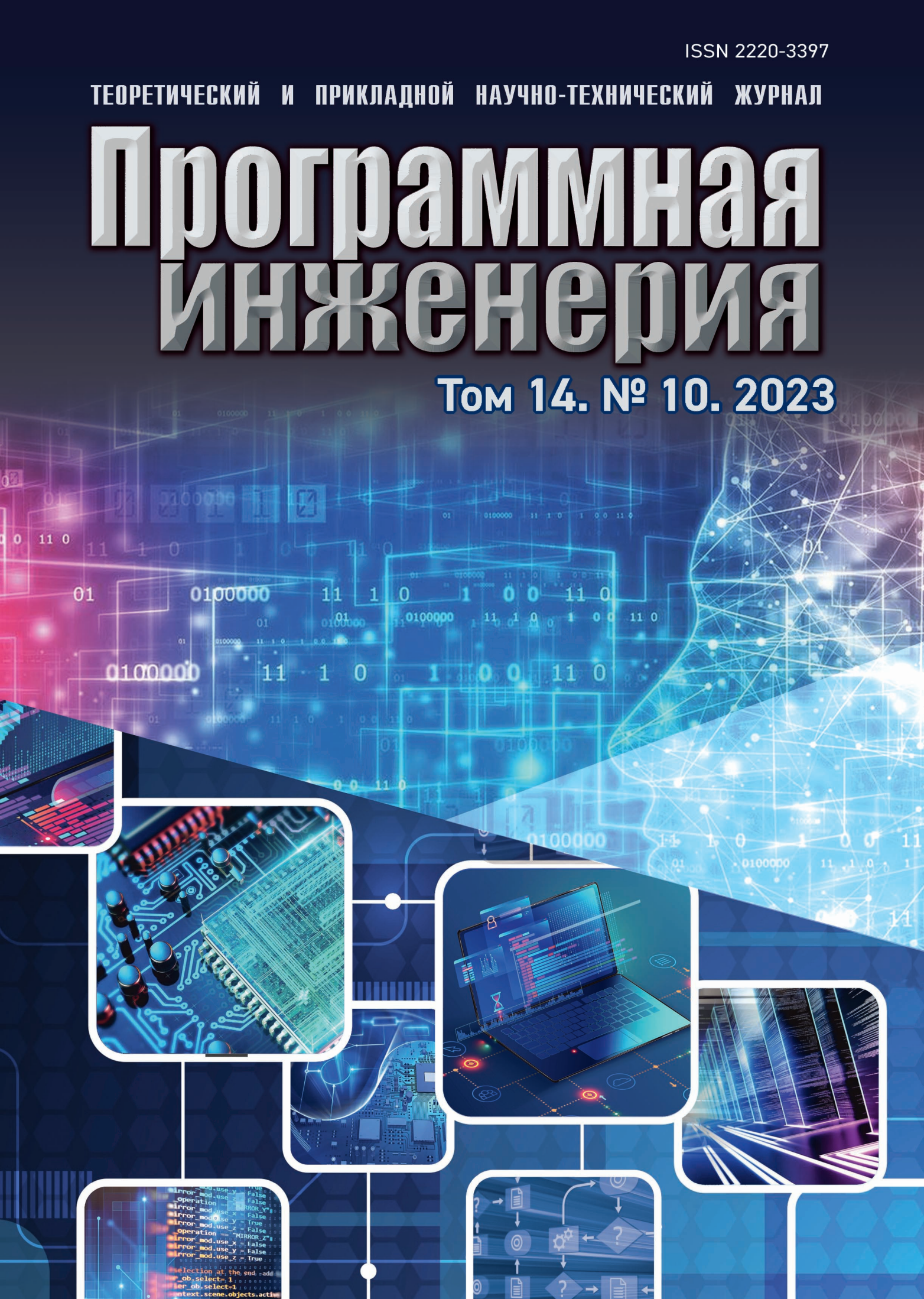
ISSN 2220-3397

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Программная инженерия

Том 14. № 10. 2023



Рисунки к статье **Е. А. Басыни, Н. Карапетьянца, М. Карапетьянца**
«ИССЛЕДОВАНИЕ СУЩЕСТВУЮЩИХ ПОДХОДОВ
К АНАЛИЗУ ТРАНЗАКЦИЙ В СЕТИ BITCOIN»

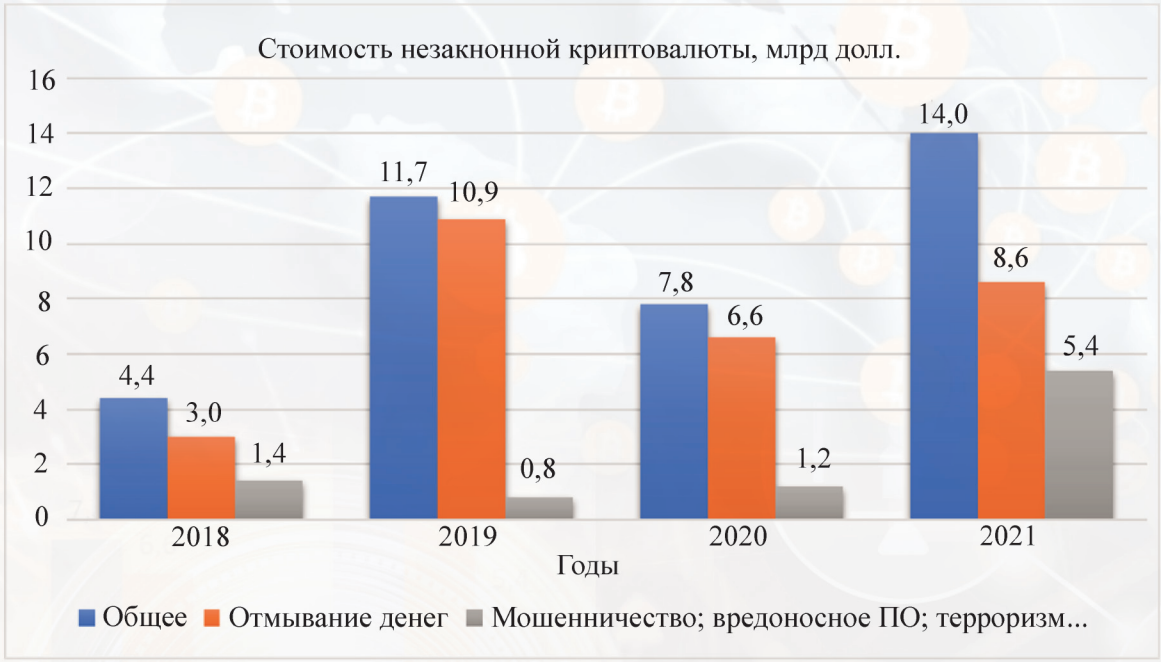


Рис. 1. Диаграмма изменения суммарной стоимости криптовалюты в незаконной деятельности

Рис. 2. Карта мира крипторегулирования в 2021 г.

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г. DOI 10.17587/issn.2220-3397 ISSN 2220-3397

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана

СОДЕРЖАНИЕ

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в подписных агентствах (индекс по Объединенному каталогу "Пресса России" — **22765**) или непосредственно в редакции (для юридических лиц).
Тел.: (499) 270-16-52.
Http://novtex.ru/prin/rus E-mail: prin@novtex.ru
Журнал включен в Российский индекс научного цитирования (РИНЦ) и Russian Science Citation Index (RSCI).
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

Vol. 14
N 10
2023

Published since September 2010 **DOI 10.17587/issn.2220-3397** **ISSN 2220-3397**

CONTENTS

Information about the jornal is available online at:
http://novtex.ru/prin/eng e-mail:prin@novtex.ru

УДК 004.42
DOI: 10.17587/prin.14.471-481

Е. А. Курако, канд. техн. наук, ст. науч. сотр., keaipu@yandex.ru,
Р. Э. Асратян, канд. техн. наук, вед. науч. сотр., rubezas@yandex.ru,
В. Л. Орлов, канд. техн. наук, вед. науч. сотр., ovl@ipu.ru,
Федеральное государственное бюджетное учреждение науки Институт проблем управления им. В. А. Трапезникова РАН, Москва

Импортозамещение информационных систем, базирующихся на языке C# и сетевой архитектуре

Поступила в редакцию 20.07.2023
Принята к публикации 11.08.2023

Рассмотрены методы преобразования информационных систем, имеющих сетевую архитектуру и созданных на основе языка C#, таким образом, чтобы они могли выполняться в среде, компоненты которой включены в реестр российского программного обеспечения. Основу рассматриваемой среды составляют операционные системы, средства управления базами данных, web-средства, а также фреймворки и программные библиотеки. Допускается дополнительное использование совокупности открытых компонентов, т. е. тех, которые содержат полный набор исходных текстов и которые можно подвергнуть процедуре верификации. Выделены основные направления подготовки исполнительной среды. Определены методы настройки web-сервера Apache 2 и создания виртуального хоста. Проведена краткая оценка способов миграции баз данных. Рассмотрены вопросы коррекции исходного кода C# при изменении операционной среды и методы преобразования исходного текста при изменении способов работы с графической средой.

Ключевые слова: *импортозамещение, информационная система, C#, Astra Linux, Моно, PostgreSQL, Apache 2, графический интерфейс, web-сервисы*

Для цитирования:

Курако Е. А., **Асратян Р. Э.**, **Орлов В. Л.** Импортозамещение информационных систем, базирующихся на языке C# и сетевой архитектуре // Программная инженерия. 2023. Том. 14, № 10. С. 471—481. DOI: 10.17587/prin.14.471-481.

Введение

Импортозамещение информационных систем в настоящее время становится актуальной задачей с точки зрения обеспечения технологического суверенитета страны. Ее решение обусловлено необходимостью повышения уровня надежности программных средств, недопущением инородных включений и разного рода закладок. Очевидно, что при разработке новых систем необходимо использовать проверенное базовое программное обеспечение (ПО) как в процессе эксплуатации, так и в процессе разработки.

Здесь очень важно иметь ввиду следующее. Если импортозамещение представляет собой замену определенного иностранного продукта на отечественный, например, переход на другую систему документооборота предприятия, то основная трудоемкость здесь приходится на настройку нового ПО, а иногда и замену используемых технологических процессов. Важно подобрать такую систему, которая во многих аспектах была бы аналогична использующейся ранее. В работах [1, 2] приведено достаточно много примеров замены продуктов с кратким описанием их возможностей.

Другой вариант импортозамещения представляет собой использование открытых продуктов в исходной системе с появившейся в настоящее время необходимостью замены одного или двух компонентов, которые не отвечают новым критериям. Например, система — это web-портал, работающий изначально под операционной системой

(ОС) Linux, а требуется отказ от использования коммерческой системы управления базами данных (СУБД) и переход на открытую СУБД. Результаты успешных работ по методам замены периодически появляются, показывая, что это не простой процесс [3].

Существует более сложный случай, когда требуется замена многих компонентов системы. Очень часто при этом предлагается сменить архитектуру системы и язык программирования [4, 5], что равносильно написанию новой информационной системы. Хотелось бы отметить, что, хотя процесс перехода на отечественное ПО охватывает все больше участников [6], эти участники являются, по большей части, госкорпорациями и ведомствами. А они, к сожалению, практически не публикуют результаты своих работ по импортозамещению.

Ниже рассматривается конкретный, но распространенный вариант, когда требуется замена ОС и СУБД. Вместе с тем предполагается сохранение неизменным языка программирования, что позволяет сократить уровень возможных изменений.

Кратко определим для рассмотрения информационную систему, базирующуюся на сетевой архитектуре и включающую серверную и клиентскую части.

Минимальный набор эксплуатационных программных средств для серверной части включает:

- операционную систему;
- web-сервер (при использовании web-технологий);
- СУБД.

Минимальный набор эксплуатационных программных средств для клиентской части включает:

- ОС;
- браузер (если он применяется для реализации функций клиента);
- офисный пакет (если используются его средства).

Минимальный набор средств разработки представляет собой:

- трансляторы и (или) интерпретаторы языков программирования;
- фреймворки и (или) наборы библиотек для построения системы.

1. Импортозамещение эксплуатационных программных средств и средств разработки при использовании языка C#

В настоящей работе рассматриваются программные системы, базирующиеся на языке C#. Это означает, что исходные информационные системы вероятнее всего использовали систему Windows как для серверных, так и для клиентских частей. В качестве web-сервера применялся Internet Information Services (IIS). Для организации баз данных в составе информационных систем наиболее часто использовались СУБД Oracle и Microsoft SQL Server. В качестве офисного пакета находил применение Microsoft Office.

Средства разработки в основном были представлены пакетом Microsoft Visual Studio, который обеспечивал построение и трансляцию проектов с использованием .NET Framework.

Если мы говорим об импортозамещении, то, прежде всего, имеем в виду наличие проверенных на практике инструментальных программ, обеспечивающих как разработку, так и процесс эксплуатации. При этом данные программы должны входить в реестр российского программного обеспечения [7].

Разумеется, определение ОС и СУБД для миграции программ является первоочередной задачей. В настоящее время существует много ОС, присутствующих в реестре российского программного обеспечения (больше 30), что затрудняет выбор. В связи с этим обстоятельством Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации определило три самые популярные российские системы на основе Linux, которые рекомендуется использовать в качестве базовых. К их числу относятся Astra Linux, Альт и Ред ОС [8].

При переходе на новую ОС целесообразно сохранять язык программирования. Для работы с языком C# целесообразно использовать исполнительную среду — пакет Моно [9], зарегистрированный в реестре отечественного ПО (№ 15242) фирмой «Лаборатория 50». Этот пакет может работать под управлением как ОС Astra Linux Special Edition 1.3-1.7, так и ОС Альт. Так как Astra Linux с максимальным уровнем защиты может использоваться для обработки информации любой категории, то в дальнейшем рассматриваем в настоящей работе использование этой ОС для информационных систем, построенных на языке C#.

Для использования механизмов web-сервисов необходимо провести активизацию web-сервера Apache 2, входящего в состав Astra Linux.

При рассмотрении вариантов СУБД приходится учитывать, что средства Oracle (а она наиболее часто использовалась в информационных системах) являются собственностью зарубежных организаций и не входят в реестр российского программного обеспечения.

В качестве СУБД для импортозамещения может использоваться широко распространенная си-

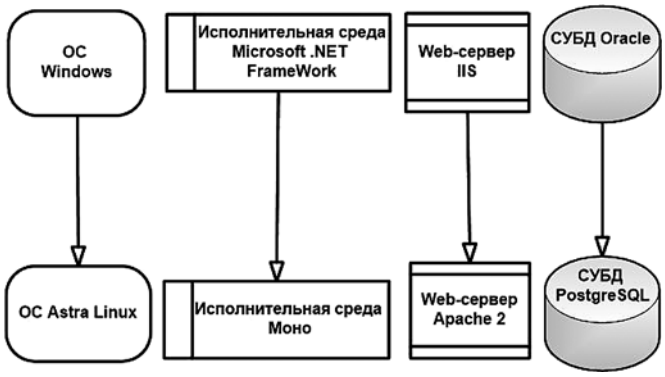


Рис. 1. Основные процедуры импортозамещения программного комплекса, разработанного на языке C#

стема PosgreSQL, поставляемая совместно с Astra Linux, а также Postgres Pro. Следует отметить, что в 2023 г. появилась новая платформа Astra DB Tantor Platform, которая базируется на семействе PostgreSQL.

Таким образом, проведение импортозамещения информационной системы, основанной на языке C#, является нетривиальной задачей комплексного типа. Для решения этой задачи необходимо заменить ОС Windows на Astra Linux. Вместо web-сервера IIS, стандартно применяющегося в среде Windows, нужно использовать web-сервер Apache 2 из состава Astra Linux. Этот web-сервер должен быть настроен таким образом, чтобы он мог работать с web-сервисами, написанными на языке C#.

Для работы с отечественным ПО необходимо провести миграцию баз данных [10] из среды Oracle в среду PostgreSQL. Наиболее сложной процедурой является перевод исходных текстов программ, включая сервисы под управлением пакета Моно. Схематично указанные преобразования показаны на рис. 1.

2. Замена операционной системы и подготовка исполнительной среды Моно

В процессе импортозамещения необходима замена ОС на другую, входящую в реестр российского программного обеспечения. В то же время одной из наиболее сильных сторон Microsoft являются развитые средства поддержки сетевой архитектуры .NET и, в частности, технологии web-сервисов как на уровне клиента, так и на уровне сервера в среде Windows. Учитывая большое значение этой архитектуры и этой технологии в разработках распределенных информационных систем, можно заключить, что поиск методов и средств организации аналогичной поддержки в среде Linux является важной частью задачи импортозамещения.

Все рабочие станции и серверы, на которых должны функционировать клиентские и сервисные компоненты, разработанные на языке C#, должны быть оснащены исполнительной средой Моно. Рассмотрим установку этой среды с помощью команды apt (Advanced Packaging Tool) — основного средства работы с программными пакетами в версиях ОС Linux типа Astra Linux, Debian, Ubuntu и т. п.

Для установки исполнительной среды Моно понадобится доступ к репозиторию компании «Лаборатория 50» [9]. Для этого компьютер, на котором выполняется установка, должен быть подключен к сети Интернет. Установка выполняется администратором системы из программы Терминал. Она включает следующие шаги.

