyperledger Fabric.

Описание модели. Объекты интернета вещей разделены на пятислойную пирамиду, как показано на рис. 1. Устройства пятого уровня содержат небольшой объем памяти, ограниченные вычислительные мощности и, следовательно, более подвержены риску кибер-атак и взломов [1].

В пирамиде интернета вещей на верхнем уровне находятся облачные серверы, это централизованные системные контроллеры, принадлежащие третьим сторонам, таким как Microsoft, Amazon, IBM. Структура IoT

в основном зависит от архитектуры централизованного облачного сервера, от способа хранения данных, аутентификации, связи и любых других дополнительных услуг. Использование централизованных решений интернета вещей сопряжено с проблемами безопасности и доверия, поскольку пользователи должны доверять надлежащей обработке данных и исключению публичного доступа к данным.

Уязвимости в системе безопасности объектов интернета вещей при совместном использовании могут привести к сбоям. Примером такого сбоя может послужить случай, произошедший в феврале 2020 г. Была произведена распределенная атака типа «Отказ в обслуживании» (DDoS) против критически важного коммерческого публичного облака, Amazon Web Services, в результате которой были отключены множество сервисов. Эти типы атак становятся все более частыми, и их трудно исправить, так как трудно определить, какая из систем неисправна, в некоторых случаях злоумышленники используют каналы связи или пароли по умолчанию.

Достижения в области информационно-коммуникационных технологий способствовали эволюции традиционной компьютерной индустрии в интеллектуальную индустрию, основанную на принятии решений на основе данных [2]. Во время этого сдвига парадигмы интернет вещей (IoT) играет важную роль в подключении физической промышленной среды к киберпространству вычислительных систем, в результате чего формируется Киберфизическая система (CPS). IoT может поддерживать широкий спектр промышленных приложений, таких как производство, логистика, пищевая промышленность и коммунальные услуги. IoT нацелен на повышение эффективности работы и производительности производства, сокращение времени простоя оборудования и повышение качества продукции. В частности, IoT обладает следующими особенностями:

1) децентрализация систем IoT;
2) разнообразие устройств и систем IoT;
3) неоднородность данных IoT;
4) сложность сети.

Рис. 1. Пятислойная пирамида

Все они приводят к проблемам, включая неоднородность системы интернета вещей, плохую совместимость, ограниченность ресурсов устройств интернета вещей, уязвимость конфиденциальности и безопасности.

Существующие проблемы в IoT могут быть решены с помощью децентрализованных архитектур, которым присущи безопасность и конфиденциальность. К таким архитектурам можно отнести Blockchain и Hyperledger [3]. При их применении к устройствам интернета вещей может обеспечиться повышение конфиденциальности и целостности по сравнению с нынешними подходами.

Технология Blockchain. Блокчейн – это децентрализованный распределенный и прозрачный глобальный реестр записанных данных, защищенный от несанкционированного доступа, позволяющий хранить любые цифровые активы. Данные цепочки стали популярными в 2008 г. в связи с появлением биткоина и криптовалют [4]. В современных блокчейн-технологиях, таких как Ethereum, они используют смарт-контракты, которые представляют собой программируемый код, который может выполняться без участия третьих сторон.

Благодаря децентрализации блокчейны могут обеспечить выполнение транзакции и проверку в распределенной системе, которой не проверяются друг другом, без вмешательства доверенной третьей стороны. В отличие от существующих систем управления транзакциями, в которых централизованное управление должно проверять транзакции, блокчейны могут обеспечивать децентрализованную проверку транзакций, тем самым значительно экономя затраты и снижая узкое место в производительности центрального агентства. Более того, каждая транзакция, сохраненная в блокчейнах, по сути, неизменна, поскольку каждый узел в сети хранит все зафиксированные транзакции в блокчейне.

В блоки записывают данные и дублируют их по распределенной сети, новые записи транзакций добавляются в конец коллекции, каждая запись данных сохраняется в виде блока, который связан со следующим набором данных, создающим блокчейн. Цепочки блоков поддерживаются и проверяются участниками сети, называемой узлами.

Распределение блокчейн-регистров по узлам обеспечивает прозрачность, поскольку используется одна и та же информационная цепочка по всей сети. Участвующие узлы в сети выполняют проверку достоверности добавленных блоков в сети, принимая и отклоняя записи, используя методы проверки набора, известные как механизмы консенсуса. Механизмы консенсуса – это криптографические алгоритмы, используемые для обеспечения правильного упорядочения транзакций в блоках.

Блокчейн использует одноранговую сеть (P2P) в качестве своей сетевой архитектуры, которая удовлетворяет цели децентрализации цепочки данных. В P2P-сети участники могут обмениваться информацией и общаться без необходимости какого-либо центрального управления. Участникам блокчейна выделяются ключи, которые генерируются криптографически, в отличие от систем учетных данных для входа в централизованных архитектурах [5]. Функция криптографии предоставляет адрес блокчейна и код закрытого ключа, которые специфичны только для этого пользователя, обеспечивая пользователям анонимность, поскольку ни один код ключа не похож, и трудно связать адрес с участником.

Каждый блок содержит заголовок блока, в котором уникальный хэш, называемый корнем, идентифицирует текущий блок, хэш для предыдущего блока и список транзакций с новыми транзакциями также находится в блоке. Блоки будут содержать одинаковое количество транзакций в структуре данных. Однако у разных пользователей будут разные транзакции. Для удобства идентификации разным блокам при создании выдается временная метка, которая обычно отличается. Блок зарождения – это первый блок в цепи, все предыдущие блоки которого связаны с предыдущими. В некоторых случаях существуют блоки, которые могут ссылаться на одного и того же предшественника, если они созданы примерно в одно и то же время, это называется разветвлением, как показано на рис. 2.

Фреймворк Hyperledger. Hyperledger – это проект с открытым исходным кодом, созданный для работы с технологией блокчейн, который предоставляет несколько отличных платформ для работы с межотраслевыми реестрами [6]. IBM и другие компании поддерживают проект Hyperledger. На 2021 г. Hyperledger объединяет шесть различных распределенных реестров.

Активные на данные момент:

– Hyperledger Fabric – технология распределенного регистра корпоративного уровня с поддержкой конфиденциальности.

– Hyperledger Besu – Ethereum на базе Java.

– Hyperledger Sawtooth – основной участник – Intel, работающий над настройкой протоколов.

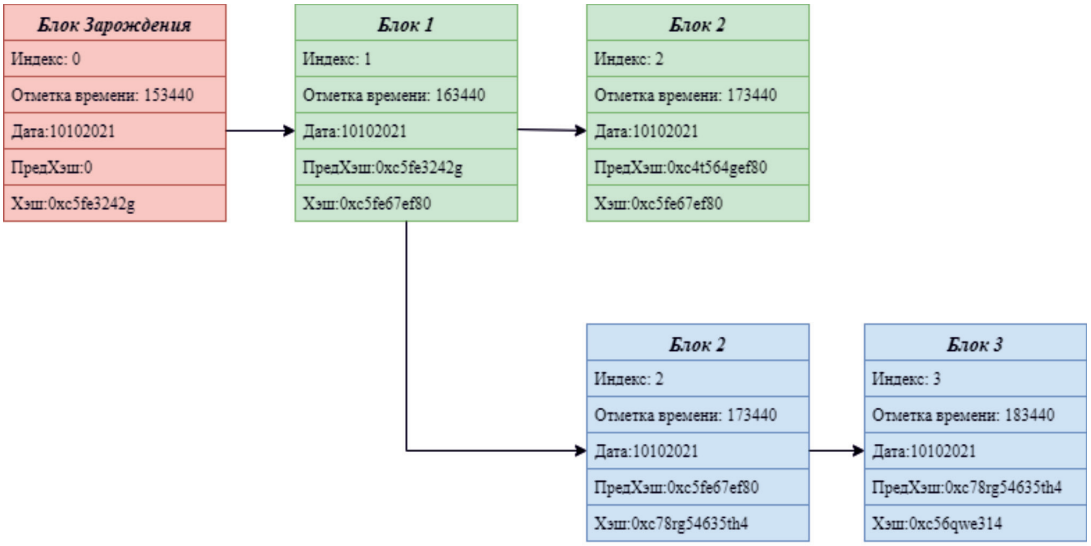

Рис. 2. Иллюстрация простой блокчейн-цепочки

В планах:

– HyperledgerBurrow–смарт-контрактов.