1. Получение открытого ключа репозиториев «Лаборатории 50» командами

```
sudo wget –qO /etc/apt/trusted.gpg.d/lab50.gpg \
     http://packages.lab50.net/lab50.gpg
sudo chmod 644 /etc/apt/trusted.gpg.d/lab50.gpg
```

Полученный ключ будет автоматически использоваться для проверки подлинности всех скаченных пакетов.

2. Добавление сведений о репозиториях «Лаборатории 50». С помощью любого текстового редактора добавление в файл /etc/apt/sources.list строк

```
deb http://packages.lab50.net/mono/ alse17 main
deb http://packages.lab50.net/alse/ alse17-security
main \
     contrib non-free
```

3. Установка основных пакетов Моно с помощью команд

```
sudo apt update
sudo apt install mono-devel
sudo apt install mono-complete
```

3. Установка и настройка web-сервера Apache 2, создание виртуального хоста

Для создания web-сервисов с использованием языка C# в среде Linux необходимо не только установить web-сервер Apache 2, но и соответствующим образом настроить его.

В отличие от IIS использование Apache 2 предполагает несколько предварительных шагов, связанных с установкой в системе дополнительных пакетов поддержки языка C#, активацией модулей

```
                        # Установка Apache 2
sudo apt update
sudo apt install apache2

# Установка дополнительных пакетов поддержки C# и исполнительной среды
sudo apt install mono-xsp4-base
sudo apt install mono-xsp4
sudo apt install mono-apache-server4
sudo apt install libapache2-mod-mono
sudo apt install libentityframework6-npgsql-cil


# Активация модуля поддержки Mono в Apache 2
sudo a2enmod mod_mono_auto
sudo systemctl restart apache2
```

Рис. 2. Команды установки Apache 2 и дополнительных пакетов

поддержки Моно в web-сервере и созданием виртуальных хостов (виртуальных web-серверов) для последующего размещения web-сервисов в Apache 2. Последовательность установки пакетов и активации модуля поддержки Моно приведена на рис. 2.

Создание виртуального хоста для размещения web-сервера выполняется по обычным правилам Apache 2 — путем подготовки конфигурационного файла нового хоста в директории /etc/apache2/sites-available и корневой директории нового хоста в директории /var/www. Однако в данном случае и содержание конфигурационного файла, и содержание корневой директории будут иметь несколько специфических черт, связанных с поддержкой Моно и web-сервисов.

Предположим, что возникла потребность создать виртуальный хост с Интернет-именем ipu.demohost.ru в web-сервере Apache 2 и, соответственно, создана корневая директория с тем же именем (/var/www/ipu.demohost.ru) для размещения web-сервиса. На рис. 3. приведен пример текста конфигурационного файла ipu.demohost.ru.conf для нового виртуального хоста. Не будем подробно разбирать его структуру (она описана в документации на web-сервер Apache 2), подчеркнем лишь, что имя хоста и его корневой директории

```
<Virtualhost *:80>
   ServerName ipu.demohost.ru
  ServerAdmin admin@ipu.demohost.ru
  DocumentRoot /var/www/ipu.demohost.ru
  MonoServerPath ipu.demohost.ru "usr/bin/mod-mono-server4"
  MonoSetEnv ipu.demohost.ru MONO_IOMAP=all
  MonoApplications ipu.demohost.ru "/:/var/www/ipu.demohost.ru"
  <Directory "/">
    Allow from all
    Order allow,deny
    MonoSetServerAlias ipu.demohost.ru
    SetHandler mono
    SetOutputFilter DEFLATE
    SetEnvIfNoCase  Request_URI  "\.(?:gif|jpe?g|png)$"  no-gzip  dont-
vary
  </Directory>
  <IfModule mod_deflate.c>
     AddOutputFilterByType DEFLATE text/html text/plane text/xml
  </IfModule>
</VirtualHost>
```

Рис. 3. Пример конфигурационного файла виртуального хоста для web-сервера

упоминаются в нем несколько раз (выделены серым фоном).

Подготовка виртуального хоста завершается его активизацией с помощью команд

```
sudo a2ensite ipu.demohost.ru
sudo systemctl restart apache2
```

4. Установка СУБД PostgreSQL

Система управления базами данных PostgresSQL, как правило, разворачивается одновременно с ОС Astra Linux. Для этого необходимо установить галочку в меню установки ОС. Если ОС установлена, а СУБД нет, то ее можно доустановить командой

```
$ sudo apt-get install postgresql postgresql-contrib
```

После этого установится само ПО СУБД и пакет библиотек с дополнительными возможностями.

Настройка и администрирование установленной базы подробно документированы на официальном сайте PostgreSQL, а также на множестве сторонних ресурсов. При этом и настройка, и администрирование одинаковые как для российской ОС Astra Linux, так и для открытых систем, например, Debian.

5. Установка интегрированной среды разработки MonoDevelop

На тех компьютерах, на которых ведется собственно разработка компонентов информационных систем, должна быть установлена интегрированная среда разработки (IDE), ориентированная на язык C# и включающая поддержку web-технологий и сетевой архитектуры .NET. Рассмотрим установку IDE MonoDevelop — одного из главных средств разработки программ на C# для ОС Linux.

В настоящее время интегрированная среда программирования MonoDevelop для ОС Linux относится к числу свободно распространяемых программ. Однако она не входит в реестр российского программного обеспечения [7], и ее использование на рабочих местах и серверах может противоречить корпоративным стандартам. Так как разработка сколько-нибудь масштабной информационной системы без IDE практически невозможна, то приходится применять затратные решения. Например, использовать несколько выделенных инструментальных компьютеров исключительно для разработки информационной системы, а на рабочие места и серверы переносить только готовые сборки (исполняемые модули и библиотеки функций). Установка MonoDevelop на инструментальный компьютер выполняется после установки исполнительной среды Моно и включает следующие шаги.

1. Получение открытого ключа репозитория проекта Моно:

```
sudo apt-key adv --keyserver hkp://keyserver.
ubuntu.com:80 \
     --recv-keys 3FA7E0328081BFF6A14DA29AA6A19B38D
3D831EF
echo "deb http://download.mono-project.com/repo/
debian vs-buster\
main" | sudo tee /etc/apt/sources.list.d/mono-
official-vs.list
```

2. Установка пакета MonoDevelop с помощью команд

```
sudo apt update
sudo apt install monodevelop
```

6. Начальная коррекция исходного кода C# в процессе миграции Windows—Linux

На начальном этапе миграции Windows—Linux с точки зрения присутствующего программного кода C# нужно обратить внимание на перечисленные далее обстоятельства.

- Построение файловой системы Windows и Linux разнится. Причем это касается и методов внешнего обращения. Например, в Linux не используется понятие «Логический диск C». То есть начальная точка пути к одному и тому же файлу для разных систем разная. Также отличаются разделители между именами каталогов в пути к файлу. Кроме того, важно помнить, что если в ОС Windows обращение к файлу не зависит от регистра букв, то в Linux файлы Sample.txt и sample.txt — это различные файлы. По этой причине возможное применение в программах непосредственного обращения к файлам требует коррекции программного текста.
- В Windows для обращения к средствам ОС используются системные функции, аналоги которых могут отсутствовать в Linux. Кроме того, для программ в среде Windows возможна работа с использованием системного реестра, который в Linux просто не существует. Поэтому следует выделить в программе фрагменты, обеспечивающие вызовы системных функций и работу с реестром, и провести их замену или коррекцию.

Нужно отметить, что исполнительная среда Моно изначально разрабатывалась как полный аналог фреймворка .NET. Поэтому очень большая часть кода за исключением графического интерфейса может запускаться и работать как есть, не требуя изменений. Кроме указанных выше случаев, когда требуется обязательное изменение кода. Дополнительно можно отметить возможность замены криптофункций, непосредственно привязанных к Windows. Однако, если рассматривать не низкоуровневое использование криптографических функций, а применение стандарта CMS/PKCS#7, то переход фактически не требует исправлений.

Чтобы миграция проходила с минимальными изменениями текста можно заранее скорректировать определенным образом проекты на языке C#, разработанные в интегрированной среде разработки Visual Studio под Windows.

При этом важно, что после трансляции как для Windows, так и для Linux программа представлена на языке IL, который интерпретируется в обеих средах. Это означает, что целесообразно в процессе выполнения определять ОС, в которой работаем, и в зависимости от результата выходить на выполнение той или иной ветки программы.

Здесь можно применить функцию, которая определяет, идет ли выполнение кода в настоящее время в системе типа Unix-Linux:

```
public static bool IsLinux
{
            get
            {
                int flag = (int)Environment.
OSVersion.Platform;
                return (flag == 4) || (flag == 6);
            }
}
```

А теперь посмотрим, как используется эта функция. Например, при вычислении абсолютного пути файла с каталогом журналов можно применить функцию «Каталог журналов»:

```
/// <summary> Каталог журналов</summary>
public static string DirLog
{
     get { return IsLinux ?
     @"/var/log/sb/" :
     @"c:\sb\log"; }
}
```

В этом примере получаем путь для записи журналов. В зависимости от того, в какой среде запущена программа, на выходе получаются разные результаты.

Если требуется определить каталог запуска программы, то рекомендуется использовать свойство текущей сборки Location, которое одинаково хорошо работает в ОС как Windows, так и Linux:

```
string startPath = System.IO.Path.GetDirectoryName(
     System.Reflection.Assembly.GetExecutingAssembly().
Location);
```

Отметим также, что для устранения неоднозначности использования разделителей между именами каталогов необходимо в windows-проекте использовать класс Path из пространства имен System.IO. Таким образом, например, создается имя файла:

```
string config = startPath + @"\preferences\
version\2.0\connection.config";
```

Однако это подходит только для Windows. Универсальная запись, которая может быть использована в каждой из применяемых ОС, создается так:

```
string config = = System.IO.Path.
Combine(startPath,
     "preferences", "version", "2.0", "connect.
config");
```

7. Перенос графического клиентского приложения в среду Linux

В качестве клиента (графического клиентского приложения) в информационной системе может использоваться web-браузер, что достаточно распространено. Если это так, то клиент готов и не требуется на этом этапе проводить какие-либо действия.

Рассматриваем также случаи, когда клиентом может быть сервис-браузер [11] и даже специально сконструированное приложение. Предполагается, что в том и другом случаях используется язык C#. В идеальном варианте, при совпадении клиентских программ Linux—Windows достаточно было бы провести трансляцию программ клиента в среде Linux. Однако сделать это не так просто, так как в качестве среды исполнения Windows использует фреймворк .NET (Windows Forms или WPF), а последние версии Моно для Linux применяют для отображения графического интерфейса пользователя (GUI) кроссплатформенный фреймворк GTK версии 2.20 и языковую привязку GTK#.

Таким образом, код, работающий с графическим интерфейсом, требует изменения при переносе в Linux. Механизм работы GUI в Windows, использующий события, аналогичен механизму для Linux, в частности, GTK. Однако вместо термина «событие» используется термин «сигнал», вместо «компонент» — «виджет». Несмотря на то что механизмы схожи, потребуется заново проектировать оконные формы и взаимодействие с ними. Это связано с тем обстоятельством, что часто события не имеют аналогов среди сигналов, а управляющие методы/свойства видом и поведением компонента сильно отличаются от методов/свойств виджета.

В силу изложенных выше соображений в процессе миграции на Linux часто возникает необходимость изменять логику поведения программы, причем часто эти изменения индивидуальны. Например, в Windows есть события, реагирующие на одиночное нажатие кнопки и на двойное нажатие кнопки мыши. В Моно и GTK# есть только сигнал одиночного нажатия на кнопку мыши. Из этого можно сделать вывод, что в этом случае либо нужно отказаться от обработки двойного клика, если это действие не важно, либо, используя только одно событие, реализовать реакцию программы на двойной щелчок.

Обратим внимание на важные вопросы, возникающие при переносе ПО в среду исполнения Моно для ОС Linux.

Основным вопросом, возникающим при разработке настольных приложений, является некоторая ограниченность дизайнера графических форм для GTK#. Некоторые действия, относящихся к проектированию интерфейса, можно выполнить только программно.

Возьмем программу, где на форме есть метка, у которой необходимо увеличить размер шрифта. Данное действие в дизайнере GUI с помощью панели свойств сделать нельзя. Для изменения размера шрифта потребуется написать код и поместить его, например, в конструктор окна:

```
labelName.ModifyFont(Pango.FontDescripton.
FromString("Normal bold 14"));
```

Можно рассмотреть еще один пример. Предположим, что при создании экранной формы требуется использовать на кнопках изображения (иконки). Среда разработки предоставляет программисту возможность выбора изображений для этой цели. После трансляции и запуска программы в некоторых ОС из семейства Linux изображения на кнопках не будут видны. Задача решается добавлением следующей строки в исходный код программы для каждой кнопки:

```
buttonName.Image.Visible = true;
```

Следует выделить еще один важный вопрос — это поведение модальных окон. Если в Windows у окна включен флаг модальности, то оно будет лежать поверх всех остальных окон этого приложения. В Linux флаг модальности блокирует действия на основной форме, но не обеспечивает возможность нахождения модального окна поверх основного. В Astra Linux можно случайно нажать на основную форму и приложение переместит ее поверх остальных, в том числе закрывая модальное окно. Так как действует блокировка действий на основной форме, в случае полнооконного приложения вернуть на место модальное окно сложно. Для обеспечения корректного поведения программы в случае модальных окон дополнительно необходимо использовать свойство TransientFor, записав в него ссылку на родительское окно.

Рассмотрим пример, демонстрирующий создание модального окна. Для упрощения воспользуемся стандартным диалоговым окном. В случае использования Windows будет представлен следующий исходный код:

```
public static bool ShowQuestion(string text,
string title)
{
        if (MessageBox.Show(text,
                  title,
                  MessageBoxButtons.YesNo,
                  MessageBoxIcon.Question) ==
                  DialogResult.Yes) return true;
        return false;
}
```

Эта функция создает диалоговое окно с вопросом, текст которого передается в параметре text. В качестве заголовка будет значение параметра title. У появившейся формы будет две кнопки: «Да» и «Нет». Функцией будет возвращено значение true, если пользователем нажата кнопка «Да», и false в ином случае.

Функция, реализующая аналогичное поведение в Моно, имеет следующий вид:

```
public static bool ShowQuestion(string text,
string title, Window parent)
   {
          if (text == null) text = "";
          if (title == null) title = "";
          if (text.Length > 300) text = text.
          Substring(0, 300);
          if (title.Length > 300) title =
          title.Substring(0, 300);
```

```
            text = text.Replace("<", "&#60;");
            MessageDialog md = new
            MessageDialog(parent,
                 DialogFlags.Modal,
                 MessageType.Question,
                 ButtonsType.YesNo,
                 text);
            md.Title = title;
            md.TransientFor = parent;
            ResponseType response =
            (ResponseType)md.Run();
            md.Destroy();
            if (response == ResponseType.Yes)
            return true;
            return false;
}
```

Как видно, даже у стандартного диалогового окна необходимо в свойство TransientFor записать ссылку на окно, из которого будет вызвана эта функция.

Отметим, что в приведенном выше тексте присутствуют еще несколько строк, разрешающих дополнительные вопросы.

Во-первых, нельзя передавать нулевые значения (null) в компоненты, связанные с GTK. Поэтому необходимо всегда проверять значения на null, что и происходит в первых двух строках функции. Переменным типа string можно вместо null присвоить пустое значение.

Во-вторых, передача слишком длинных строк в диалоговое окно GTK# может привести к падению всего приложения. Таким образом, следует обрезать слишком длинные значения строковых переменных.

Диалоговое окно вызывает исключение, если текст содержит знак «<». Решение для этого случая заключается в использовании HTML-кода этого знака.

Следует иметь в виду, что число визуальных компонентов для построения действительно удобного настольного приложения в GTK# не очень велико. Даже те компоненты, которые есть, обладают недостаточной функциональностью. Решение заключается в пополнении набора элементов, что на начальном этапе может быть очень ресурсоемко.

В тоже время, если исследовать более широкую задачу доступности дополнительных библиотек, которые не являются частью .NET Framework, но являются очень полезными, то можно найти частичное решение, которым является менеджер пакетов NuGet. В Моно этот менеджер пакетов может подключать сторонние библиотеки, разработанные для .NET. Следует отметить, что с библиотеками надо обращаться осторожно — некоторые не будут работать, в том числе, опирающиеся на системные функции Windows или использующие графический дизайн (Windows Forms, WPF). Отметим, что даже без этого список доступных пакетов большой и разнообразный. Среди списка работающих можем отметить библиотеки работы с базами данных, например, библиотеку Npgsql для работы с базой данных PostgreSQL.

Таким образом, разрешение основных проблемных вопросов заключается в более углубленной работе с исходными текстами проекта. В связи с этим обстоятельством скажем несколько слов о структуре проекта. Она в целом аналогична структуре проекта Visual Studio, за исключением дополнительного пункта «Пользовательский интерфейс». Как видно, в отличие от среды Microsoft, где каждый модуль с визуальным содержимым содержит свой собственный подпункт, в данной среде разработки эти модули дублируются в отдельном пункте. В данной папке содержатся файлы, созданные автоматически системой для визуальных компонентов и взаимодействия с ними.

Если рассматривать каталог проекта, то этот пункт соответствует папке gtk-gui. В то же время можно заметить два важных файла из этого каталога — gui.stetic и generated.cs, которые не показаны в MonoDeveloper. Это файлы GUI дизайнера Моно GTK#. Файл gui.stetic — это исходный код графического интерфейса в формате XML, на основе которого уже создаются файлы partial-классов с описанием графического интерфейса на языке C# для объединения с основным исходным кодом. Файл generated.cs отвечает за общее поведение экранных форм.

Важно отметить, что несмотря на отмеченные вопросы, процесс переноса графических приложений из Windows в Linux вполне осуществим. Более того, как показывает практика, после переноса хотя бы одного проекта эти вопросы начинают восприниматься программистами скорее как особенности MonoDevelop.

8. Перенос web-сервисов в среду Linux

Парадоксальным является то обстоятельство, что миграция таких сложных объектов, как web-сервисы, по маршруту Windows—Linux не является такой трудоемкой задачей, которой она поначалу представлялась авторам. Тем не менее здесь имеется ряд особенностей, которые должны быть учтены.

Создание web-сервиса начинается с создания проекта, а создание проекта — с выбора шаблона типа «Пустой проект ASP.NET» (рис. 4).

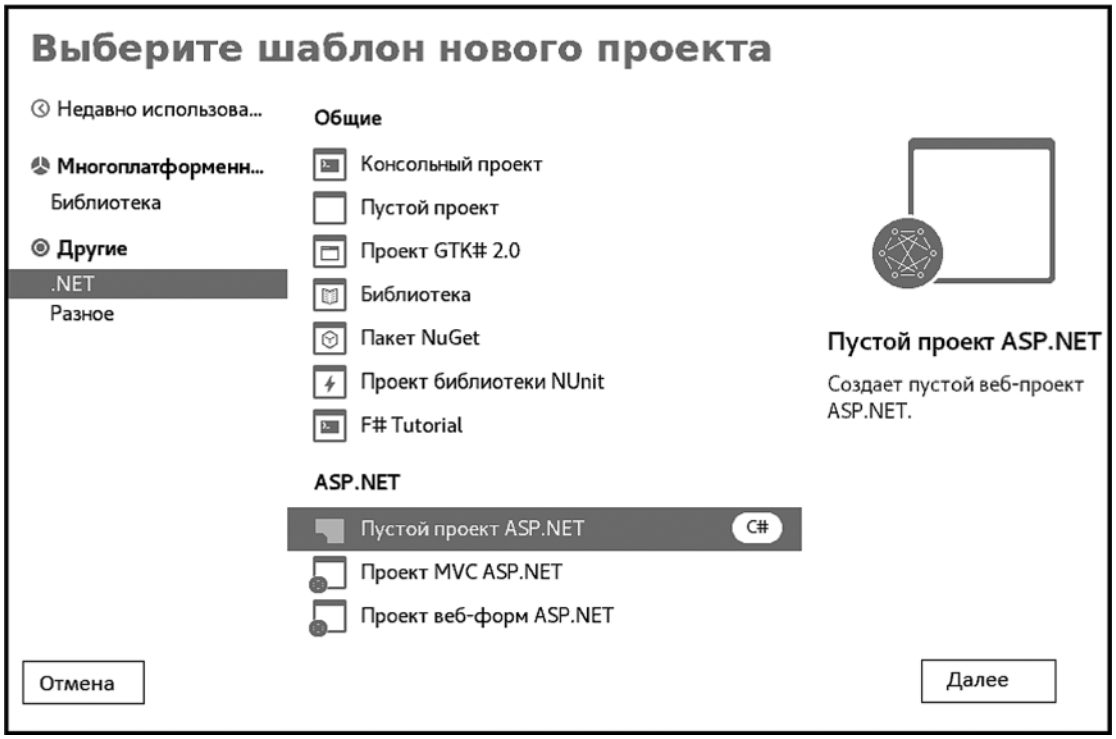


Рис. 4. Выбор шаблона проекта для web-сервиса

Последующие шаги разработки мало отличаются от тех, к которым привыкли пользователи Microsoft Visual Studio. К их числу относятся выбор названия проекта и директории проекта, выбор версии исполнительной среды, подключение к проекту необходимых библиотек функций, создание одного или нескольких программных модулей на C#, содержащих описание классов web-сервиса и сервисных функций, и т. п. Поэтому сосредоточимся лишь на отличительных особенностях разработки, характерных для MonoDevelop и ОС Linux.

К основным из этих особенностей можно отнести перечисленные далее.

- В проект web-сервиса должны быть включены три библиотеки функций из репозитория компании «Лаборатория 50»: EntityFramework.dll, EntityFramework6.Npgsql.dll и Npgsql.dll. Эти библиотеки автоматически заносятся в директорию /usr/lib/mono/gac в результате установки пакета libentityframework6-npgsql-cil (см. разд. 2).
- После компиляции проекта следует перенести ряд файлов из директории проекта на инструментальной машине в корневую директорию виртуального хоста на web-сервере. К этим файлам относятся текстовый файл web.config (конфигурационный файл web-сервиса), текстовый файл с расширением asmx, содержащий стартовый скрипт web-сервиса, и двоичные файлы-сборки, содержащие библиотеки сервисных функций и вспомогательные библиотеки. На рис. 5 проиллюстрирован результат переноса файлов из директории проекта web-сервиса с именем demowebsrv в корневой каталог уже знакомого нам виртуального хоста ipu.demohost.ru. Предполагается, что программа web-сервиса demowebsrv использует интерфейсную библиотеку Npgsql.dll для выполнения обращений к СУБД PostgreSQL.

Как видно на рис. 5, корневая директория web-сервиса содержит только файлы, создан-

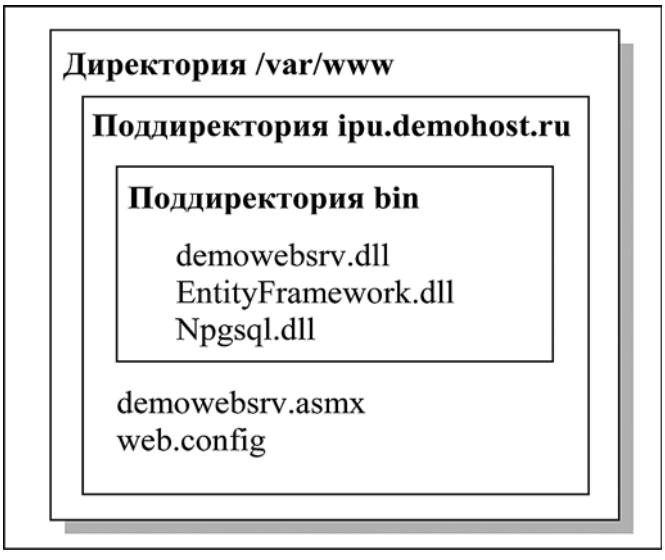


Рис. 5. Пример размещения файлов в корневой директории web-сервиса

ные разработчиком (web.config, demowebsrv.asmx и demowebsrv.dll), и библиотеки функций от Лаборатории 50 (EntityFramework.dll и Npgsql.dll) и не содержит каких-либо иных библиотек от других производителей. Обращение к web-сервису должно выполняется через URL:

```
http://ipu.demohost.ru/demowebsrv.asmx
```

(разумеется, интернет-имя ipu.demohost.ru должно быть определено или в файле /etc/hosts клиента, или в обслуживающем DNS-сервере, причем в качестве IP должен быть задан интернет-адрес web-сервера).

Следует отметить, что разработка WCF-сервисов не рекомендована, так как некоторые компоненты не были реализованы, хотя необходимо признать, что технология может использоваться с ограничениями [12].

9. Миграция баз данных

Как уже отмечалось выше, информационные системы, базирующиеся на языке C#, как правило, были ориентированы на использование СУБД Oracle и MS SQL Server. В рамках настоящей работы рассматривается переход в процессе импортозамещения на СУБД PostgreSQL и ее разновидности. В настоящее время разработано много методов, позволяющих проводить миграцию такого типа, например, описанный в работе [10].

Заключение

Вопросы организации импортозамещения информационных систем, для которых основным языком является C#, представляются непростыми. Это связано с тем обстоятельством, что язык C# изначально появился в среде ОС Windows. В то же время сложившиеся методы и способы импортозамещения, как правило, требуют использования ОС типа Linux. Несмотря на то что наличие открытых трансляторов, входящих в реестр российского программного обеспечения, во многом снимает обозначенные вопросы, работа с графическим интерфейсом в разных средах, к сожалению, во многом проводится по-разному. Это определяется тем обстоятельством, что графика для Windows и Linux изначально строилась на основе оригинальных разработок, а для ее отображения использовались различные средства. Это означает, что практически невозможно оттранслировать в среде Linux исходный текст C#, включающий графические примитивы и написанный для Windows. С учетом обязательной коррекции кода, требующейся в процессе миграции, а также работы по переносу графического интерфейса пользователя, можно осуществить переход Windows—Linux при реальных затратах по трудоемкости. Использование существующих способов проведения перевода баз данных на СУБД типа PostgreSQL позволяет завершить перевод информационных систем, базирующихся на языке C#, в среды, соответствующие критериям импортозамещения.

Список литературы

1. **В поисках** альтернативы: варианты импортозамещения ПО в России. URL: https://aif.ru/boostbook/importozameshchenie-po.html?ysclid=lkwlesp6bu729683692#solut (дата обращения 07.08.2023).
2. **Крупин А.** Курс на импортозамещение: выбираем российские аналоги иностранного ПО. URL: https://3dnews.ru/1062353/russian-sofware-guide?ysclid=lkwocl18s7528099068 (дата обращения 07.08.2023).
3. **Бородин В.** История успеха Яндекс.Почты с PostgreSQL // Профессиональная конференция разработчиков высоконагруженных систем HighLoad++ 2016. URL: https://habr.com/ru/articles/321756/ (дата обращения 07.08.2023).
4. **Трошков С. Н.** Об опыте миграции приложений на свободно распространяемое программное обеспечение с открытым кодом // Вестн. НГУ. Серия: Информационные технологии. 2018. Том 16, № 2. С. 86—94. DOI: 10.25205/1818-7900-2018-16-2-86-94.
5. **Белышев Д. В.** Опыт импортозамещения в медицинской информационной системе «Интерин PROMIS Alpha» // Программные системы: теория и приложения. 2022. Том 13, № 4 (55). С. 93—109. DOI: 10.25209/2079-3316-2022-13-4-93-109.
6. **Беликов С. Б.** Локомотивы импортозамещения в сфере российских информационных технологий: госкорпорации и ведомства в одном вагоне // Аспирант. 2022. № 2. С. 22—26.
7. **Реестр** российского программного обеспечения. URL: https://reestr.digital.gov.ru/ (дата обращения 07.08.2023).
8. **Патракова А.** Минцифры определило три самых популярных российских Linux // CFNews. 01 Ноября 2022. URL: https://www.cnews.ru/news/top/2022-11-01_razrabotchikov_po_zastavyat?ysclid=ldu6l2154s660667031 (дата обращения 07.08.2023).
9. **Моно** — Лаборатория 50 (lab50.net). URL: https://lab50.net/моно (дата обращения 07.08.2023).
10. **Курако Е. А., Орлов В. Л.** К вопросу миграции баз данных из среды Oracle в среду PostgreSQL // Программная инженерия. 2022. Том 13, № 1. С. 32—40. DOI: 10.17587/prin.13.32-40.
11. **Курако Е. А., Орлов В. Л.** Сервис-браузеры для информационных систем // Программная инженерия. 2017. Том 8, № 9. С. 413—421. DOI: 10.17587/prin.8.413-421.
12. **Mono.** WCF Development. URL: https://www.mono-project.com/docs/web/wcf/ (дата обращения 07.08.2023).

Import Substitution of Information Systems based on the C# Language and Network Architecture

E. A. Kurako, Researcher, keaipu@yandex.ru,
R. E. Asratian, Researcher, rubezas@yandex.ru,
V. L. Orlov, Researcher, ovl@ipu.ru,
V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences,
Moscow, 117997, Russian Federation

Corresponding author:

Evgeniy A. Kurako, Researcher,
V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, 117997, Russian Federation
E-mail: keaipu@yandex.ru

Received on July 20, 2023
Accepted on August 11, 2023

The purpose of the article is to study methods of migration to other environments of information systems previously created in Microsoft Windows in the C# language. These environments either belong to the open class, or were developed directly in Russia. One of the main migration tools is the separation of systems into components. In this case, the transformation tasks of each component are solved separately.

Methods of transformation of information systems functioning in an environment based on operating systems, database management tools, web tools, frameworks and libraries are considered.

The main directions of preparation of the executive environment, the components of which are included in the register of Russian software, or provide open source codes, are highlighted.

Operating systems that meet the given criteria are selected. Basically, these systems are founded on Linux kernel. Methods for configuring the Apache 2 web server and creating a virtual host are defined. A brief assessment of database migration methods was carried out. The issues of correcting the C# source code when changing the operating environment and methods of converting the source text when changing the ways of working with the graphical user interface are considered. The main problems during the transition to the new system and ways to solve them are shown.

Taking into account the mandatory code correction required during the migration process and the development of a new graphical user interface, the transition from Windows to Linux is possible at real time-consuming costs. Using existing methods of migrating databases, web services and client applications allows you to complete the migration of information systems based on the C# language into environments that meet the criteria for import substitution.

Keywords: *import substitution, information system, C#, Astra Linux, Mono, PostgreSQL, Apache 2, graphical interface, web services*

For citation:

Kurako E. A., Asratian R. E., Orlov V. L. Import Substitution of Information Systems based on the C# Language and Network, *Programmnaya Ingeneria,* 2023, vol. 14, no. 10, pp. 471—481. DOI: 10.17587/prin.14.471-481.

References

1. **Looking** for alternatives: software import substitution options in Russia, available at: https://aif.ru/boostbook/importozameshchenie-po.html?ysclid=lkwlesp6bu729683692#solut (date of access 07.08.2023) (in Russian).
2. **Krupin A.** The course towards import substitution: choosing Russian analogues of foreign software, available at: https://3dnews.ru/1062353/russian-sofware-guide?ysclid=lkwocl18s7528099068 (date of access 07.08.2023) (in Russian).
3. **Borodin V.** Success story of Yandex.Mail with PostgreSQL, Professional conference of developers of igh-load systems — HighLoad++ 2016, available at: https://habr.com/ru/articles/321756/ (date of access 07.08.2023) (in Russian).
4. **Troshkov S. N.** On experience in migrating applications to the freely distributable open source software, *Vestnk NGU. Seriya Informacionnye technologii*, 2018, vol. 16, no. 2, pp. 86—94. DOI: 10.25205/1818-7900-2018-16-2-86-94 (in Russian).
5. **Belyshev D. V.** Import Substitution Experience in the Interin PROMIS Alpha Healthcare Information System, *Program systems: Theory and applications,* 2022, vol. 13, no. 4 (55), pp. 93—109. DOI: 10.25209/2079-3316-2022-13-4-93-109 (in Russian).
6. **Belikov S. B**. Locomotives of import substitution in the sphere of Russian information technologies: state corporationsand departments in one wagon, *Aspirant,* 2022, no. 2, pp. 22—26 (in Russian).
7. **Register** of Russian software, available at: https://reestr.digital.gov.ru/ (date of access 07.08.2023) (in Russian).
8. **Patrakova A.** The Ministry of Finance has identified three of the most popular Russian Linux, *CNews*, 01.11.2022, available at: https://www.cnews.ru/news/top/2022-11-01_razrabotchikov_po_zastavy at?ysclid=ldu6l2154s660667031 (date of access 07.08.2023) (in Russian).
9. **Mono.** Laboratory 50 (lab50.net), available at: https://lab50.net/%D0 %BC%D0 %BE%D0 %BD%D0 %BE/ (date of access 07.08.2023) (in Russian).
10. **Kurako E. A., Orlov V. L.** On the issue of migrating databases from Oracle to PostgreSQL, *Programmnaya Ingeneria,* 2022, vol. 13, no. 1, pp. 32—40. DOI: 10.17587/prin.13.32-40 (in Russian).
11. **Kurako E. A., Orlov V. L.** Service browsers for information systems, *Programmnaya Ingeneria*, 2017, vol. 8, no. 9, pp. 413—421, DOI: 10.17587/prin.8.413-421 (in Russian).
12. **Mono.** WCF Development, available at: https://www.mono-project.com/docs/web/wcf/ (date of access 07.08.2023).

УДК 004.95
DOI: 10.17587/prin.14.482-492

Г. Р. Воробьева, д-р техн. наук, проф., gulnara.vorobeva@gmail.com,
Э. Ф. Фарваев, аспирант, farvaev.emil@gmail.com,
ФГБОУ ВО Уфимский университет науки и технологий

Архитектура распределенной системы сбора и обработки геопространственных данных на основе паттернов веб-проектирования

Поступила в редакцию 15.06.2023
Принята к публикации 26.07.2023

Обсуждены вопросы повышения реактивности и стабильности работы веб-ориентированных геоинформационных систем. Предложено решение, основанное на интеграции известных паттернов веб-проектирования, которое обеспечивает высокую внутреннюю и низкую внешнюю связность модулей в плагинной архитектуре приложения. На примере данных по результатам регистрации значений параметров геомагнитного поля и его вариаций проведен вычислительный эксперимент, обеспечивающий повышение скорости серверного отклика приложений в целом на 47 %.

Ключевые слова: *геоинформационная система, веб-архитектура, паттерны веб-разработки, программные модули, связность модулей, микросервисная архитектура*

Для цитирования:

Воробьева Г. Р., Фарваев Э. Ф. Архитектура распределенной системы сбора и обработки геопространственных данных на основе паттернов веб-проектирования // Программная инженерия. 2023. Том 14, № 10. С. 482—492. DOI: 10.17587/prin.14.482-492.