– Hyperledger Indy – инструмент и библиотека для запуска независимых идентификаций в распределенных реестрах.

– Hyperledger Iroha – ориентирован на мобильные приложения и имеет ту же кодовую базу, что и Fabric.

Hyperledger также предоставляет список инструментов и библиотек, которые можно использовать с различными платформами. Основными инструментами в проекте Hyperledger, которые представляют интерес для работы с IoT, а в частности с блокчейн, являются:

– Hyperledger Cello – этот инструмент помогает обеспечить проверку производительности развернутого блокчейна с помощью набора сценариев.

– Hyperledger Caliper – этот инструмент помогает развертывать решения в блокчейн-системах, что приводит к сокращению объема работы за счет предоставления автоматизированных способов создания, завершения и управления цепочками блоков.

– Hyperledger Explorer – с помощью этого инструмента можно вызывать, развертывать и запрашивать блоки.

Блокчейн Hyperledger Fabric. Данная технология объединяет набор узлов в организации, которые создают сеть, взаимодействующую с внешними приложениями. Организации рассматриваются как участники, которые являются частью блокчейн-сети. Каждая организация идентифицируется по идентификатору Membership Service Provider (MSP), который управляет тем, как новым членам могут выдаваться цифровые подписи и проверяться [5]. Организации могут быть разных масштабов. Однако эти предприятия должны быть организациями, не являющимися заказчиками в сети блокчейна. Организация транзакций в сети Hyperledger fabric представлена на рис. 3.

Узлы в сети управляются поставщиком услуг (MSP) [7], который действует как менеджер идентификации, предоставляя действительные цифровые подписи. Узлы в сети блокчейн можно разделить на три группы:

– *Клиенты* – веб-приложения, мобильные приложения, наборы для разработки программного обеспечения, которые взаимодействуют с сетью, отправляя запросы на выполнение транзакций и запрашивая порядок транзакций. Однако цепной код работает за пределами блокчейн-цепочки, взаимодействуя с узлами, а также с децентрализованным, распределенным, глобальным реестром.

– *Одноранговые узлы* – они ведут реестр блокчейна и выполняют цепной код. Существует два типа одноранговых узлов привязки и одноранговых узлов поддержки. Одноранговые узлы поддержки обрабатывают утверждение транзакций в соответствии с запросами. Узлы привязки определяются при создании сети, и они обеспечивают связь между различными организациями в сети, обмениваясь данными между соответствующими предприятиями.

– *Заказчик* – создает логический порядок транзакций, упаковывая их в блоки, а затем передавая в общую сеть. Заказчики отличаются от одноранговых узлов и существуют как совокупность узлов, которые заказывают транзакции по принципу FIFO «первым пришел – первым вышел».

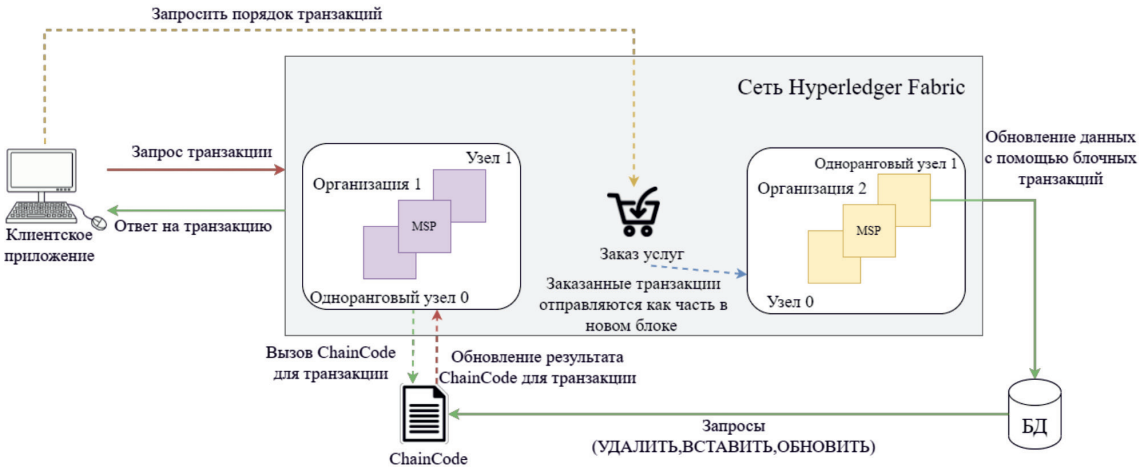

Рис. 3. Организация транзакций в сети Hyperledger fabric

Узлы в сети могут взаимодействовать друг с другом с помощью каналов, это частный и безопасный способ, который гарантирует, что узлы могут отправлять сообщения и блоки без вмешательства извне. В одной сети может быть несколько каналов между различными организациями и одноранговыми узлами для повышения конфиденциальности, добавление к каналу осуществляется с помощью MSP.

Блокчейн при применении к устройствам IoT может обеспечить повышение конфиденциальности и целостности по сравнению с нынешними подходами. Hyperledger Fabric, являющаяся разрешенным блокчейном, по своей сути имеет конфиденциальность и целостность, включающие дизайн [8]. Данная технология может быть интегрирована с различными устройствами интернета вещей.

Заключение

Таким образом, Hyperledger Fabric – это частный блокчейн, который предлагает решения для безопасного управления доступом и ведения реестра для объектов IoT. Структура Hyperledger по своей конструкции использует протоколы шифрования, политики доступа и криптографических функций, которые защищают от вредоносных атак. С помощью этого средства могут быть реализованы механизмы контроля доступа, а также может быть обеспечена конфиденциальность пользователей, поскольку центр сертификации использует открытый и закрытый ключи, защищающие личность пользователя.

Таким образом, можно сделать вывод о том, что представленные технологии достаточно эффективны для решения вопросов безопасности в IoT.

Список литературы

1. Ганкин Н.М., Михейлис Д.А. Hyperledger – инструментарий разработки отраслевых блокчейнов // Juvenis scientia. 2018. № 4. С. 17–19.

2. Вишняков В.А. Организация интернет-маркетинга с использованием интеллектуальных и блокчейн-технологий // Системный анализ и прикладная информатика. 2020. № 1. С. 18–23.

3. Ray P.P. A survey on Internet of Things architectures. Journal of King Saud University – Computer and Information Sciences. 2018. Vol. 30. No. 3. P. 291–319.

4. Ali M.S., Vecchio M., Pincheira M., Dolui K., Antonelli F., and M.H. Rehmani, Applications of Blockchains in the Internet of Things: A Comprehensive Survey. IEEE Communications Surveys & Tutorials. 2019. Vol. 21. No. 2. P. 1676–1717.

5. Lade P., Ghosh R., Srinivasan S. Manufacturing Analytics and Industrial Internet of Things. IEEE Intelligent Systems. 2017. Vol. 32. No. 3. P. 74–79.

6. Kang J., Yu R., Huang X., Wu M., Maharjan S., Xie S., Zhang Y. Blockchain for secure and efficient data sharing in vehicular edge computing and networks. IEEE Internet of Things Journal. 2019. Vol. 6. No. 3. P. 4660–4670.