Введение

В настоящее время одной из основных проблем в проектировании и разработке геоинформационных систем остается необходимость обработки большого объема информации, поступающей по различным протоколам доступа от распределенных источников. Веб-ориентированная реализация такого класса систем обеспечивает доступ к приложению широкому кругу конечных пользователей и должна заведомо рассматриваться в качестве приоритетной технологии разработки при создании соответствующего программного продукта.

Вследствие управления большим объемом гетерогенной информации (табличные, графические, мультимедиа, текстовые и иные данные) современные геоинформационные системы усложняются. В неменьшей степени усугубляет проблему и геопространственный характер данных, что также накладывает определенные ограничения (в том числе и регламентируемые стеком технологий и используемыми моделями и методами) на их обработку и хранение.

Анализ показал, что отличительной особенностью и значимым недостатком существующих решений является их узкопрофильность: информационные системы ориентированы под определенные сферы деятельности, подразделения специализированных организаций и пр. При этом большинство вычислительных операций не определяются спецификой деятельности и/или, например, видом регистрируемых процессов, по которым должны быть представлены данные. Однако модули по сбору, обработке, хранению данных, по сути, однотипные, реализованы в одних системах и полностью отсутствуют в других. Кроме того, немаловажным является тот факт, что нередко технически невозможно отследить стабильность работы каждого программного компонента в геоинформационной системе.

В большинстве случаев автоматизированное рабочее место пользователя прикладной гео-

информационной системы представляет собой тонкий или толстый клиент, обеспечивающий доступ к локальному и/или централизованному удаленному хранилищу данных по одному из известных сетевых протоколов. При этом важный вопрос сопряжен с низкой скоростью работы соответствующих приложений (как правило, имеет место двухуровневая клиент-серверная архитектура с небольшой реактивностью). Это обстоятельство, в свою очередь, может привести к снижению эффективности работы с информационной системой, а также к нежелательным коллизиям, связанным с недостаточно эффективным способом организации мультипользовательской работы в рамках такой программной архитектуры. Возникает актуальная задача разработки архитектуры и инфраструктуры информационной системы, позволяющих повысить реактивность существующих веб-ориентированных геоинформационных систем прикладного характера, с одной стороны, и усилить режим их многопользовательской работы, с другой стороны.

Состояние вопроса

Задачу распределения вычислений в современном мире могут решать при помощи различного программного обеспечения. Используемые в такого рода программных средствах подходы могут различаться, так же как и условия их эффективного применения. Распространенным примером набора программных решений задачи распределения вычислений является программный комплекс Apache Hadoop (далее — Hadoop). Этот комплекс получил большое распространение в программных продуктах, утилизирующих распределенные вычисления, в частности, в веб-приложениях, работающих на большую аудиторию и подвергающихся большим вычислительным нагрузкам [1—4]. Программный комплекс Hadoop состоит из распределенной файловой системы, системы-планировщика заданий и управления кластером вычислительных узлов, а также системы MapReduce, представляющей собой платформу для выполнения распределенных вычислений. Параллельная обработка информации при использовании Hadoop позволяет добиться повышения реактивности веб-приложений путем уменьшения времени отклика сервера.

Одним из недостатков программных решений на основе Hadoop и MapReduce является невозможность мультипредикатных запросов к данным, что может сильно ограничить применимость такой технологии для ряда программных систем. При необходимости получения более комплексных результатов данное ограничение возможно обойти путем многократных однопредикатных запросов к распределенной системе. Разумеется, такой подход может негативно сказаться на скорости выполнения сложных запросов, так как комбинирование запросов предполагает более интенсивное использование дискового пространства, а также более высокую нагрузку на сеть, что, в свою очередь, сопровождается сильным замедлением работы программного комплекса при плохом или нестабильном соединении.

При работе с Hadoop процессы разбиваются на так называемые маперы и редьюсеры. Их число остается постоянным во время выполнения запроса. Вычислительные ресурсы распределяются между данными типами процессов без возможности перераспределения, т. е. в ситуации завершения работы маперами дополнительные ресурсы для редьюсеров выделены не будут. Технология Apache Spark [2, 5] является развитием MapReduce и призвана частично решить указанные выше проблемы, но также не лишена недостатков, в частности, трудностей с параллельным программированием мультипредикативных запросов, сложности интеграции аналитики и др.

Другой пример решения задачи распределенных вычислений — технология MOMIS [3, 6—8], в основе которой лежит медиатор — программный модуль для создания прикладных информационных систем с учетом характеристик и объемов данных.

Следует также отметить существование технологии IBM InfoSphere [9], основное назначение которой заключается в формировании платформы интеграции различных источников данных, в том числе распределенных. Такое решение обеспечивает масштабируемость системы интеграции данных и поддержку аналитических программных средств и механизмов контроля качества.

Все упомянутые решения предоставляют инструментальные средства для реализации парадигмы распределенных вычислений на различных вычислительных узлах. Важно отметить, что такие узлы не обязательно должны быть распределены географически, они могут находиться на одной машине. Подобная конфигурация, очевидно, имеет свои ограничения, в частности, в вычислительных мощностях. Однако она имеет и сильное преимущество в виде гораздо более эффективного обмена сигналами или сообщениями между узлами [10].

В контексте настоящей статьи речь идет о работе с географическими пространственными данными, которые являются данными специального назначения. Рассмотренные ранее программные

средства не учитывают подобной специфики, и создание прикладной геоинформационной системы с их использованием связано с определенными сложностями и ограничениями, которые не позволяют эффективно удовлетворить растущие потребности потребителей пространственно зависимой информации. В то же время развитие технологий визуализации данных, в частности, геопространственных, указывает на то, что данное направление анализа данных является одним из наиболее перспективных.

Таким образом, возникает актуальная задача разработки распределенной архитектуры геоинформационной системы, которая, с одной стороны, обеспечивает высокую реактивность при веб-ориентированной реализации, а также предоставляет возможность масштабирования, с другой стороны, позволяет интегрировать взаимонезависимые микросервисные программные компоненты. При этом ожидается, что каждый программный микросервис реализует укрупненную функцию или совокупность функций, объединенных семантически, административно или по иным критериям соответствующей предметной области.

Постановка задачи

Очевидным решением задачи повышения реактивности и масштабируемости веб-ориентированной геоинформационной системы является применение микросервисной архитектуры [11], которая предполагает формирование веб-ориентированного приложения на основе совокупности независимых программных модулей. При этом предполагается, что каждый программный модуль (микросервис) может быть интегрирован в сторонние программные проекты (например, по технологии CDN) таким образом, чтобы стать его частью, оставаясь при этом составляющей исходной геоинформационной информационной системы.

Представляется целесообразным отметить, что при реализации микросервисной архитектуры основным проблемным вопросом является ослабление связности составляющих ее программных модулей. Задача при этом сводится к тому, чтобы программные модули минимально зависели (в наилучшем случае — не зависели вовсе) друг от друга. Тогда с точки зрения каждого программного модуля все остальные микросервисы представляются своеобразными «черными ящиками», для которых может быть известен некоторый набор входных и выходных параметров. При этом полностью инкапсулируется логика работы программы и используемые при этом внутренние промежуточные параметры/переменные.

Основой предлагаемого подхода представляется использовать плагинную архитектуру, реализующую принцип открытости/закрытости. Данный принцип заключается в том, что программные сущности (классы, модули, функции и т. п.) являются открытыми для расширения, но закрытыми для модификации. Такой подход призван обеспечивать гибкость и расширяемость программной системы. Он предполагает возможность быстрого внесения изменений в программу, а также способность добавлять в систему новые сущности и функции, не нарушая ее основной структуры. При этом рассматриваемая архитектура также обеспечивает возможность повторного использования программных модулей в сторонних системах.

При реализации плагинной архитектуры первостепенным является решение задачи снижения сложности приложения. Основным решением этой задачи является функциональная декомпозиция, ведущая к модульной иерархии внутри приложения. При этом подразумеваются декомпозиция программы на подсистемы (функциональные модули, сервисы, слои, подпрограммы) и организация их взаимодействия друг с другом и внешним миром. На базовом уровне система декомпозируется на бизнес-логику, данные и интерфейс. При дальнейшем разбитии модули должны быть сгруппированы сообразно выполняемым функциям в прикладной области.

При декомпозиции программной системы необходимо стремиться к минимальной зависимости модулей друг от друга, с одной стороны, и максимальной зависимости внутренних компонентов модуля (принцип High Cohesion + Low Coupling [12]), с другой стороны. При этом высокая сопряженность компонентов внутри модуля проявляемся в том, что модуль сфокусирован на решении одной узкой задачи и не предполагает выполнение разнородных функций.

Большинство хорошо зарекомендовавших себя методик проектирования информационных систем предполагает иерархическую декомпозицию программной системы на составляющие (компоненты). При этом выделение компонентов осуществляется последовательно: сначала система делится на укрупненные подсистемы, те, в свою очередь, — на более мелкие и т. д. Предполагаемая веб-ориентированная реализация геоинформационной системы позволяет при таком подходе отделить, как минимум, представление от бизнес-логики и данных, что реализовано большинством шаблонов веб-приложений.

Декомпозиция системы на программные модули завершается при достижении каждым из модулей следующих критериев. Во-первых, каждый модуль реализует одну функцию и может без дополнительных средств / модулей выполнять полный набор вычислительных операций для реализации своей функции. Во-вторых, модулем реализуется стандартный принцип «вход — процесс — выход». Это означает, что модуль получает на вход определенный набор значений и возвращает один набор выходных данных. В-третьих, модуль логически независим, т. е. результат его работы определяется только набором входных данных и не зависит от других программных модулей. В-четвертых, как следует из обозначенного выше принципа связности программных модулей (High Cohesion + Low Coupling), информационные связи модуля с другими модулями должны быть минимизированы.

Одним из рекомендуемых способов ослабления связи между программными модулями является введение в архитектуру дополнительных модулей, реализующих паттерн «Наблюдатель» (Observer) [13, 14]. Указанный паттерн использует так называемую систему рассылок, предполагающую наличие в архитектуре основного модуля, отправляющего всем остальным модулям (подписчикам) одинаковые сообщения. Модули, заинтересованные в сообщении, реагируют на него (подписываются). Такой подход позволяет основному модулю (так называемому ядру приложения) оставаться независимым и при этом взаимодействовать с остальными модулями.

Еще один способ ослабления связи между модулями предусматривает применение веб-паттерна «Посредник» (Mediator) [15]. В этом случае в архитектуру также должен быть введен дополнительный модуль, который выступает в качестве так называемого центра связи между всеми остальными программными модулями. Поступающий от модуля запрос направляется непосредственно в медиатор, где выполняется его анализ на предмет определения модуля-адресата. В результате взаимодействие программных модулей реализуется опосредованно, фактически модуль-посредник инкапсулирует взаимодействие между множеством модулей.

Развитием паттерна «Посредник» является замена прямых зависимостей на синхронизацию через общее ядро. Для реализации подхода в архитектуру вводится дополнительный модуль — обобщенный посредник, при этом все остальные программные модули автоматически становятся его клиентами, использующими функции посредника или выполняющими обработку доступных в нем данных. Фактически при таком подходе обобщенный посредник выступает в роли буфера общих команд остальных программных модулей, с одной стороны, и промежуточных данных, пересылаемых модулями, с другой стороны. Реализация указанного подхода позволит программным модулям взаимодействовать через посредника и не знать ничего друг о друге.

В случае описанного подхода роль обобщенного посредника может быть двоякой. С одной стороны, он может быть полностью независимым от остальных программных модулей в архитектуре, с другой стороны, ядро-посредник может располагать информацией о модулях-клиентах и даже управлять ими. Так, к примеру, если на уровне веб-ориентированного приложения реализовать обобщенный посредник на серверной стороне, то к нему могут независимо подключаться различные интерфейсные модули, отвечающие, в частности, за клиентский рендеринг и визуализацию.

Каждый из представленных паттернов проектирования веб-приложений, безусловно, является основанием для построения так называемой «чистой» архитектуры, эффективной для применения в различных окружениях, устойчивой к пиковым нагрузкам, а также упрощающей процедуру тестирования на различных этапах жизненного цикла приложения. Немаловажным является сохранение и даже повышение характерных для модульной структуры характеристик гибкости и масштабируемости приложения, возможности простой трансформации в микросервисную архитектуру со многими входными точками.

Вместе с тем перечисленные шаблоны проектирования не лишены недостатков. Так, к примеру, шаблон «наблюдатель» не позволяет отследить непредвиденные обновления в субъектах, что значимо усложняет его работу. При реализации шаблона «посредник» размеры самого модуля-посредника могут существенно возрасти по мере расширения архитектуры программы, при этом увеличение количества модулей-посредников может привести к чрезвычайно перегрузке архитектуры и нивелированию всех положительных качеств модульной структуры приложения. При этом, как говорилось выше, преимущества каждого из рассматриваемых шаблонов неоспоримы.

В связи с этим возникает актуальная задача разработки нового подхода к проектированию архитектуры веб-ориентированной информационной системы на базе двух распространенных шаблонов («наблюдатель» и «посредник»). Синтезированная на основе этих двух шаблонов архитектура должна унаследовать их преимущества по связности

элементов модульной структуры, однако при этом снизить негативный эффект от характерных для этих паттернов недостатков.

Формализация решения задачи

Представляется целесообразным использовать формализованное описание задачи и ее решения посредством соотношений в терминах теоретико-множественного подхода для повышения информативности соответствующих теоретических положений и формулировок. В связи с этим архитектуру географической информационной системы M представляется необходимым описать в виде совокупности модулей различного уровня. При этом принцип отделения данных от представления позволяет в первом приближении разделить все множество программных модулей следующим образом:

$$M = \{F,\ V,\ D\}, \tag{1}$$

где F — подсистема бизнес-логики; V — подсистема визуализации; D — подсистема хранения и обработки геопространственных данных.

В соответствии с обозначенной выше схемой иерархической декомпозиции каждая из выделенных подсистем не является атомарной и должна быть разделена на программные составляющие по заданному критерию. Такой составляющей может являться, в частности, функция или группа семантически объединенных функций. Подсистему бизнес-логики F, например, можно представить следующим образом:

$$F = \{f_1, \ldots, f_k\},\ k \in N;\ f_k = \{f_{k1}, \ldots, f_{kn}\},\ n \in N, \tag{2}$$

где f_k — модуль подсистемы визуализации (всего может быть k модулей); f_{kn} — модуль модуля f_k подсистемы визуализации (всего может быть n модулей).

Аналогичным образом должны быть выделены модули в оставшихся подсистемах — визуализации и работы с данными. При этом декомпозиция каждого модуля завершается разработчиком в том момент, когда модуль становится атомарным, выполняет одну функцию и имеет единственный вход и единственный выход.

В результате будет сформирована плагинная архитектура F', такая, что каждый ее отдельный модуль f будет характеризоваться только протоколом и интерфейсом своего использования. Обозначим как объект $P1$ запрос, поступающий к модулю f, и как объект $P2$ — сформированный модулем ответ (результат работы модуля). В терминах теории вычислительных процессов и структур указанные объекты можно связать соотношением вида

$$\begin{gathered} P1\, f \to P2 : P1 = (x_1, \ldots, x_m); \\ P2 = (y_1, \ldots, y_n);\ m, n \ni N, \end{gathered} \tag{3}$$

где входной объект $P1$ характеризуется набором значений m параметра x_i; выходной объект $P2$ — набором значений n параметра y_j.

Протокол общения с модулем представлен набором правил применения модуля сторонними программными модулями. Таким образом модуль инкапсулирует бизнес-логику и внутренние связности вычислительных процедур и данных от интерфейса взаимодействия с ним. Соответственно, каждое правило логики модуля характеризует процедуру формирования набора выходных значений из полученных входных данных:

$$b: P1 = f(x_1, \ldots, x_m) : P2 = (y_1, \ldots, y_n);\ m,\ n \ni N, \tag{4}$$

где b — множество правил бизнес-логики программного модуля f.

Интерфейс использования модуля определяет формат входной и выходной информации для модуля f_j, усиливая его программную инкапсуляцию с точки зрения сторонних программных модулей, с одной стороны, а также иных программных модулей в составе той же архитектуры (одного и того же приложения), с другой стороны:

$$\begin{gathered} \forall x_i,\ y_j\ \exists d = \{n,\ s,\ t\}, \\ i = 1, \ldots, m,\ j = 1, \ldots, n;\ m, n \ni N, \end{gathered} \tag{5}$$

где d — метаданные входного x_i или выходного y_j параметра, представленные соответственно именем параметра n, размерности s, типа данных t.

В соответствии с принципами построения архитектурного паттерна «Наблюдатель» в существующую архитектуру F должен быть введен дополнительный модуль-наблюдатель o:

$$\begin{gathered} F = \{f_1, \ldots, f_k,\ o\},\ k \in N; \\ f_k = \{f_{k1}, \ldots, f_{kn}\},\ n \in N. \end{gathered} \tag{6}$$

Остальные модули в архитектуре F становятся модулями-подписчиками основного модуля o. В связи с этим обстоятельством для модуля-наблюдателя формируется так называемый список рассылки, обобщенная структура которого может быть представлена следующим образом:

$$M = \{src,\ dst,\ msg\}, \tag{7}$$

где *src* — источник сообщения; *dst* — получатель сообщения; *msg* — текст сообщения (в частности, к примеру, формируемые в результате работы модуля данные).

В качестве источника и получателя сообщений указываются идентификационные параметры соответствующих модулей. Для корректной маршрутизации сообщений в составе модуля-наблюдателя присутствует дополнительная структура — реестр модулей архитектуры *F*, куда попадают все существующие программные модули, а также новые вводимые в информационную систему модули. В общем виде реестр можно представить следующим образом:

$$R = \left\{\{id_1\ name_1\ URI_1\},\ldots,\{id_k\ name_k\ URI_k\}\right\},\ k \in N. \quad (8)$$

В выражении (8) каждый программный модуль имеет уникальный идентификатор id_i и наименование $name_i$, при этом непосредственно наименование модуля является для модуля-наблюдателя опциональным и может отсутствовать. Обязательным является наличие параметра, характеризующего путь к программному модулю — универсальный идентификатор ресурса, по аналогии с адресом URL, поскольку в данном конкретном случае речь идет непосредственно о веб-ориентированной архитектуре геоинформационной системы.

Формируемые модулем-наблюдателем сообщения доступны всем зарегистрированным в архитектуре программным модулям в фоновом режиме. При обнаружении нового сообщения модуль сравнивает указанный в нем идентификатор источника с собственным и в случае их совпадения извещает об этом модуль-наблюдатель. В ответ модуль-наблюдатель направляет отозвавшемуся модулю полное сообщение по имеющемуся в реестре уникальному *URI* программного модуля. При этом представляется целесообразным отметить, что в рамках рассматриваемого паттерна программирования целевыми узлами могут быть представлены одновременно несколько модулей, и это не должно вызвать никаких коллизий.

В случае применения архитектурного паттерна «Посредник» создается так называемый основной модуль (для удобства он может быть помечен *o,* как в формуле (6)). Однако в отличие от предыдущего рассмотренного шаблона, формируемое основным модулем (модулем-посредником) сообщение доступно только самому программному модулю. В целом это означает, что сообщения генерируются не только модулем-медиатором, но остальными программными модулями в архитектуре. При этом источник сообщения автоматически заполняется по идентификатору сгенерировавшего его программного модуля. Важно отметить, что процедура регистрации программного модуля в реестре основного модуля (в данном случае — модуля-медиатора) предполагает также наличие необходимых метаданных для каждого компонента приложения, в том числе уникальный *URI*.

Важным недостатком обоих паттернов проектирования является отсутствие возможности обновления метаданных о присутствующих в архитектуре программных модулях. Действительно, при масштабировании приложения посредством добавления новых программных компонентов/модулей по умолчанию выполняется их регистрация в реестре модулей приложения, используемом основным модулем (наблюдателем или медиатором). При этом нельзя исключить такие ситуации, при которых в случае рефакторинга или по иным причинам один или более программных модулей перестанут существовать в проекте, либо поменяются их метаданные. Однако соответствующие изменения в реестре в этом случае, как правило, не предусмотрены. В совокупности это приводит к всевозможным коллизиям, нарушающим нормальное функционирование веб-приложения.

Отмеченный выше недостаток касается обоих описанных архитектурных паттернов и может быть описан следующим соотношением:

$$\begin{gathered} A = \{id\colon 'a1',\ name\colon 'A',\ URI\colon 'URI'\}; \\ M = \{'a2',\ 'a1',\ msg\}; \\ R = \begin{Bmatrix} \{id\colon 'a2',\ name\colon 'A2',\ URI\colon 'URI2'\}, \\ \{id\colon 'a3',\ name\colon 'A3',\ URI\colon 'URI3'\} \end{Bmatrix}, \end{gathered} \quad (9)$$

где *A* — произвольный программный модуль с собственными метаданными; *M* — сообщение в главном модуле *o*; *R* — реестр модулей в архитектуре.

Как видно из примера (9), присутствующий в архитектуре программный модуль не зарегистрирован в реестре основного модуля, однако ссылка на него имеется в одном из сгенерированных сообщений в качестве адресата. В результате наблюдается коллизия в маршрутизации сообщения. Аналогичные ситуации могут возникнуть в том случае, если при рефакторинге или модификации приложения один из модулей был переименован, перемещен или вовсе удален из архитектуры, что также приводит к всевозможным коллизиям в сообщениях.

В связи с отмеченным выше обстоятельством представляется целесообразным дополнить известные подходы триггерной функцией, выполняемой при создании, изменении или удалении

программного модуля из архитектуры приложения. При реализации функция становится архитектурной настройкой, которая с заданной периодичностью посредством сообщений «обходит» всю структуру приложения. Данный подход позволит отследить все возможные изменения, которые происходят с модулями геоинформационной системы, а также гарантировать избегание маршрутизационных коллизий в дальнейшем.

Представляется целесообразным использовать аналогичную триггерную функцию и на уровне отдельных программных модулей, входящих в архитектуру геоинформационной системы. Это означает, что операция добавления, изменения или удаления программного модуля из обобщенной структуры недопустима без отправки соответствующего сообщения центральному модулю. Очевидно, что по причине технических сбоев в системе могут возникнуть ситуации, при которых сообщение от модуля может не дойти от наблюдателя или медиатора. В этом случае описанный выше подход систематического мониторинга компонентов архитектуры позволит решить такую задачу.

Еще одним нововведением в архитектуру геоинформационной системы является совмещение паттернов «Наблюдатель» и «Посредник» в единый интегрированный шаблон. Такой подход означает, что по аналогии с паттерном «Наблюдатель» сообщения в информационном взаимодействии будут доступны всем модулям, но только декларативно. Согласно такому подходу, доступное всем сообщение будет содержать только идентификатор модуля-адресата, а непосредственно тело сообщения по-прежнему останется только в ведении управляющего модуля.

Далее по принципу работы паттерна «Наблюдатель» модуль, идентифицировавший свои данные в общедоступном сообщении, направляет ответное сообщение управляющему модулю. На последующем шаге управляющий модуль функционирует непосредственно по типу паттерна «Посредник». В этом случае после поступления сообщения-отклика от модуля-адресата управляющий модуль по известному ему идентификатору и параметрам *URI* адресно направляет сообщение искомому модулю.

Ожидаемым результатом от предложенного решения является, с одной стороны, отсутствие коллизий при информационном взаимодействии программных модулей в архитектуре геоинформационной системы, поскольку двухпотоковая актуализация зарегистрированных в системе компонентов исключает неизвестных или отсутствующих адресатов. С другой стороны, снижается трафик при межмодульном взаимодействии: при сохранении распространения сообщений по подписке от основного модуля, непосредственно тело сообщения направляется не всеобъемлющим потоком, а строго адресно, по отклику.

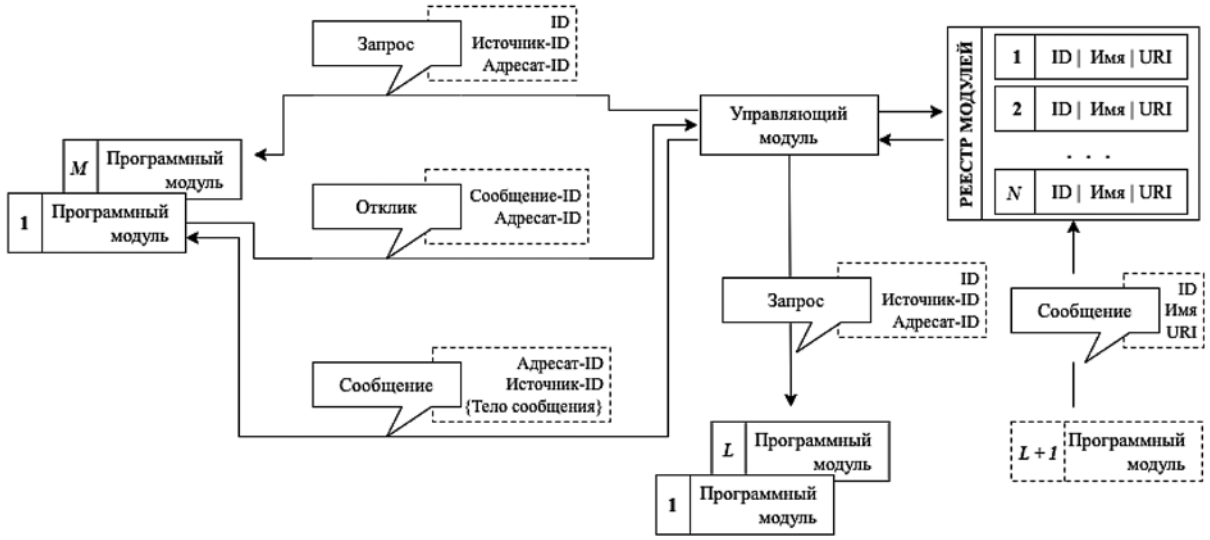
Архитектура решения

Основой практической реализации предложенного подхода является непосредственно архитектура соответствующей веб-ориентированной геоинформационной системы. Характерная для веб-приложений концепция клиент-серверного взаимодействия наследуется предлагаемой архитектурой, приняв соответствующие модификации в соответствии с предложенной интеграцией паттернов проектирования и сохранив успешно зарекомендовавший себя подход отделения данных от их представления.

В общем виде предлагаемая архитектура представляет собой описанную выше плагинную структуру, предполагающую в данном случае последовательную функциональную декомпозицию программных подсистем, модулей и классов геоинформационной системы. Отличительной особенностью и при этом сложностью соответствующей прикладной области является доминирование особой категории данных, условно обозначаемой как «персональные данные». Их сбор, хранение, обработка и анализ требуют особых подходов, обеспечивающих защищенность информации в соответствии с федеральным законодательством.

Для упрощения описания предлагаемой архитектуры на рисунке изображены L программных модулей, формирующих соответствующее приложение для геоинформационной системы без привязки к конкретным функциям. Такая нотация представляется оправданной ввиду того, что непосредственно функциональная декомпозиция программной архитектуры хорошо изучена и формализована на данный момент, что подтверждается, в частности, целым комплексом работ российских и зарубежных ученых и разработчиков. Центральным звеном рассматриваемой структуры является так называемый управляющий модуль, сочетающий в себе функции наблюдателя и медиатора из соответствующих архитектурных паттернов.

Взаимодействие между программными модулями осуществляется посредством стандартного сетевого протокола HTTPS, предполагающего обмен сообщениями, запросами и откликами между клиентами и серверами. При этом роли клиентов и серверов в данном взаимодействии приобретают несколько иной смысл по сравнению с извест-



Архитектура предлагаемого решения

ной системой клиент-серверного взаимодействия в веб-приложениях. В данном случае в качестве клиента и сервера рассматриваются последовательно адресат и отправитель сетевых сообщений, а по мере взаимодействия роли модулей могут меняться.

Как следует из приведенной на рисунке схемы, а также предложенных выше решений, инициатором информационного взаимодействия выступает управляющий модуль. Он формирует короткое сообщение, содержащее параметры адресата и идентификатор непосредственно сообщения. Сообщение направляется всем зарегистрированным программным модулям согласно доступному управляющему модулю реестру. Модуль-адресат реагирует на собственный идентификатор и направляет ответное сообщение управляющему модулю. В ответ управляющий модуль формирует новое сообщение непосредственно с передаваемыми параметрами и инструкциями и направляет их адресно запросившему модулю. Такое взаимодействие повторяется на протяжении всего сеанса работы геоинформационной системы.

Добавление в архитектуру нового программного модуля (на рисунке данный компонент помечен как $L + 1$) сопровождается формированием нового сообщения, содержащего весь необходимый набор метаданных для регистрации в соответствующем реестре программных модулей. Сформированное новым модулем сообщение упаковывается в пакет согласно используемому сетевому протоколу (аналогичная операция выполняется для всех сообщений в рамках информационного взаимодействия) и направляется управляющему модулю. Последний, в свою очередь, анализирует поступившее сообщение и фиксирует полученную информацию в реестре модулей. Запросившему подключение модулю в случае успешного завершения операции направляется сообщение, разрешающее ему участвовать в информационном взаимодействии.

Здесь представляется целесообразным отметить, что модули, не зарегистрированные в реестре программных модулей по тем или иным причинам, по умолчанию исключаются из информационного взаимодействия в архитектуре геоинформационной системы. При этом управляющий модуль с заданной разработчиком или администратором системы периодичностью опрашивает зарегистрированные в реестре модули посредством отправки служебного сообщения с соответствующим идентификатором искомого модуля. В случае, если на отправленное сообщение не получен отклик от модуля-адресата, последний исключается из реестра модулей системы.

Аналогичные вычислительные операции должны быть проведены в случае модификации метаданных или удаления программного модуля. Соответствующие действия в обязательном порядке сопровождаются информационным сообщением с обновленными метаданными и/или меткой удаления, направляемым управляющему программному модулю. Получив сообщение, последний вносит соответствующие изменения в реестр модулей или удаляет из него требуемую запись об убывающем модуле. По завершению операции управляющий модуль направляет запросившему разрешение операции модулю подтверждающее сообщение, тем самым завершая его транзакцию. В противном случае (если подтверждающего/разрешающего сообщения получено не было) изме-

нения в модуле не сохраняются, происходит откат соответствующей транзакции.

Апробация и оценка качества решения

Для оценки качества предложенных решений был разработан исследовательской прототип геоинформационной системы, комплекс реализуемых функций которого обеспечивает хранение, извлечение геофизической информации о значениях параметров геомагнитного поля и его вариаций. В качестве источников были использованы данные из веб-ориентированных сервисов SuperMAG[1] и INTERMAGNET[2]. На основании реальных данных были сформированы тестовые данные, в общем случае характеризующие информацию о пространственно-временной анизотропии геомагнитного поля, поступающую из неравномерно распределенных по земной поверхности магнитных обсерваторий и вариационных станций. Сформированный набор данных характеризует ежеминутное изменение параметров геомагнитного поля в течение календарного года в среднем в 300 точках земной поверхности.

Разработанный прототип геоинформационной системы представляет собой веб-ориентированное приложение, доступное конечным пользователям (в качестве которых в данном случае выступают ученые и специалисты геофизического профиля) по трехуровневой клиент-серверной архитектуре. Стек используемых программных технологий представлен фреймворком Django, обеспечивающим применение языка программирования Python для разработки серверных сценариев, с одной стороны, и кодирование клиентского компонента приложения на основе традиционной связки HTML5/CSS3/JavaScript, с другой стороны.

Для проведения сравнительного анализа, а также количественной и качественной оценки представленных решений был разработан аналогичный исследовательский прототип, реализующий характерный для фреймворка Django архитектурный паттерн «Модель — представление — контроллер». При этом предложенный подход к интеграции паттернов «Наблюдатель» и «Посредник» был намеренно исключен из данного решения.

Для вычислительного эксперимента были выделены несколько функций, выполнение которых представляется характерным для геоинформационной системы: получение данных из распределенных источников, их хранение, обработка и визуализация на клиентской стороне.

В ходе проведения вычислительных экспериментов была выполнена оценка стабильности и реактивности соответствующего исследовательского прототипа веб-ориентированной геоинформационной системы. В каждом из прототипов было проанализировано выполнение типовых функций по извлечению, модификации, удалению и визуализации геопространственной информации. В качестве тестовых были использованы данные относительно небольшого объема (менее 1 Гбайта).

Экспериментальный стенд на клиентской стороне был представлен персональным компьютером со следующими характеристиками: CPU Intel Core i5 10300H 2,5 ГГц (4 ядра, 8 потоков), оперативная память 4 ГБайта, скорость интернет-соединения ~52,4 Мбит/с. Результаты вычислительного эксперимента показали, что подход со стандартной архитектурой обеспечивает получение серверного отклика за 30...40 с при стабильном интернет-соединении. Предложенный подход за счет комбинирования архитектурных паттернов «Наблюдатель» и «Посредник» позволяет сократить значение указанного параметра до 15...20 с.

Кроме того, были проведены вычислительные эксперименты с тестовыми данными для оценки изменения скорости получения серверного отклика при направлении простого одномерного запроса к серверу. Подход, основанный на известной архитектуре, показал значение указанного параметра 35...50 с, а предложенный подход — 20....25 с. При этом были установлены пути повышения реактивности соответствующих веб-ориентированных приложений посредством применения технологии Node.js на сервере (предназначенной для работы с высоконагруженными приложениями).

Заключение

В настоящей работе предложено повысить реактивность и автономность вычислительных операций, реализуемых при проектировании и разработке веб-ориентированных геоинформационных систем. Такой подход призван увеличить эффективность информационных систем такого класса, одними из важнейших проблемных вопросов которого являются преимущественно узкая направленность, привязанная к определенной прикладной области, с одной стороны, и недостаточно высокая степень автоматизации соответствующих прикладных процессов, с другой стороны. Проведенный анализ показал, что подавляющее большинство известных подходов не решают или только частично разрешают обозначенные вопросы.

[1] https://supermag.jhuapl.edu/info/

[2] https://intermagnet.github.io/

Основной акцент в предлагаемом решении сделан на модернизацию архитектуры геоинформационной системы. При этом по умолчанию основой является веб-ориентированная реализация информационной системы, что позволяет сделать ее доступной широкому кругу пользователей. В представленной работе предлагается модернизировать существующий подход к проектированию веб-ориентированных геоинформационных систем посредством комбинирования популярных архитектурных паттернов веб-разработки «Наблюдатель» и «Посредник». Последовательная декомпозиция (программная, функциональная и пр.) геоинформационной системы приводит ее к так называемой плагинной архитектуре, отличительной особенностью которой является отделение данных от их представления и возможность повторного использования и адаптации программных компонентов.