7. Dai Y., Xu D., Maharjan S., Qiao G., and Zhang Y. Artificial Intelligence Empowered Edge Computing and Caching for Internet of Vehicles. IEEE Wireless Communications Magazine. 2019. Vol. 26. No. 3. P. 12–18.

8. Dorri A., Kanhere S.S., Jurdak R. MOF-BC: A memory optimized and flexible blockchain for large scale networks. Future Generation Computer Systems. 2019. Vol. 92. P. 357–373.

УДК 004.932

ПРОЕКТИРОВАНИЕ СИСТЕМЫ ОБНАРУЖЕНИЯ И РАСПОЗНАВАНИЯ ДОРОЖНЫХ ЗНАКОВ

Чулин К.В., Белов Ю.С.

ФГБОУ ВО «Московский государственный технический университет имени Н.Э. Баумана», филиал, Калуга, e-mail: fn1-kf@mail.ru

Распознавание дорожных знаков – одна из важных областей в интеллектуальных транспортных системах. Это связано с важностью дорожных знаков и светофоров в повседневной жизни. Распознавание дорожных знаков – это метод, который использует компьютерное зрение и искусственный интеллект для извлечения дорожных знаков из изображений на открытом воздухе в условиях неконтролируемого освещения, когда эти знаки могут быть закрыты другими объектами и могут иметь различные проблемы, такие как выцветание цветов, дезориентация и вариации по форме и размеру. Эту задачу можно разделить на два этапа: обнаружение и распознавание. В этой статье мы продемонстрировали эффективную систему обнаружения и распознавания дорожных знаков с новыми надежными функциями извлечения и представления. Система также может обеспечить неизменность освещения, масштаба и позы. Обладая чрезмерно полным набором функций, наша система способна полностью выделять отличительные черты для различных дорожных знаков и, следовательно, может быть легко расширена для размещения новых знаков. При использовании с отслеживанием можно использовать корреляцию между последовательными кадрами и добиться обнаружения дорожных знаков в реальном времени.

Ключевые слова: обнаружение дорожных знаков, распознавание дорожных знаков, изображения, классификация

DESIGN OF A ROAD SIGN DETECTION AND RECOGNITION SYSTEM

Chulin K.V., Belov Yu.S.

Bauman Moscow State Technical University, Kaluga branch, Kaluga, e-mail: fn1-kf@mail.ru

Traffic sign recognition is one of the important areas in intelligent transport systems. This is due to the importance of road signs and traffic lights in everyday life. Traffic sign recognition is a technique that uses computer vision and artificial intelligence to extract traffic signs from outdoor images under uncontrolled lighting conditions where these signs may be obscured by other objects and can have various problems such as color fading, disorientation and variation in shape and size. This task can be divided into two stages: detection and recognition. In this article, we have demonstrated an efficient traffic sign detection and recognition system with new robust extraction and presentation functions. The system can also provide consistent lighting, scale and pose. With an overly complete set of functions, our system is capable of completely distinguishing features for various road signs and therefore can be easily expanded to accommodate new signs. When used with tracking, you can use the correlation between successive frames and achieve real-time traffic sign detection.

Keywords: traffic sign detection, traffic sign recognition, images, classification

Интеллектуальные транспортные системы (ИТС) обладают огромным потенциалом для экономии времени, денег, спасения жизней и улучшения окружающей среды. Распознавание дорожных знаков – одна из важных областей в ИТС. Это связано с важностью дорожных знаков и светофоров в повседневной жизни. Они определяют визуальный язык, который могут интерпретировать водители. Они отображают текущую дорожную ситуацию, показывают опасности и трудности вокруг водителей, предупреждают их и помогают им в навигации, предоставляя полезную информацию, которая делает вождение безопасным и удобным. Идентификация дорожных знаков осуществляется в два основных этапа: обнаружение и распознавание. На этапе обнаружения изображение предварительно обрабатывается, улучшается и сегментируется в соответствии со свойствами знака, такими как цвет или форма. Результатом является сегментированное изображение, содержащее потенциальные области, которые могут быть распознаны как возможные дорожные знаки. Эффективность и скорость обнаружения являются важными факторами, которые играют важную роль во всем процессе, поскольку они сокращают пространство поиска и указывают только на потенциальные области. На этапе распознавания каждый из кандидатов проверяется на соответствие определенному набору признаков (шаблону), чтобы решить, входит ли он в группу дорожных знаков или нет, а затем в соответствии с этими характеристиками они классифицируются в разные группы [1].

Цель исследования: разработать систему обнаружения и распознавания дорожных знаков с надежными функциями извлечения и представления и обеспечивающую неизменность освещения, масштаба и положения.

Описание модели. Для решения задачи распознавания становится целесообразным

комбинировать несколько алгоритмов. Работа системы обнаружения и распознавания дорожных знаков происходит в два этапа: обнаружение дорожных знаков; распознавание дорожных знаков. Общая модель системы распознавания дорожных знаков состоит из ряда взаимосвязанных модулей.

Цветовая кодировка. Цвет предупреждающих дорожных знаков является ценным ключом к распознаванию знаков. Однако, поскольку дорожные знаки размещены в различных условиях освещения, трудно добиться обнаружения инварианта освещенности, используя непосредственно абсолютные значения цвета. Поэтому мы хотели бы кодировать информацию о цвете косвенно и использовать относительный вид вместо абсолютных значений на разных цветовых слоях [2].

Возьмем, к примеру, знак остановки. На слое R в пространстве RGB, поскольку и белый, и красный цвет пикселей имеют высокие значения интенсивности, величина градиента на этом слое мала. Однако на слоях G и B красные пиксели имеют низкое значение, в то время как значение интенсивности белых пикселей все еще высокое; таким образом, величина градиента на этих краях между белыми и красными пикселями высока. В результате градиент одного и того же пикселя на разных слоях может иметь очень разные значения, и эти значения коррелированы. Поэтому мы объединяем значения градиента на трех слоях в вектор и нормализуем его, используя сумму значений градиента по трем слоям.

Функция на основе HOG. Как и SIFT, мы адаптировали HOG для извлечения функций. Есть два преимущества использования методов на основе HOG. Во-первых, дорожные знаки имеют отличительную форму, а узоры на разных слоях могут иметь четкие края. Следовательно, градиент может эффективно улавливать эти особенности. Во-вторых, использование HOG может помочь добиться масштабной инвариантности.

Во-первых, изображение разделяется на квадратные области 3 на 3 и определяются 14 подблоков, показанных как заштрихованные области на рис. 1. Во-вторых, проектируем 15 шаблонов и делим диапазон ориентации [0 2π] на 8 ячеек, как показано на рис. 2. Каждый шаблон представляет собой квадратный блок с заштрихованной ячейкой [3].

Учитывая цветное изображение, функция оценивается на разных слоях в i-м подблоке с использованием j-го шаблона и k-й ориентации. Каждая функция обозначается $F(i,j,k)=[f_r(i,j,k),f_g(i,j,k),f_b(i,j,k)]^T$, где три компоненты соответствуют трем слоям RGB. Обозначим i-й подблок B_i и заштрихованную область ячейки в j-м шаблоне C_j. Затем для *fr* (*i*, *j*, *k*) мы сначала вычисляем градиент в каждом пикселе (x, y) на слое R как

$$G_{rx}(x,y)=[-101]*I_r(x,y),$$

$$G_{ry}(x,y)=[-101]^T*I_r(x,y);$$

сила градиента в точке (x, y) равна

$$G_r(x,y)=\sqrt{G_{rx}(x,y)^2+G_{ry}(x,y)^2},$$

и ориентация:

$$\theta_r(x,y)=tan^{-1}\left(\frac{G_{ry}(x,y)}{G_{rx}(x,y)}\right).$$

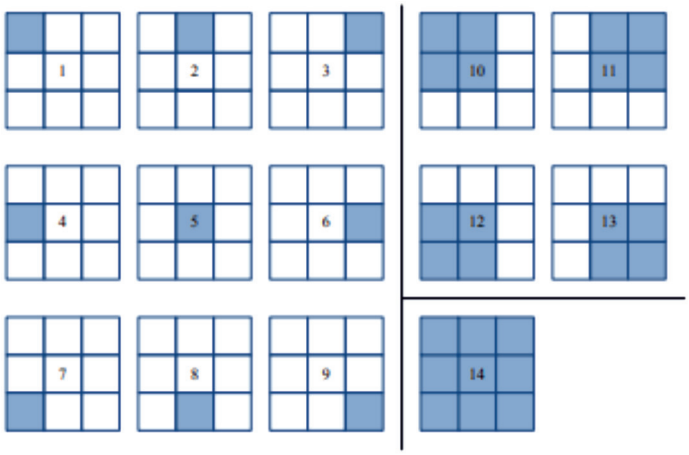

Рис. 1. Подблоки изображения

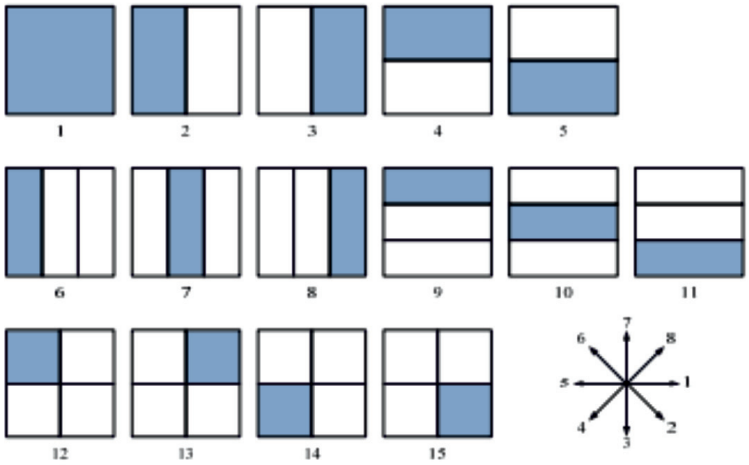

Рис. 2. Шаблоны и подборки ориентации

Помните, что мы разделили диапазон ориентации [0 2π] на 8 интервалов и обозначили значение k-го интервала как

$$\psi_{rk}(x,y)\begin{cases} G_r(x,y), if\ \theta_r(x,y) \in bin_k \\ 0, \text{ иначе} \end{cases}.$$

Значение признака для f_r (i, j, k) определяется как

$$f_r(i,j,k) = \frac{\sum_{(x,y)\in C_j} \psi_{rk}(x,y)}{\sum_{(x,y)\in B_i} (G_r(x,y) + G_g(x,y) + G_b(x,y))},$$

где $G_g(x, y)$ и $G_b(x, y)$ – величины градиента на слоях G и B соответственно.

Аналогично имеем

$$f_g(i,j,k) = \frac{\sum_{(x,y)\in C_j} \psi_{gk}(x,y)}{\sum_{(x,y)\in B_i} (G_r(x,y) + G_g(x,y) + G_b(x,y))},$$

$$f_b(i,j,k) = \frac{\sum_{(x,y)\in C_j} \psi_{bk}(x,y)}{\sum_{(x,y)\in B_i} (G_r(x,y) + G_g(x,y) + G_b(x,y))}.$$

Наконец, у нас есть

$$F(i,j,k) = [f_r(i,j,k), f_g(i,j,k), f_b(i,j,k)]^T.$$

Поскольку существует 14 подблоков, 15 шаблонов и 8 ориентаций, у нас всего 14×15×8 = 1680 функций.

Интегральные изображения широко используются в наших алгоритмах и, таким образом, сокращают повторные вычисления. Предварительное вычисление обеспечивает быстрое вычисление при предварительной обработке, а также включает этапы извлечения. Следующая формула показывает вычисление целостного изображения. Любое показание в интегральном изображении представляет собой сумму от (1,1) до этой точки.

Таким образом, суммирование области С может быть выполнено с использованием только четырех арифметических вычислений показаний в четырех углах этой области.

$$I_{Gk}(x,y) = \sum_{0\le x'\le x, 0\le y'\le y} \psi_k(x',y')$$

$$\sum_{(x,y)\in C} \psi_k(x,y) = I_{Gk}(x_c-1, y_c-1) + I_{Gk}(x_c+w_c-1, y_c+h_c-1)$$

$$-I_{Gk}(x_c-1, y_c+h_c-1) - I_{Gk}(x_c+w_c-1, y_c-1).$$

Рис. 3. Блок-схема фазы обучения

Методы обучения. Набор из 1680 функций является избыточным. Следовательно, нам необходимо выбрать признаки для каждого из признаков, которые могут лучше всего описать и различить их. Более того, процесс выбора признаков переплетается с процессом построения классификатора. Обзор этапа обучения показан на рис. 3. Наша цель – построить каскадный классификатор для каждого из признаков [4].

Во-первых, для каждого знака мы создали обучающий набор, в котором изображения текущего обучающего знака являются положительными образцами, а все другие знаки и случайно выбранные изображения являются отрицательными образцами. После этого вычисляются 1680 характеристик для каждого из образцов.

Во-вторых, для каждой функции мы используем машину опорных векторов с ядром RPF, чтобы классифицировать все образцы. Поскольку количество отрицательных образцов примерно в четыре раза больше, чем количество положительных образцов, мы скорректировали параметры штрафа члена ошибки для двух классов, чтобы справиться с этой несбалансированной ситуацией. Кроме того, мы использовали разные наборы параметров штрафа с разным акцентом на положительные и отрицательные образцы, чтобы найти компромисс между частотой обнаружения и частотой ложных срабатываний.

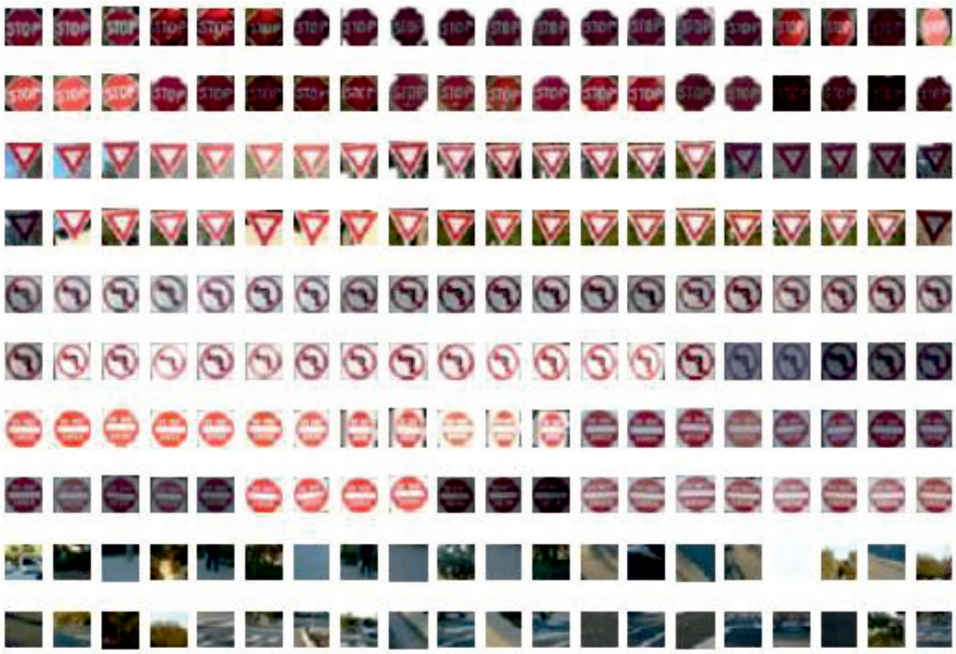

Рис. 4. Пример изображений в наборе данных

В-третьих, после получения результатов классификации с использованием каждой функции мы выбрали несколько основных функций с наивысшим уровнем обнаружения и относительно меньшим количеством ложных срабатываний. Мы перечислили различные варианты выбора и комбинации этих функций, и для каждой комбинации мы сложили выбранные функции как функцию более высокого измерения, а затем использовали новую отдельную функцию для классификации обучающего набора. Новая функция с наивысшим уровнем обнаружения и относительно меньшей частотой ложных срабатываний выбрана в качестве единственного классификатора первого уровня [5].