Совместное применение обозначенных архитектурных паттернов позволяет эффективно использовать их преимущества, связанные с ослаблением внешней связности программных модулей. При этом характерные для указанных паттернов недостатки нивелируются соответствующей их архитектурной интеграцией.

Для оценки эффективности предложенных решений была проведена серия вычислительных экспериментов на примере сбора, хранения, обработки и визуализации разнородных геофизических данных небольшого объема (менее 1 Гбайта). Результаты проведенных вычислительных экспериментов (на тестовых данных, автоматически сформированных на основе реальных по пространственно-временным вариациям геомагнитного поля) в заданных технических условиях клиент-серверного веб-ориентированного взаимодействия показали, что применение предложенного подхода к решению перечисленных выше задач позволит существенно сократить время получения отклика от серверной части (по сравнению с традиционным подходом) в среднем на 47 %.

Список литературы

1. **Aji A., Wang F., Vo H.** et al. Hadoop-GIS: A High Performance Spatial Data Warehousing System over MapReduce // Proceedings of the VLDB Endowment. 2013. Vol. 6, No. 11. P. 1009—1020.
2. **Azhir E., Hosseinzadeh M., Khan F., Mosavi A.** Performance Evaluation of Query Plan Recommendation with Apache Hadoop and Apache Spark // Mathematics. 2022. Vol. 10, No. 19. Article 3517. DOI: 10.3390/math10193517.
3. **Sala A.** Data and Service Integration: Architectures and Applications to Real Domains: PhD thesis. Modena and Reggio Emilia, Emilia-Romagna, Italy, 2010. 147 p.
4. **Apache** Hadoop. URL: https://hadoop.apache.org/ (дата обращения 27.04.2023).
5. **Apache** Spark™ — Unified engine for large-scale data analytics. URL: https://spark.apache.org/ (дата обращения 27.04.2023).
6. **Magnotta L.** Analysis and development of advanced data integration solutions for data analytics tools: PhD thesis. Modena and Reggio Emilia, Emilia-Romagna, Italy, 2018. 176 p.
7. **Bergamaschi S., Beneventano. D., Corni A.** et al. The Open Source release of the MOMIS Data Integration System // SEBD 2011 Proceedings of the 19th Italian Symposium on Advanced Database Systems. 2011. P. 175—186.
8. **Momis** — DataRiver. URL: https://www.datariver.it/en/momis/ (дата обращения 27.04.2023).
9. **IBM** InfoSphere Information Server. URL: https://www.ibm.com/information-server (дата обращения 27.04.2023).
10. **Comparison** between different Observer Pattern implementations. URL: https://github.com/millermedeiros/js-signals/wiki/Comparison-between-different-Observer-Pattern-implementations (дата обращения 27.04.2023).
11. **Trabelsi I., Abdellatif M., Abubaker A.** et al. From legacy to microservices: A type-based approach for microservices identification using machine learning and semantic analysis // Journal of Software: Evolution and Process. 2022. DOI: 10.1002/smr.2503.
12. **Jiao. Q.-J.** Functional Units in Complex Networks: Beyond Cohesive Modules // IFAC Proceedings Volumes. 2013. No. 46. P. 94—99. DOI: 10.3182/20130708-3-CN-2036.00040.
13. **Sharma S.** Web pattern analysis using partitioning algorithm in hyperlink structure// International Journal of Emerging Trends in Engineering and Development. 2020. No. 10. DOI: 10.26808/rs.ed.i10v2.03.
14. **Bazeer Ahamed B., D. Yuvaraj D., Shitharth S.** et al. An Efficient Mechanism for Deep Web Data Extraction Based on Tree-Structured Web Pattern Matching // Wireless Communications and Mobile Computing. 2022. P. 1-10. DOI: 10.1155/2022/6335201.
15. **Yadav P.** A Novel Approach of Development of Web Pattern by Focusing on Web Structure Mining Techniques // International Journal on Recent and Innovation Trends in Computing and Communication. 2019. No. 7. P. 01—04. DOI: 10.17762/ijritcc.v7i2.5223.

Distributed Architecture of Geospatial Data Collection and Processing System Based on Web Design Patterns

G. R. Vorobeva, Professor, gulnara.vorobeva@gmail.com,
E. F. Farvaev, Postgraduate Student, farvaev.emil@gmail.com,
Ufa University of Science and Technology, Ufa, 450008, Russian Federation

Corresponding author:

Emil F. Farvaev, Postgraduate Student,
Ufa University of Science and Technology, Ufa, 450008, Russian Federation
E-mail: farvaev.emil@gmail.com

Received on June 15, 2023
Accepted on July 26, 2023

This article discusses the issues of improving the reactivity and stability of web-based geographic information systems. It proposes a solution based on the integration of well-known web design patterns, which provides high internal and low external coupling of modules in the plugin architecture of the application. A prototype system was developed utilizing web design and programming patterns such as Model-View-Controller pattern and message-based communication between modules. The experiment was conducted on the dataset based on the results of registering the values of the Earth's geomagnetic field parameters and its variations. The experiment demonstrates increased performance when using the suggested approach. The results show a reduction in server response time by 47 %.

Keywords: *geographic information system, web architecture, web development patterns, software modules, module connectivity, microservice architecture*

For citation:

Vorobeva G. R., Farvaev E. F. Distributed Architecture of Geospatial Data Collection and Processing System based on Web Design Patterns, *Programmnaya Ingeneria,* 2023, vol. 14, no. 10, pp. 442—492. DOI: 10.17587/prin.14.442-492.

References

1. **Aji A., Wang F., Vo H.** et al. Hadoop-GIS: A High Performance Spatial Data Warehousing System over MapReduce, *Proceedings of the VLDB Endowment*, 2013, vol. 6, no. 11, pp. 1009—1020.

2. **Azhir E., Hosseinzadeh M., Khan F., Mosavi A.** Performance Evaluation of Query Plan Recommendation with Apache Hadoop and Apache Spark, *Mathematics*, 2022, vol. 10, no. 19, article 3517. DOI: 10.3390/math10193517.

3. **Sala A.** *Data and Service Integration: Architectures and Applications to Real Domains*, Modena and Reggio Emilia, Emilia-Romagna, Italy, 2010, 147 p.

4. **Apache** Hadoop, available at: https://hadoop.apache.org/ (date of access 27.04.2023).

5. **Apache** Spark™ — Unified engine for large-scale data analytics, available at: https://spark.apache.org/ (date of access 27.04.2023).

6. **Magnotta L.** *Analysis and development of advanced data integration solutions for data analytics tools*, Modena and Reggio Emilia, Emilia-Romagna, Italy, 2018, 176 p.

7. **Bergamaschi S., Beneventano. D., Corni A., Kazazi E.** et al. The Open Source release of the MOMIS Data Integration System, *SEBD 2011 — Proceedings of the 19th Italian Symposium on Advanced Database Systems*, 2011, pp. 175—186.

8. **Momis** — DataRiver, available at: https://www.datariver.it/en/momis/ (date of access 27.04.2023).

9. **IBM** InfoSphere Information Server, available at: https://www.ibm.com/information-server (date of access 27.04.2023).

10. **Comparison** between different Observer Pattern implementations, available at: https://github.com/millermedeiros/js-signals/wiki/Comparison-between-different-Observer-Pattern-implementations (date of access 27.04.2023).

11. **Trabelsi I., Abdellatif M., Abubaker A**. et al. From legacy to microservices: A type-based approach for microservices identification using machine learning and semantic analysis, *Journal of Software: Evolution and Process*, 2022. DOI: 10.1002/smr.2503.

12. **Jiao Q.-J.** Functional Units in Complex Networks: Beyond Cohesive Modules, *IFAC Proceedings Volumes*, 2013, no. 46, pp. 94—99. DOI: 10.3182/20130708-3-CN-2036.00040.

13. **Sharma S.** Web pattern analysis using partitioning algorithm in hyperlink structure, *International Journal of Emerging Trends in Engineering and Development*, 2020, no. 10. DOI: 10.26808/rs.ed.i10v2.03.

14. **Bagrudeen B. A., Yuvaraj Dr. D.** et al. An Efficient Mechanism for Deep Web Data Extraction Based on Tree-Structured Web Pattern Matching, *Wireless Communications and Mobile Computing*, 2022, pp. 1—10. DOI: 10.1155/2022/6335201.

15. **Yadav P.** A Novel Approach of Development of Web Pattern by Focusing on Web Structure Mining Techniques, *International Journal on Recent and Innovation Trends in Computing and Communication*, 2019, no. 7, pp. 01—04. DOI: 10.17762/ijritcc.v7i2.5223.

УДК 004.021
DOI: 10.17587/prin.14.493-501

Е. А. Басыня, канд. техн. наук, доц., вед. науч. сотр., eabasynya@mephi.ru,
Н. Карапетьянц, ассистент, nkarapetyants@mephi.ru,
М. Карапетьянц, инженер, mkarapetyants@mephi.ru,
Национальный исследовательский ядерный университет «МИФИ», Москва

Исследование существующих подходов к анализу транзакций в сети Bitcoin

Поступила в редакцию 14.06.2023
Принята к публикации 03.08.2023

Целью работы, результаты которой представлены в статье, является анализ существующих методов проверки транзакции сети Bitcoin. В рамках работы предложена архитектура системы, реализующая комплексный подход к процессу анализа транзакций. Приведено описание каждого из этапов этого процесса, а именно сбора, агрегации, обработки и анализа информации. Рассмотрен расширенный набор эмпирических правил (эвристик) анализа транзакций, которые используются в существующих методах кластеризации адресов. Результаты настоящей работы могут способствовать совершенствованию существующих методов проверки транзакции Bitcoin и разработке новых подходов, повышающих эффективность процесса идентификации средств, полученных незаконным путем, и их источников.

Ключевые слова: *блокчейн, Bitcoin, анализ транзакции, KYC, KYT, кластеризация, эвристика*

Для цитирования:

Басыня Е. А., **Карапетьянц Н.**, **Карапетьянц М.** Исследование существующих подходов к анализу транзакций в сети Bitcoin // Программная инженерия. 2023. Том. 14, № 10. С. 493—501. DOI: 10.17587/prin.14.493-501.

Введение

На настоящее время доля криптовалют в сфере финансовых операций перманентно возрастает: по сравнению с 2020 г. общий объем криптовалют в 2021 г. вырос с 4,3 до 16 трлн долл. США. При этом пик рыночной капитализации криптовалют приходится на 2021 г., что стало возможным благодаря высокой степени вовлеченности людей в область использования криптовалюты в качестве финансово-технического средства. Однако нельзя не отметить, что с каждым годом прослеживается тенденция к росту доли использования криптовалют в незаконной деятельности, в том числе в отмывании денежных средств, спонсировании терроризма, а также в использовании злоумышленниками программ-вымогателей и другого вредоносного программного обеспечения (ПО) (рис. 1, см. вторую сторону обложки) [1]. В качестве примера стоит привести следующие факты. Для отмывания денежных средств в 2020 г. использовались такие способы, как покупка и продажа невзаимозаменяемых токенов (NFT, *non-fungible token*), а также микширование украденных криптовалют через специализированные сервисы. Последнее подразумевает сбор поступающих криптомонет в «общем процессоре», в котором эти цифровые финансовые активы перемешиваются и затем рассылаются на различные кошельки.

Криптовалюта вызывает большую заинтересованность со стороны регуляторных органов на международной арене, что подтверждается большим числом публикаций на данную тематику [2, 3]. При этом формирование нормативно-правовых актов, связанных с криптовалютами, как в Российской Федерации (РФ), так и в международной сфере происходит с запаздыванием.

К настоящему времени не получается полностью сформировать унифицированный глобальный (трансграничный) подход по регулированию криптовалют. Несмотря на это, в РФ постепенно закладывают фундамент законодательной базы по

различным частям использования криптовалют. При этом основными органами исполнительной власти в данной области в РФ являются: Федеральная налоговая служба, Федеральная служба безопасности, Министерство внутренних дел, Росфинмониторинг и ряд других служб. Следует отметить, что особым конституционно-правовым статусом обладает Центральный банк Российской Федерации (ЦБ РФ): определено его исключительное право на осуществление денежной эмиссии и, в качестве основной функции, защиту и обеспечение устойчивости рубля. Соответственно, регулирование в сфере криптовалют не может остаться без внимания ЦБ РФ.

На международной арене страны по-разному подходят к регулированию криптовалют: большая часть развитых государств, например, придерживается признания криптовалюты как законного платежного средства и реализует механизмы соответствующего контроля за отмыванием денежных средств (рис. 2, см. вторую сторону обложки). Существуют и противники использования криптовалюты в качестве платежного средства. Так, в Китае, начиная с 2013 г., полностью были ограничены операции, связанные с Bitcoin. Помимо всего прочего, некоторые страны Африки, такие как Кения и Гана, не имеют нормативно-правовых баз по регулированию криптовалют, несмотря на то обстоятельство, что за последние несколько лет наблюдается большой рост проведения криптовалютных транзакций. Данный факт связан с ростом рынка прямой торговли без участия посредника (P2P, *peer-to-peer*) и использованием криптовалюты как финансового средства (далее — инструмента) взамен классической банковской системы, имеющей слабое развитие в текущем регионе.

Основополагающим документом по регулированию криптовалютной отрасли в РФ является Федеральный закон (ФЗ) № 259 от 31 июля 2020 г. «О цифровых финансовых активах, цифровой валюте и о внесении изменений в отдельные законодательные акты Российской Федерации», который запрещает использование криптовалюты в качестве средства платежа. По данному документу до сих пор ведутся споры по той причине, что он имеет достаточное количество противоречий и неясностей. Несмотря на это, ФЗ № 259 позволил заложить основы и провести четкое разделение между цифровыми финансовыми активами и цифровыми валютами.

В 2022 г. было проведено внедрение цифрового сервиса со стороны МВД РФ, который называется «Личный кабинет правоохранительного органа». Данный инструмент направлен на выявление недобросовестных владельцев криптокошельков. Также Росфинмониторинг отслеживает криптовалютные транзакции и выявляет преступную деятельность с помощью сервиса «Прозрачный блокчейн».

Процесс идентификации сущностей и средств в сети Bitcoin осуществляется каждым вендором (например, Chainalysis, Crystal Blockchain, CipherTrace и т. д.) по-разному. На настоящий момент особенности работы таких алгоритмов не раскрываются в силу коммерческой тайны. При этом следует отметить, что работа большинства базовых алгоритмов осуществляется путем поиска до первого известного субъекта сети (ранее владевшего средствами) исходя из данных цепочки транзакций. Основным недостатком такого способа является возможность легализации источника средств посредством их перевода через доверенные источники, например, биржи. Предполагается, что в ряде стран, где осуществляется регуляторная деятельность криптовалют, биржи обязаны проводить процедуры «Знай своего клиента» (KYC, *know your client*), а в некоторых случаях и процедуру «Знай свою транзакцию» (KYT, *know your transaction*). Однако данный подход не гарантирует точных результатов в режиме реального времени, а функционирует с запаздыванием. В качестве примера можно привести ситуацию, когда перевод средств был одобрен биржей, а информация о том, что источник средств связан с незаконной деятельностью, поступила позже.

Для решения описанных выше задач научным сообществом разрабатываются алгоритмы, методы и методики, которые можно условно разбить на перечисленные далее два направления.

- Извлечение, преобразование, загрузка (ETL, *extract—transform—load*), представляют собой разработку метода, который подразумевает извлечение, преобразование, загрузку данных из сети Bitcoin и дальнейший анализ адресов пользователей сети. В данной области проводят исследования следующие ученые: А. Л. Сердечный, Д. А. Безуглов, П. В. Разумов, А. А. Зеленский, Л. В. Черкесова, Zhang Wenyin, N. L. Virginia Franqueira, Lee Youngseok. К недостаткам отмеченных выше исследований можно отнести слабую масштабируемость предлагаемых решений. При увеличении числа транзакции растет и время, затрачиваемое на обработку [2—6].
- Классификация субъектов, кластеризация адресов и выявление аномальных транзакции в сети Bitcoin. Данный подход подразумевает использование машинного обучения для деанонимизации экосистемы Bitcoin в целях выявления

контрагентов (сущностей), связанных с незаконной деятельностью. В данной области проводят научно-практические изыскания следующие ученые: Л. П. Бакуменко, Е. А. Фельдман, А. Н. Ручай, Zhu Liehuang, Du Xiaojiang. Недостатком предлагаемых решений являются ограничения на начальные условия и набор данных. Точность построенных моделей, в первую очередь, зависит от качества набора данных [7—9].

Таким образом, для решения задач в рамках рассматриваемой проблематики возрастает актуальность разработки методов и систем проверки транзакции в сети Bitcoin.

Постановка задачи

Целью исследования, результаты которого представлены в настоящей статье, являлось выполнение анализа существующих подходов к проверке транзакции в сети Bitcoin для выявления особенностей на этапах сбора, обработки и анализа информации. Для достижения поставленной цели были определены следующие задачи:

- исследование алгоритмов и методов анализа транзакций в сети Bitcoin;
- исследование инструментов и средств анализа транзакций в сети Bitcoin;
- описание эмпирических правил (эвристик) анализа транзакций;
- систематизация проблематики предметной области.

Далее будет последовательно рассмотрена каждая задача и предложены подходы к их решению.

Исследование предметной области

Процесс анализа транзакций состоит из трех этапов, включающих сбор, обработку и анализ информации об осуществляемых действиях в сети Bitcoin. Этап сбора информации включает в себя извлечение данных из разных источников. Анализ существующих публикаций показал, что получение данных происходит за счет использования методов сбора разного рода данных, которые можно получить следующими способами.