В-четвертых, получив единый классификатор первого этапа, мы использовали другой набор для проверки, чтобы проверить текущий классификатор, чтобы убедиться, что он имеет очень высокий уровень обнаружения (>98,5 %), и записали ложноположительные образцы. Затем мы объединили ложноположительные образцы как из обучающего набора, так и из набора проверки, как новый набор отрицательных образцов. Этот новый набор отрицательных выборок и положительные образцы из обучающего набора объединяются в новый обучающий набор.

Наконец, используя новый обучающий набор, мы повторили шаги со второго по четвертый, чтобы сформировать еще один новый одноэтапный классификатор, который каскадируется после текущего классификатора. Эта процедура повторяется до тех пор, пока каскадный классификатор не достигнет предопределенной частоты ложных срабатываний, имея при этом степень обнаружения выше, чем предопределенная наименьшая частота обнаружения [6].

Каскадные детекторы знаков. Каскадные классификаторы для разных знаков последовательно комбинируются для формирования окончательного классификатора предупреждающих дорожных знаков. Как показано на рис. 5, входное изображение сначала проверяется классификатором знака остановки. Если изображение проходит все этапы каскадного классификатора знаков остановки, мы помечаем его как возможную область знака остановки. В противном случае мы передаем его в классификатор знака «Въезд запрещен», и та же процедура выполняется для классификаторов без поворота налево и выхода. Если мы обнаруживаем, что текущее изображение не принадлежит ни к одному из этих знаков, мы объявляем его областью, которая не является знаком [7].

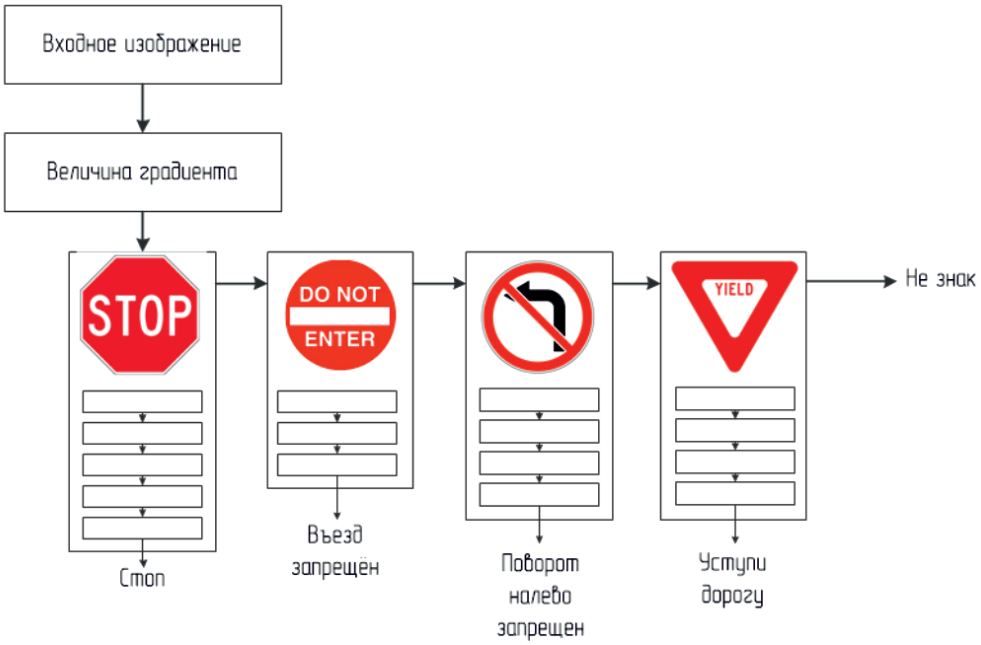

Рис. 5. Каскадные детекторы знаков

Заключение

В последнее десятилетие большой объем работы был проделан в области обнаружения и распознавания дорожных знаков. В этом исследовании принимали участие многие компании и исследовательские группы, и были достигнуты очень хорошие результаты. Распознавание дорожных знаков – это метод, который использует компьютерное зрение и искусственный интеллект для извлечения дорожных знаков из изображений на открытом воздухе в условиях неконтролируемого освещения, когда эти знаки могут быть закрыты другими объектами и могут иметь различные проблемы, такие как выцветание цветов, дезориентация и вариации по форме и размеру.

В данной статье продемонстрирована система обнаружения и распознавания дорожных знаков с надежными функциями извлечения и представления. Система также может обеспечить неизменность освещения, масштаба и положения. Обладая чрезмерно полным набором функций, система способна полностью выделять отличительные черты для различных дорожных знаков и, следовательно, может быть легко расширена для размещения новых знаков [8].

Список литературы

1. Dhar P., Abedin Md.Z., Biswas T. Traffic sign detection – A new approach and recognition using convolution neural network. IEEE Region 10 Humanitarian Technology Conference. 2017. P. 416–419.

2. Смольянинов В.А., Белов Ю.С. Проектирование программного комплекса обнаружения и распознавания дорожных знаков в потоковом видео // Научное обозрение. Технические науки. 2021. № 4. С. 16–21.

3. Pei S., Tang F., Ji Y., Fan J., Ning Z. Localized traffic sign detection with multi-scale deconvolution networks. IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). 2018. P. 355–360.

4. Belghaouti O., Handouzi W., Tabaa M. Improved traffic sign recognition using deep ConvNet architecture. Procedia Computer Science. 2020. Vol. 177. P. 468–473.

5. Saadna Y., Behloul A. An overview of traffic sign detection and classification methods. International Journal of Multimedia Information Retrieval 6. 2017. P. 193–210.

6. Аккуратов В.В., Белов Ю.С. Выбор архитектуры приложения для задачи распознавания дорожных знаков // Электронный журнал: наука, техника и образование. 2018. № 3 (21). С. 26–34.

7. Tabernik D., Skocaj D. Deep learning for large-scale traffic-sign detection and recognition. IEEE Transactions on Intelligent Transportation Systems. 2020. Vol. 21. No. 4. P. 1427–1440.

8. Sheikh M.A.A., Kole A., Maity T. Traffic sign detection and classification using colour feature and neural network. Conference on Intelligent Control Power and Instrumentation (ICICPI). 2016. P. 307–311.

УДК 004.051

ОСОБЕННОСТИ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ СОБЫТИЙНО-ОРИЕНТИРОВАННЫХ ВЕБ-ПРИЛОЖЕНИЙ

Ганьжа А.Ю., Карелова Р.А.

Нижнетагильский технологический институт (филиал)
ФГАОУ ВО «Уральский федеральный университет», Нижний Тагил, e-mail: riya2003@mail.ru

В статье рассмотрены преимущества и проблемы построения микросервисной архитектуры веб-приложений. Микросервисная архитектура представляет собой способ разработки программного обеспечения, при котором единое приложение разбивается на набор небольших автономных компонентов (микросервисов). При таком подходе каждый компонент работает в собственном процессе и обслуживает конкретную бизнес-задачу, взаимодействуя с другими микросервисами посредством протоколов для асинхронной передачи и распространения данных. К преимуществам такой архитектуры приложений относятся разбиение продукта через сервисы, проектирование «под отказ», децентрализованное управление данными. За счет этих особенностей микросервисная архитектура положительно влияет на развертывание и масштабирование продукта в целом, а жесткие границы модуля позволяют для разработки сервисов использовать разные языки программирования, выход из строя одного микросервиса не влияет на работу других, каждый сервис имеет собственную базу данных. В статье также описаны возможности архитектурных паттернов Event Sourcing и Command Query Responsibility Segregation (CQRS) для решения проблемы обеспечения согласованности данных при использовании микросервисной архитектуры. Для наглядности описаний приводятся графические иллюстрации в виде схем и таблиц, примеры листингов.

Ключевые слова: микросервисы, микросервисная архитектура, архитектурные паттерны, веб-приложения

FEATURES OF THE MICROSERVICE ARCHITECTURE OF EVENT-ORIENTED WEB-APPLICATIONS

Ganzha A.Yu., Karelova R.A.

Nizhny Tagil Technological Institute (branch) of Ural State University,
Nizhny Tagil, e-mail: riya2003@mail.ru

The article discusses the advantages and problems of building a microservice architecture for web applications. Microservice architecture is a way of software development in which a single application is broken down into a set of small, self-contained components (microservices). With this approach, each component operates in its own process and serves a specific business task, interacting with other microservices through protocols for asynchronous data transfer and distribution. The benefits of such an application architecture include product partitioning across services, failover design, and decentralized data management. Due to these features, the microservice architecture has a positive effect on the deployment and scaling of the product as a whole, and the rigid boundaries of the module allow using different programming languages to develop services, the failure of one microservice does not affect the work of others; each service has its own database. This article also describes the capabilities of the Event Sourcing and Command Query Responsibility Segregation (CQRS) architectural patterns to address data consistency issues when using a microservice architecture. For clarity of descriptions, graphical illustrations are given in the form of diagrams and tables, examples of listings.

Keywords: microservices, microservice architecture, design patterns, web-applications

По мере увеличения функциональных возможностей веб-приложения появляется потребность в управлении его гибкостью и отказоустойчивостью. С этой задачей призвана справиться правильно спроектированная архитектура приложения.

Существует несколько подходов к архитектурному проектированию веб-приложений. Так, вариантом архитектуры может быть монолитное приложение, построенное как единое целое, где вся логика по обработке запросов помещается внутри одного процесса.

Актуальным же решением на настоящий момент является проектирование архитектуры веб-приложений с использованием микросервисов. Микросервисный подход подразумевает разбиение приложения на небольшие сервисы, каждый из которых задействуется в собственном процессе, взаимодействуя с остальными посредством протоколов передачи данных.

Цель работы заключалась в систематизации и иллюстрации особенностей веб-приложений, построенных на основе микросервисной архитектуры.

Материалы и методы исследования

В рамках работы был проанализирован и систематизирован опыт современных разработчиков программного обеспечения, отраженный в публикациях, в том числе тематических форумах.

Результаты исследования и их обсуждение

Традиционный подход к разработке веб-приложений, как правило, подразуме-

вает использование монолитной архитектуры приложения, когда оно разрабатывается в виде единого блока. При таком подходе зачастую используется трехуровневая клиент-серверная архитектура, при которой веб-приложение состоит из трех основных частей: клиентская часть, серверная часть и база данных. Серверная часть веб-приложения занимается обработкой HTTP-запросов, получает и изменяет данные в базе данных, а также заполняет HTML-страницы, которые затем отправляются на клиентскую часть. Именно серверная часть веб-приложения представляет собой монолитную структуру, и любые изменения в ней требуют создания и развертывания новой версии данного приложения [1, с. 42]. В данном случае вся логика запроса обрабатывается в единственном процессе. Это означает, что внедрение новой технологии повлечёт за собой необходимость модернизации всего приложения. При этом масштабировать монолитное веб-приложение придется целиком, даже если это необходимо только для одного модуля приложения (рис. 1) [2].

Рис. 1. Схема масштабирования веб-приложения с монолитной архитектурой

Указанные выше недостатки привели к появлению и стремительному развитию микросервисной архитектуры – подходу к разработке единого веб-приложения в виде набора небольших сервисов, каждый из которых запущен в собственном процессе, в то время как коммуникации с другими сервисами осуществляются с помощью протоколов передачи данных, например HTTP.

Микросервисная архитектура имеет ряд характерных преимуществ. Одним из таких преимуществ является *деление на сервисы*. Разработка веб-приложений на основе микросервисной архитектуры подразумевает использование таких компонентов, как библиотеки и сервисы. С точки зрения микросервисного подхода библиотеки представляют собой определенные компоненты, которые могут подключаться к программе, а также быть вызваны ей в одном процессе. В свою очередь, сервисами считаются компоненты, которые могут выполняться в отдельном процессе и связываться друг с другом, например, с помощью такого архитектурного стиля взаимодействия компонентов, как REST. При таком подходе отпадает необходимость в повторном развертывании всего веб-приложения после внесения изменений, поскольку приложение, разбитое на сервисы, потребует развертывания только сервиса, который был изменён.

Помимо независимого масштабирования продукта (рис. 2), каждый сервис также обеспечивает жесткие границы модулей, делая при этом возможной разработку сервисов на разных языках программирования [1, с. 43].

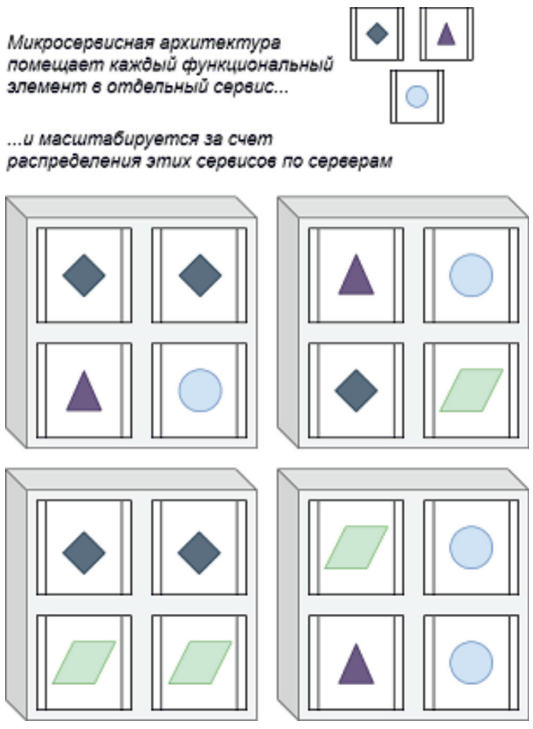

Рис. 2. Схема масштабирования веб-приложения с микросервисной архитектурой

Вторым характерным преимуществом микросервисной архитектуры является *проектирование под отказы*. Это означает, что сбой в работе отдельных микросервисов несущественно скажется на работе других: остальные сервисы продолжат работать. Обеспечение отказоустойчивости монолитного веб-приложения является более трудоемкой задачей, так как выход из строя единой базы данных влечет за собой неработоспособность всех модулей веб-приложения.

Кроме того, одним из важнейших преимуществ микросервисной архитектуры является *децентрализация управления данными*. Применение микросервисного подхода подразумевает наличие собственной базы данных у каждого сервиса (рис. 3), в то время как для монолитной архитектуры характерно использование одной базы данных для всего веб-приложения.

Использование одного хранилища разными сервисами возможно в случае, если службы идентичны друг другу, но их базы данных никак не взаимодействуют друг с другом [3, с. 123; 4, с. 116].

Затрагивая тему децентрализации управления данными, следует упомянуть о том, как клиенты микросервисного приложения, такие как веб-клиенты или мобильные клиенты, могут получать доступ к отдельным видам услуг. Выделяют разные модели взаимодействия с микросервисами, наиболее общепринятой из них считается API Gateway (рис. 4).

API Gateway можно рассматривать как дополнительный сервис веб-приложения, который формирует запросы к основным сервисам. Таким образом, API Gateway выступает в роли шлюза, который обрабатывает клиентские запросы и перенаправляет их к нужному сервису.