1. Bitcoin core — программная реализация для доступа к локальной сети блокчейна и синхронизации данных. Полученные данные связаны непосредственно с информацией о транзакции, хеше, блоках, что позволяет представить полную картину сети Bitcoin [6].

2. Поставщик обработанных данных: способ подразумевает сбор информации о бирже и ее непосредственных участниках, которая позволяет повысить точность идентификации средств, полученных незаконным путем, и их источников [4].

3. Поисковые роботы. Этот способ основан на сборе данных вне цепочки посредством использования поискового робота (веб-краулер). Способ предоставляет возможность получить адреса пользователей, опубликованных на сторонних веб-страницах и форумах [5].

На рис. 3 представлен фрагмент UML-диаграммы компонента системы анализа транзакции сети Bitcoin, которая описывает сбор информации. Информация поступает из трех источников: из сети Bitcoin, с общедоступных сайтов сети Интернет, а также с ресурсов поставщика обработанных данных. Извлечение данных из сети Bitcoin происходит с помощью пользовательского клиента Bitcoin Core через удаленный вызов процедур (RPC). Данный способ позволяет получить исчерпывающую информацию о транзакциях в сети Bitcoin, за исключением информации, позволяющей идентифицировать участников сети. Информация для идентификации участников собирается с помощью поискового робота или предо-

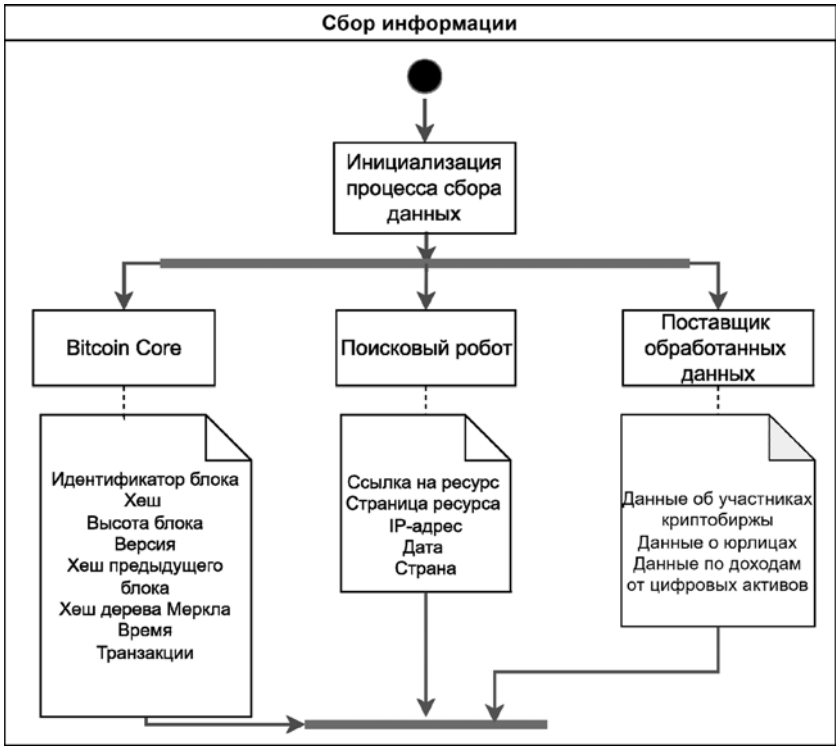


Рис. 3. UML-диаграмма компонента «Сбор информации» системы анализа транзакции сети Bitcoin

ставляется поставщиком обработанных данных. В первом случае сбор информации осуществляется в автоматизированном режиме из открытых источников общедоступной сети Интернет с помощью робота. Информация, найденная таким образом, нуждается в проверке, поскольку может не соответствовать действительности. К надежным источникам информации можно отнести внешнего поставщика, который предоставляет данные в структурированном виде.

Полученная из открытых источников информация должна храниться непосредственно в локальном доступе для ее дальнейшего анализа, поэтому для хранения частично структурированных данных предусмотрена отдельная база данных. В настоящее время для хранения и обработки данных [6], как правило, используются реляционные системы управления базами данных (СУБД). Отметим, что в перспективе это обстоятельство станет значимым проблемным вопросом в связи с ростом количества информации в глобальной вычислительной сети Интернет. Начинает появляться необходимость в использовании более продвинутых технологий.

В отличие от существующих методов анализа больших данных, анализ данных блокчейна сталкивается со многими вопросами. Данные блокчейна хранятся в клиенте с использованием разнородной и достаточно сложной структуры, которую нельзя проанализировать без предварительной обработки.

Процедура проведения обработки заключается в реализации трех основных операций: преобразования данных цепочки блоков транзакций, парсинга и индексации страниц ресурса, а также преобразования сериализованных данных, связанных с информацией о контрагентах сети Bitcoin (рис. 4).

Проведенные операции позволяют получить информацию о транзакциях и блоках, которые также необходимо хранить в базе данных. Однако в этом случае исследователи сталкиваются с трудностями, обусловленными высокой избыточностью и согласованностью хранимых данных. Это обстоятельство может негативно сказаться на реализации последующих этапов метода.

Анализ обработанных данных подразумевает реализацию нескольких компонентов поочеред-

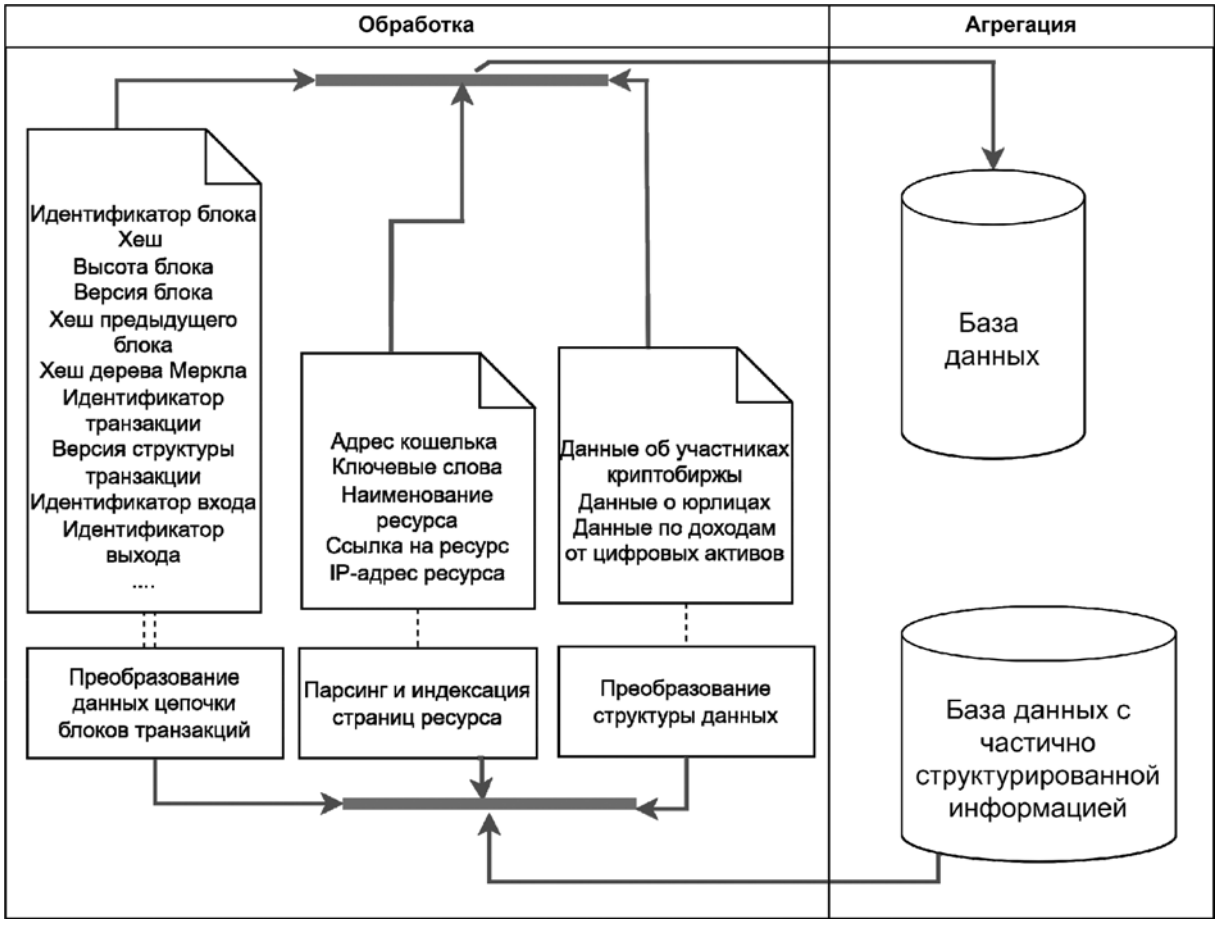


Рис. 4. UML-диаграмма компонентов «Обработка данных» и «Агрегация данных» системы анализа транзакции сети Bitcoin

ных процедур, которые включают в себя оценку риска посредством базового распространения меток. Полученные данные о риске каждого контрагента или транзакции необходимо отслеживать, подразделяя их на признаки отклонения нормы, подразумевающей отсутствие взаимосвязи с мошенническими схемами. Возможность выполнять группировку и фильтрацию на основе различных атрибутов и визуализировать результаты на временной шкале позволяет качественно подходить к анализу данных. Подозрительные транзакции на основе проанализированных данных могут получить соответствующую метку. Отметим, что при формировании новых данных не учитываются решения пользователей существующих систем анализа транзакций в сети Bitcoin.

Существующие системы анализа транзакций (рис. 5) предоставляют следующие инструменты: обозреватель блокчейн, граф и диаграммы транзакций, отслеживание транзакций, мониторинг событий. Обозреватель блокчейн предоставляет информацию о субъекте в сети, принадлежащих ему адресах и транзакциях.

Граф и диаграммы транзакций позволяют отследить передвижение средств в сети Bitcoin в соответствии с цепочкой транзакции. Инструмент для отслеживания транзакций представляет собой сервис для автоматизированного мониторинга адресов, которым принадлежат непотраченные Bitcoin-монеты. Инструмент мониторинга предоставляет актуальную информацию об имеющихся средствах, в том числе и показатель риска. Решение об одобрении или отклонении транзакции пользователь принимает самостоятельно, исходя из опыта и информации, предоставляемой системой.

Описание эвристик

Определение принадлежности адресов конкретному владельцу является важной задачей в сети Bitcoin. Она усложняется тем фактом, что существуют различные способы сокрытия владельца, например, использование алгоритмов маскировки транзакций или замены адресов во время транзакций. Для решения этой задачи могут быть использованы различные эвристики, каждая из которых обладает своими уникальными особенностями и применяется для нахождения определенных типов связей между транзакциями и адресами.

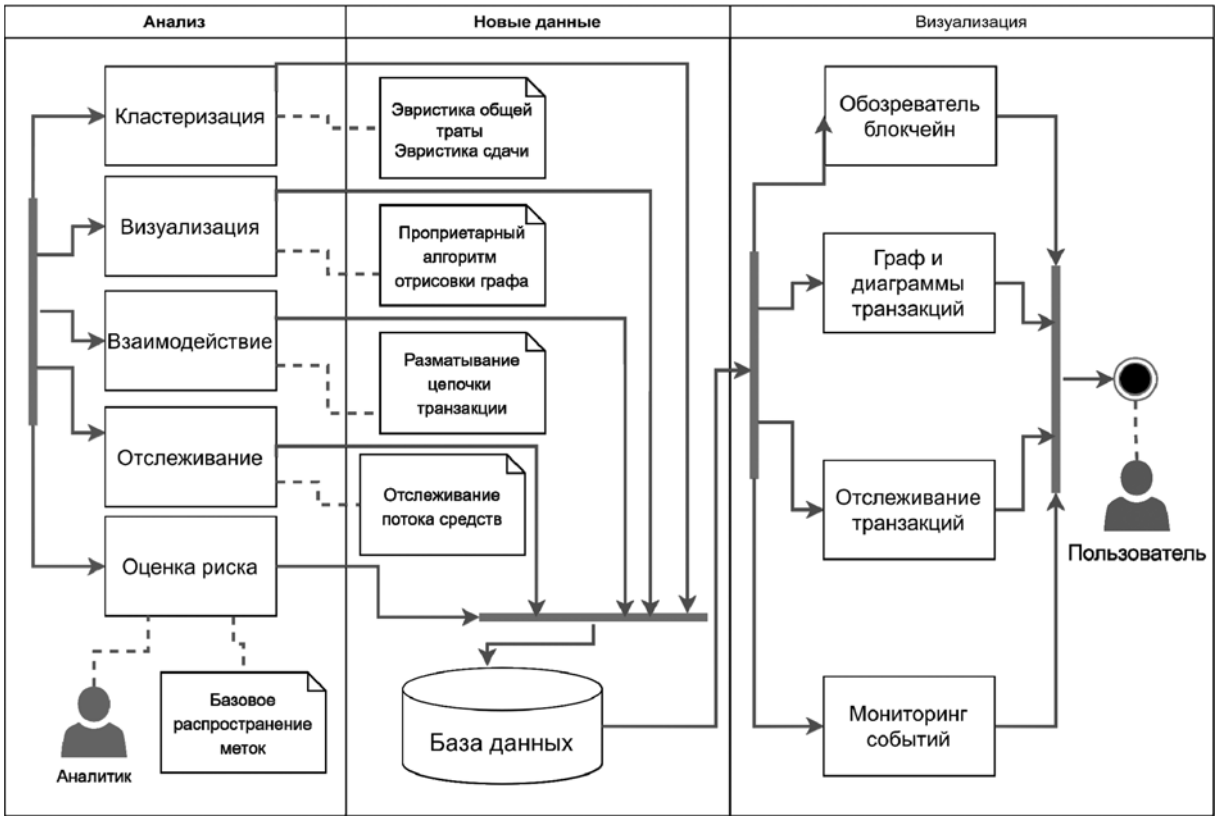


Рис. 5. UML-диаграмма компонентов «Анализ», «Новые данные» и «Визуализация» системы анализа транзакции сети Bitcoin

Можно выделить следующие эвристики, используемые для решения задач кластеризации адресов:

- эвристика общей траты;
- эвристика сдачи (эвристики смены адреса);
- эвристика Coinbase (майнинг) транзакции;
- эвристики майнингового пула;
- эвристика запутанных транзакций;
- эвристика зеленого адреса;
- эвристика многократного использования адреса;
- эвристика консолидации;
- эвристика пакетного расходования;
- чистая транзакция;
- метатранзакция;
- эвристика круглой суммы траты;
- эвристика оптимальной траты;
- эвристика времени блокировки;
- эвристика типа адреса.

Эвристика общей траты основывается на одном из базовых принципов сети Bitcoin: все входы одной транзакции принадлежат одному владельцу [10]. При этом, если хотя бы один адрес входа из ранних транзакций встретится с последующими, значит, и все сопутствующие адреса входов данной транзакции принадлежат одному владельцу, за исключением случаев, когда были использованы алгоритмы маскировки транзакции или сторонние маскировочные сервисы.

Эмпирическое правило сдачи или эвристика смены адреса позволяет определить принадлежность адресов на основе адреса сдачи. Такой адрес формируется всеми актуальными версиями клиентов сети автоматически в случае, когда суммарное количество имеющихся в наличии средств превышает сумму траты. Клиентом автоматически создается новый адрес выхода, который содержит разницу между суммой имеющихся средств и суммой траты.

Совместно с эвристикой сдачи может применяться и эвристика оптимальной траты [11]. Предполагается, что набор входов транзакции оптимальный, а значит, сумма выхода сдачи будет минимальной среди сумм всех выходов.

Определить адрес сдачи возможно по формату адреса выхода транзакции. Форматы адресов выходов могут не совпадать [12]. В таком случае, если один из адресов выхода транзакции имеет такой же формат, как и адрес входа транзакции, можно сделать вывод, что они все принадлежат одному владельцу.

Обнаружить субъекты, которые осуществляют добычу монет Bitcoin, можно с помощью эвристики Coinbase [13]. Адреса выходов транзакций Coinbase принадлежат одному лицу. Исключением может быть ситуация, если лицо является владельцем пула транзакции, который осуществляет распределение между участниками пула в соответствии с работой каждого майнера.

Распределение вознаграждения между участниками пула осуществляется за одну транзакцию. Поэтому, если в транзакции более 100 выходов и известно, что один из адресов принадлежит майнинговому пулу, это означает, что остальные выходы также принадлежат данному пулу.

Многие участники сети Bitcoin для обеспечения анонимности источника средств используют сторонние сервисы для смешивания транзакций, алгоритмы CoinJoin и Payjoin. Основным признаком таких транзакций являются одинаковые или почти одинаковые суммы переводов между участниками сети и сервисом смешивания.

Эвристика зеленого адреса позволяет выявить транзакции, с помощью которых осуществляются переводы средств на сторонний адрес через специализированный сервис [14]. Признаком таких транзакций можно считать адрес выхода, который является общедоступным и принадлежит данному сервису.

В процессе создания новых транзакций в сети Bitcoin используются непотраченные выходы транзакций (UTXO, *Unspent Transaction Output*). Количество непотраченных выходов транзакций, связанных с адресом получателя в сети, может быть большим, что напрямую влияет на размер комиссии. Для уменьшения комиссии за транзакцию используется консолидация всех потраченных выходов. Консолидационные транзакции объединяют несколько UTXO в один и переводят его на новый адрес.