Однако помимо преимуществ микросервисная архитектура имеет и ряд характерных недостатков. В частности, выделяют проблему обеспечения согласованности данных микросервисов. Причина её возникновения кроется в устойчивости транзакций и запросов к разделению на сервисы. Это существенно усложняет процесс разработки транзакционных веб-приложений, построенных на основе микросервисной архитектуры. Рассмотрим сущность указанной проблемы на примере микросервисов интернет-магазина.

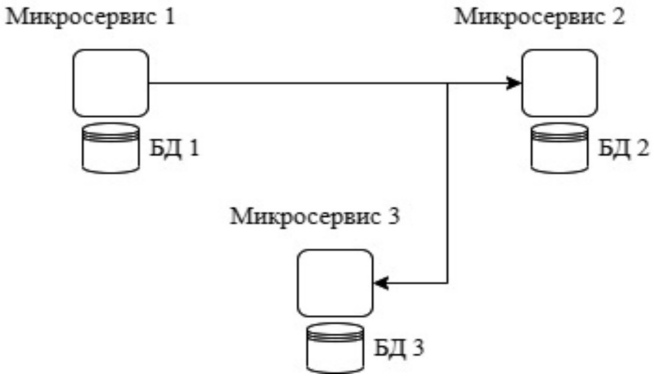

Рис. 3. Схема экземпляров баз данных микросервисов

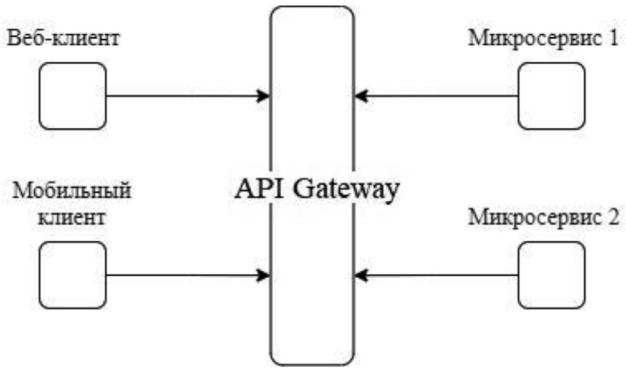

Рис. 4. Схема взаимодействия API Gateway

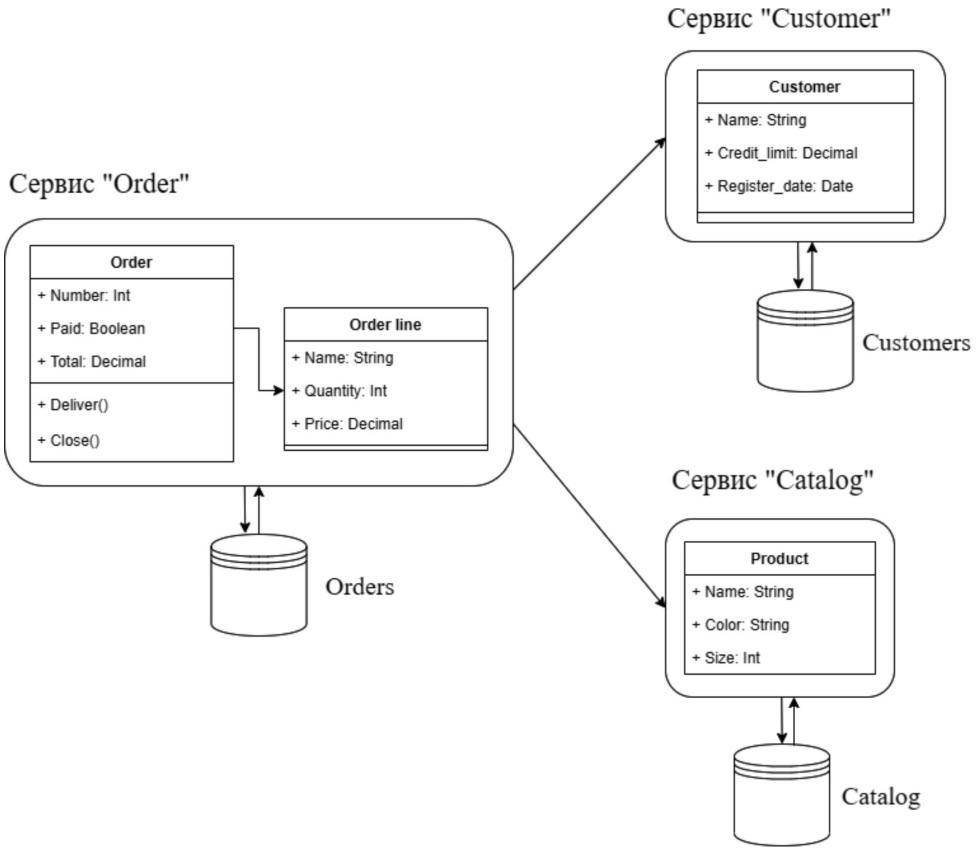

Рис. 5. Схема микросервисной архитектуры интернет-магазина

Архитектура интернет-магазина включает в себя такие микросервисы, как «Order», «Customer» и «Catalog». Причем каждый сервис обладает собственной базой данных и содержит в себе классы, которые являются его составными частями (рис. 5).

Создание заказа подразумевает проверку кредитного лимита клиента: веб-приложение не должно допустить его превышения при размещении нескольких заказов. При использовании классического монолитного подхода таблицы «Customers» и «Orders» располагались бы в одной базе данных, и в таком случае наиболее простым решением обеспечения согласованности данных являлось бы применение транзакций (листинг 1).

Листинг 1. Транзакция проверки кредитного лимита клиента

```
BEGIN TRANSACTION
…
SELECT credit_limit FROM customers WHERE customer_id = ?
…
SELECT order_total FROM orders WHERE customer_id = ?
…
INSERT INTO ORDERS …

COMMIT TRANSACTION
```

Однако применение такого подхода в сочетании с микросервисами не представляется возможным в силу изолированности таблиц «Customers» и «Orders», находящихся в разных базах данных и микросервисах.

По этой же причине возникают неудобства с реализацией запросов к базе данных. Применение оператора JOIN, предназначенного для выполнения операции соединения данных из двух или более таблиц, становится невозможным (листинг 2).

Листинг 2. JOIN-запрос на поиск недавно зарегистрированных клиентов

```
SELECT * FROM customers c
INNER JOIN orders o ON c.customer_id = o.customer_id
WHERE c.register_date > ?
```

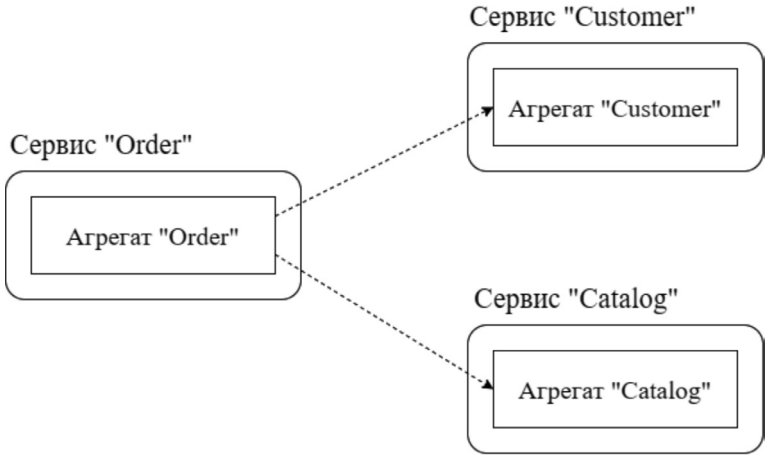

Рис. 6. Схема агрегатов интернет-магазина

Пример таблицы «Events»

Event_id	Event_type	Aggregate_id	Aggregate_type	Event_data
11	Order Created	10	Order	…
12	Order Confirmed	10	Order	…
13	Order Shipped	10	Order	…
14	Order Delivered	10	Order	…

Решением вышеперечисленных проблем микросервисной архитектуры может стать использование таких событийно-ориентированных подходов, как Event Sourcing и Command Query Responsibility Segregation (далее – CQRS). Под событийно-ориентированным подходом подразумевают парадигму архитектуры программного обеспечения, способствующую порождению, обнаружению, потреблению событий и реакции на них. При таком подходе ядром веб-приложения становятся события, создаваемые для запуска и обмена данными между сервисами. Сервисы, в свою очередь, отслеживают определенные события и реагируют на них. Сервис, создавший событие, не знает, какой именно сервис его отслеживает. Таким образом реализуется слабая связанность микросервисов, способствующая обеспечению согласованности данных веб-приложения и упрощению его масштабирования.

Событийно-ориентированный подход Event Sourcing подразумевает сохранение цепочки изменений, вносимых в состояние веб-приложения. Таким образом, сохраненная последовательность может быть использована как указатель актуального состояния приложения или, например, как список возникших событий [5]. При использовании указанного подхода можно добиться высокой степени децентрализации чтения и изменения данных.

В Event Sourcing указателем актуального состояния веб-приложения является агрегат. Агрегатом называют набор элементов предметной области, нераздельно связанных друг с другом (рис. 6). Чтобы получить его актуальное состояние, необходимо запустить загрузку событий и их исполнение. Изменение агрегата требует создания определенного события, которое включает в себя логику изменения какой-либо части агрегата. При этом сам агрегат и его состояние не будут сохранены [5].

В качестве примера рассмотрим применение механизма Event Sourcing для сервиса «Order». В данном случае сервис не будет хранить заказы в виде строки в таблице «Orders», вместо этого будет сохранена последовательность событий для каждого агрегата «Order». Например, такие события, как «Order created», «Order confirmed», «Order shipped» и «Order delivered», будут находиться в определенной таблице «Events» (таблица).

Столбцы Event_id, Event_type и Event_data содержат в себе идентификатор, тип и описание события, столбцы Aggregate_id и Aggregate_type, в свою очередь, являются идентификаторами агрегата.

Применение архитектурного паттерна Event Sourcing имеет целый ряд преимуществ, главное из которых состоит в том, что события гарантированно публикуются при изменении агрегата, что существенно помогает в построении микросервисной архитектуры, управляемой событиями.

Паттерн Event Sourcing способен хранить историю изменений каждого агрегата, что является ощутимым преимуществом при работе с временными запросами, способными возвращать прежнее состояние агрегата.

Однако все перечисленные преимущества архитектурного паттерна Event Sourcing не способны избавиться от проблемы осуществления запросов к базе данных. В её решении может помочь применение паттерна CQRS.

CQRS представляет собой архитектурный шаблон, разделяющий процедуры чтения и обновления хранилища данных. Использование данного паттерна подразумевает разбиение веб-приложения на командную и запросную составляющие. В таком случае командная составляющая будет работать с командами для создания, чтения, обновления и удаления агрегатов. Запросная же составляющая будет задействована в обработке запросов, например POST или GET-запросов [5].

новления данных о клиентах и их заказах. В свою очередь сервис «Viewing customers» будет располагаться в запросной составляющей веб-приложения, представляя собой API-интерфейс получения данных о клиентах интернет-магазина. Данный сервис будет наблюдать за событиями, которые исполняются командной составляющей веб-приложения, а также заниматься обновлением данных в хранилище интернет-магазина.

Таким образом, разделение веб-приложения на командную и запросную составляющие повышает производительность, гибкость и безопасность всей системы. Кроме того, с помощью применения паттерна CQRS совместно с событийно-ориентированным подходом решается проблема согласованности данных в микросервисах, позволяя веб-приложению эффективно осуществлять различные виды запросов.

Заключение

Динамично развивающийся подход к разработке веб-приложений на основе микросервисов позволяет создавать отказоустойчивые веб-приложения, которые довольно просто масштабировать и обслуживать. А такая особенность, как разбиение через сервисы, помогает увеличить скорость и эффективность разработки приложения за счет деления участников команды по конкретным бизнес-задачам. Однако следует принимать во внимание возможные трудности применения микросервисной архитектуры, главным образом связанные с поддержанием согласованности данных.

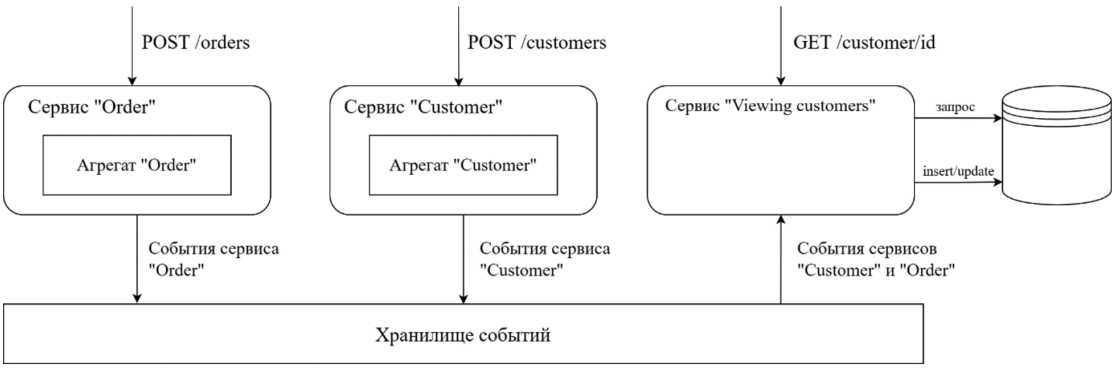

Рис. 7. Схема применения паттерна CQRS в архитектуре интернет-магазина

Рассмотрим пример применения паттерна CQRS для интернет-магазина (рис. 7). Сервисы «Customer» и «Order» вынесем в командную составляющую веб-приложения – теперь они будут представлять API-интерфейсы создания и об-

Эффективным способом решения этой проблемы можно считать совместное использование архитектурных паттернов Event Sourcing и CQRS, позволяющих реализовывать различные виды запросов в микросервисной архитектуре, повышая при этом

производительность и безопасность всего веб-приложения.

Список литературы

1. Осипов Д.Б. Проектирование программного обеспечения с помощью микросервисной архитектуры // Вестник науки и образования. 2018. № 5 (41). С. 41–46. [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/proektirovanie-programmnogo-obespecheniya-s-pomoschyu-mikroservisnoy-arhitektury (дата обращения: 06.12.2021).

2. Fowler M. Microservices [Электронный ресурс]. URL: https://martinfowler.com/articles/microservices.html (дата обращения: 06.12.2021).

3. Шитько А.М. Проектирование микросервисной архитектуры программного обеспечения // Труды БГТУ. Серия 3: Физико-математические науки и информатика. 2017. № 9. С. 122–125. [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/proektirovanie-mikroservisnoy-arhitektury-programmnogo-obespecheniya (дата обращения: 06.12.2021).

4. Щелбанин А.В. Особенности применения микросервисной архитектуры при разработке программного обеспечения // Фундаментальные и прикладные научные исследования: актуальные вопросы, достижения и инновации: сборник статей V Международной научно-практической конференции. В 4 ч. Ч. 4. Пенза: МЦНС «Наука и Просвещение», 2017. С. 112–117.

5. Макеев А. Использование паттернов Event Sourcing и CQRS для разработки приложения на Spring Boot и Axon Framework. [Электронный ресурс]. URL: https://tproger.ru/articles/ispolzovanie-patternov-event-sourcing-i-cqrs-dlja-razrabotki-prilozhenija-na-spring-boot-i-axon-framework/ (дата обращения: 06.12.2021).