Механизм пакетного расходования предназначен для уменьшения комиссии и чаще всего используется биржами. Суть его заключается в проведении как можно большего числа платежей посредством одной транзакции.

В сети Bitcoin преобладают чистые транзакции (*peel transaction*). Их основными признаками являются два выхода: выход траты и выход сдачи. При этом число входов таких транзакции может быть любым. Также в данных случаях применяется дополнительное эмпирическое правило: эвристика круглой суммы. Сформировать пользователю сумму выхода со сдачей, не имеющей десятичную дробную часть, затруднительно. Поэтому суть данной эвристики заключается в том, что выход, представленный в виде целого числа, является платежом, а другой — выходом сдачи.

Сокрыть источник средств позволяет механизм рекурсивной очистки транзакции [15]. Сокрытие обеспечивается за счет создания множества свя-

занных транзакций, где сумма выхода траты существенно меньше суммы выхода сдачи. Таким образом, выход сдачи используется для вывода средств на другие адреса.

Кроме транзакций, которые свидетельствуют о факте передачи права владения средствами, существуют метатранзакции, содержащие любую другую информацию, в том числе и текст.

Владельца адреса также может выдать время добавления транзакции в цепочку блоков. Адреса в двух других транзакциях одного блока могут принадлежать одному владельцу, если совпадают адреса выходов и время блокировки.

Параметры сети Bitcoin совместно с внешними данными и данными эвристики представляют собой большие данные. В таком случае необходимы алгоритмы кластеризации адресов, скорость работы которых позволит выполнить расчеты в рамках рационального времени. Для использования более точных алгоритмов необходимо уменьшить размерность в целях увеличения скорости работы. К таким алгоритмам относятся Лувенский метод [16] и алгоритм распространения меток [17].

В задаче кластеризации адресов в сети Bitcoin вкупе с внешними данными эвристиками используются алгоритмы с высокой скоростью работы. Связанно это с объемом обрабатываемых данных. По сравнению с традиционными методами кластеризации адресов время работы алгоритмов с оптимизацией быстродействия на несколько порядков меньше, что позволяет осуществить повторный перерасчет с учетом новой информации.

Лувенский метод обнаружения сообществ представляет собой эвристический метод, основанный на модульной оптимизации. По скорости работы он превосходит большинство существующих алгоритмов поиска сообществ и позволяет обработать сеть, состоящую из 2 млн вершин, за 2 мин [16].

В исследовании [17] был предложен алгоритм распространения меток, который до 700 раз опережает исходный по скорости работы как на эталонных, так и эмпирических сетях. Без учета эвристик данный алгоритм обрабатывает сеть, состоящую из более чем 6 млн нод, за 37 с, когда оригинальный алгоритм выводит результат после 45 мин работы.

Исходя из общей тенденции проводимых исследований в области анализа транзакций в сети Bitcoin, возникает необходимость в разработке метода, который будет использовать эмпирические правила, внешние данные и алгоритм поиска сообществ.

Заключение

В ходе научно-практических изысканий были проведены системные исследования анализа транзакций в сети Bitcoin, исследованы соответствующие алгоритмы, методы и методики. В результате такого исследования было выявлено два ключевых направления анализа транзакций: ETL и машинное обучение. Первое направление решает задачу эффективной обработки больших данных, а второе — определения владельцев в сети. Существующие методы и механизмы обработки транзакций сети Bitcoin не позволяют обрабатывать данные в реальном времени. Методы анализа транзакций с использованием машинного обучения совместно с эвристическими правилами обладают хорошими показателями быстродействия. Однако следует отметить, что точность таких методов кластеризации адресов напрямую зависит от качества набора данных, на которых проводится обучение, а это порождает необходимость продолжения исследований.

В ходе работы были описаны существующие методы, которые включают в себя три аспекта: сбор, обработку и анализ информации об осуществляемых действиях в сети Bitcoin. Использование реляционных СУБД приводит к возникновению проблемных вопросов, связанных с низкой согласованностью и большой избыточностью данных, что негативно сказывается на последующих этапах метода. Точность определения владельцев адресов является важной задачей в сети Bitcoin и использование только традиционных алгоритмов кластеризации адресов приводит к снижению точности и скорости анализа транзакции и адресов. Отсутствие использования в системе интеллектуальных методов анализа данных приводит к снижению достоверности идентификации субъектов блокчейн Bitcoin.

В рамках дальнейших исследований планируется разработать метод проверки транзакции в сети Bitcoin, устраняющий выделенные недостатки существующих решений и повышающий эффективность процесса идентификации средств, полученных незаконным путем, и их источников. Результаты таких исследований авторы планируют представить в следующей публикации.

Данная работа выполнена при поддержке Министерства науки и высшего образования (проект государственного задания № FSWU-2023-0031).

Список литературы

1. **Chainalysis** — The 2022 Crypto Crime Report. Chainalysis. 2022. URL: https://blockbr.com.br/wp-content/uploads/2022/06/2022-crypto-crime-report.pdf (дата обращения 02.02.2023).

2. **Сердечный А. Л., Скогорева Д. А., Длинный Е. П.** и др. Картографическое исследование blockchain-транзакций и смарт-контрактов киберпреступников, атакующих автоматизированные информационные системы, и оценка ущербов от реализации их атак // Информация и безопасность. 2021. Том 24, № 4. С. 471—500. DOI: 10.36622/VSTU.2021.24.4.001.

3. **Родивилина В. А., Родивилин И. П., Коломинов В. В.** Проблемы противодействия использованию анонимности в сети интернет в преступных целях // Криминалистика: вчера, сегодня, завтра. 2021. № 4. С. 68—79. DOI: 10.24412/2587-9820-2021-4-68-76.

4. **Balaskas A., Franqueira V. N. L.** Analytical tools for blockchain: Review, taxonomy and open challenges // 2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security). IEEE, 2018. P. 1—8. DOI: 10.1109/CyberSecPODS.2018.8560672.

5. **Song W., Zhang W., Wang J.** et al. Blockchain data analysis from the perspective of complex networks: Overview // Tsinghua Science and Technology. 2022. Vol. 28, No. 1. P. 176—206. DOI: 10.26599/TST.2021.9010080.

6. **Mun H., Kim S., Lee Y.** A RDBMS-based Bitcoin analysis method // International Conference on Information Security and Cryptology. Cham: Springer International Publishing, 2020. P. 235—253. DOI: 10.1007/978-3-030-68890-5_13.

7. **Бакуменко Л. П., Васильева Н. С.** Классификация методом опорных векторов мошеннических программ кражи биткоина // Учет и статистика. 2022. Том 68, № 4. С. 112—122.

8. **Фельдман Е. В., Ручай А. Н., Матвеева В. К., Самсонова В. Д.** Модель выявления аномальных транзакций биткоинов на основе машинного обучения // Челябинский физико-математический журнал. 2021. Том 6, № 1. С. 119—132. DOI: 10.47475/2500-0101-2021-16110.

9. **Zheng B., Zhu L., Shen M.** Identifying the vulnerabilities of bitcoin anonymous mechanism based on address clustering // Science China Information Sciences. 2020. Vol. 63. P. 1—15. DOI: 10.1007/s11432-019-9900-9.

10. **He S., He K., Lin S., Yang C., Mao H.** Bitcoin address clustering method based on multiple heuristic conditions // IET Blockchain. 2022. Vol. 2, No. 2. P. 44—56. DOI: 10.1049/blc2.12014.

11. **Long T., Xu J., Fu L., Wang X.** Analyzing and de-anonymizing Bitcoin networks: An IP matching method with clustering and heuristics // China Communications. 2022. Vol. 19, No. 6. P. 263—278. DOI: 10.23919/JCC.2022.06.019.

12. **Möser M., Narayanan A.** Resurrecting address clustering in Bitcoin // International Conference on Financial Cryptography and Data Security. Cham: Springer International Publishing, 2022. Vol. 13411. P. 386—403. DOI: 10.1007/978-3-031-18283-9_19.

13. **Zhao Z., Wang J., Shi Q., Zhang H.** Improving Address Clustering in Bitcoin by Proposing Heuristics // IEEE Transactions on Network and Service Management. 2022. Vol. 19, No. 4. P. 3737—3749. DOI: 10.1109/TNSM.2022.3186466.

14. **Chang T. H., Svetinovich D.** Improving bitcoin ownership identification using transaction patterns analysis // IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2018. Vol. 50, No. 1. P. 9—20.
DOI: 10.1109/TSMC.2018.2867497.

15. **Fisher J. A., Palechor A., Dell'Aglio D., Bernstein A., Tesson C. J.** The complex community structure of the bitcoin address correspondence network // Frontiers in Physics. 2021. Vol. 9. P. 681798. DOI: 10.3389/fphy.2021.681798.

16. **Traag V. A.** Faster unfolding of communities: Speeding up the Louvain algorithm // Physical Review E. 2015. Vol. 92, No. 3. P. 032801. DOI: 10.1103/PhysRevE.92.032801.

17. **Traag V. A., Shubel L.** Large network community detection by fast label propagation // Scientific Reports. 2023. Vol. 13, No. 1. P. 2701. DOI: 10.1038/s41598-023-29610-z.

A Study of Existing Approaches to Transaction Analysis in the Bitcoin Network

E. A. Basinya, Associate Professor, Leading Researcher, eabsynya@mephi.ru,
N. Karapetyants, Assistant of Department, nkarapetyants@mephi.ru,
M. Karapetyants, IICS Engineer, mkarapetyants@mephi.ru,
National Research Nuclear University "MEPhI", Moscow, 115409, Russian Federation

Corresponding author:

Evgeniy A. Basynya, Associate Professor, Leading Researcher,
National Research Nuclear University "MEPhI", Moscow, 115409, Russian Federation
E-mail: eabsynya@mephi.ru

Received on June 14, 2023
Accepted on August 03, 2023

Today, the Bitcoin network faces a number of challenges, such as flawed user identification and fraudulent transaction methods that are used by criminals to conduct illegal activities. As a result, there is a growing need to improve existing tools for tracking transactions, as well as to develop new methods for identifying money in the Bitcoin network. The paper presents a study and systematization of the subject area problems, and also considers possible approaches to neutralize it. The purpose of this work is to analyze the existing methods of transaction verification of the Bitcoin network. Within the framework of the work, a system architecture is proposed, which includes a comprehensive approach to the process of transaction analysis. Each of the stages of this process is described: information gathering, aggregation, processing and analysis. An extended set of empirical rules (heuristics) for transaction analysis, which are used in existing clustering methods, is considered. The results of this work will provide an opportunity to improve the existing Bitcoin transaction verification methods and develop a new one with the possibility of increasing the efficiency of the process of identification of illegally obtained funds and their sources.

Keywords: *blockchain, Bitcoin, KYC, KYT, transaction analysis, clusterization, heuristic*

Acknowledgements: *This work was supported by the Ministry of Science and Higher Education of the Russian Federation (state task project No. FSWU-2023-0031).*

For citation:

Basynya E. A., **Karapetyants N., Karapetyants M.** A Study of Existing Approaches to Transaction Analysis in the Bitcoin Network, *Programmnaya Ingeneria,* 2023, vol. 14, no. 10, pp. 493—501. DOI: 10.17587/prin.14.492-501.

References

1. **Chainalysis —** The 2022 Crypto Crime Report, Chainalysis, accessed 2 February 2023, available at: https://blockbr.com.br/wp-content/uploads/2022/06/2022-crypto-crime-report.pdf.

2. **Serdechny A. L., Skogoreva D. A., Longny E. P.** et al. Cartographic study of blockchain transactions and smart contracts of cybercriminals attacking automated information systems and assessment of damages from the implementation of their attacks, *Informacija i bezopasnost'*, 2021, vol. 24, no. 4, pp. 471—500. DOI: 10.36622/VSTU.2021.24.4.001 (in Russian).

3. **Rodivilina V. A., Rodivilin I. P., Kolominov V. V.** Problems of countering the use of anonymity on the Internet for criminal purposes, *Kriminalistika: vchera, segodnja, zavtra,* 2021, no. 4, pp. 68—79. DOI: 10.24412/2587-9820-2021-4-68-76 (in Russian).

4. **Balaskas A., Frankeira V. N. L.** Analytical tools for blockchain: Review, taxonomy and open challenges, *2018 International conference on cybersecurity and protection of digital services (cybersecurity), IEEE*, 2018, pp. 1—8. DOI: 10.1109/CyberSecPODS.2018.8560672.

5. **Song W., Zhang W., Wang J.** et al. Blockchain data analysis from the perspective of complex networks: Overview, *Tsinghua Science and Technology*, 2022, vol. 28, no. 1, pp. 176—206. DOI: 10.26599/TST.2021.9010080.

6. **Moon H., Kim S., Li Y.** A RDBMS-based Bitcoin analysis method, *Information Security and Cryptology — ICISC,* 2020, vol. 12593, pp. 235—253. DOI: 10.1007/978-3-030-68890-5_13.

7. **Bakumenko L. P., Vasilyeva N. S.** Classification by the support vector method of bitcoin theft fraud programs, *Uchet i statistika*, 2022, vol. 68, no. 4, pp. 112—122 (in Russian).

8. **Feldman E. V., Ruchai A. N., Matveeva V. K., Samsonova V. D.** A model for detecting anomalous bitcoin transactions based on machine learning, *Cheljabinskij fiziko-matematicheskij zhurnal*, 2021, vol. 6, no. 1, pp. 119—132. DOI: 10.47475/2500-0101-2021-16110 (in Russian).

9. **Zheng B., Zhu L., Shen M.** Identifying the vulnerabilities of bitcoin anonymous mechanism based on address clustering, *Science China Information Sciences,* 2020, vol. 63, pp. 1—15. DOI: 10.1007/s11432-019-9900-9.

10. **He S., He K., Lin S., Yang C., Mao H.** Bitcoin address clustering method based on multiple heuristic condition, *IET Blockchain*, 2022, vol. 2, no. 2, pp. 44—56. DOI: 10.1049/blc2.12014.

11. **Long T., Xu J., Fu L., Wang X.** Analyzing and de-anonymizing Bitcoin networks: An IP matching method with clustering and heuristics // China Communications, 2022, vol. 19, no. 6, pp. 263—278. DOI: 10.23919/JCC.2022.06.019.

12. **Möser M., Narayanan A.** Resurrecting address clustering in Bitcoin, *Financial Cryptography and Data Security: 26th International Conference*, 2022, vol. 13411, pp. 386—403. DOI: 10.1007/978-3-031-18283-9_19

13. **Zhao Z., Wang J., Shi Q., Zhang H.** Improving Bitcoin Address Clustering by Proposing Heuristics, *IEEE Transactions on Network and Service Management,* 2022, vol. 19, no. 4, pp. 3737—3749. DOI: 10.1109/TNSM.2022.3186466.

14. **Chang T. H., Svetinovich D.** Improving bitcoin ownership identification using transaction patterns analysis, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018, vol. 50, no. 1, pp. 9—20. DOI: 10.1109/TSMC.2018.2867497.

15. **Fisher J. A., Palechor A., Dell'Aglio D., Bernstein A., Tesson C. J.** The complex community structure of the bitcoin address correspondence network, *Frontiers in Physics,* 2021, vol. 9, pp. 1—16. DOI: 10.3389/fphy.2021.681798.

16. **Traag V. A.** Faster Community Deployment: Accelerating the Louvain Algorithm, *Physical Review E,* 2015, vol. 92, no. 3, pp. 032801. DOI: 10.1103/PhysRevE.92.032801.

17. **Traag V. A., Shubel L.** Detection of a large network community by fast label propagation, *Scientific report,* 2023, vol. 13, pp. 2701. DOI: 10.1038/s41598-023-29610-z

ИНФОРМАЦИЯ

УДК 004.5
DOI: 10.17587/prin.14.502-512

Д. К. Левоневский, канд. техн. наук, ст. науч. сотр., levonevskij.d@iias.spb.su,
А. И. Мотиенко, канд. техн. наук, ст. науч. сотр., anna.gunchenko@gmail.com,
Санкт-Петербургский Федеральный исследовательский центр
Российской академии наук (СПб ФИЦ РАН)

Моделирование и автоматизация процессов сбора и обработки данных в умной медицинской палате

Поступила в редакцию 07.07.2023
Принята к публикации 11.08.2023

В процессе исследования, результаты которого представлены в статье, были рассмотрены существующие решения в области умных медицинских палат. Такие решения позволяют обеспечить повышенную безопасность пациентов, улучшить качество их лечения и повысить комфорт, оптимизировать рабочие процессы в стационарном учреждении и снизить затраты. Вместе с тем указанные решения являются частными, используют различные технологии и протоколы и реализуют отдельные сценарии функционирования палаты. Этого недостаточно для создания комплексного подхода к автоматизации ведения пациентов в умных палатах, так как возникают вопросы совместимости и интеграции разнородных модулей, систем и протоколов, что приводит к растущим затратам и снижению рентабельности. Чтобы ответить на эти вопросы, комплекс необходимо реализовывать с использованием единого фреймворка с типовыми компонентами, связями, сценариями функционирования. Для этого на первом этапе реализации проекта разрабатывается концептуальная модель умной палаты, приводится характеристика ее компонентов, а также сценариев ее функционирования. Разработан алгоритм сбора и обработки данных для диагностики и стратификации ряда заболеваний, проведено моделирование процессов функционирования умной палаты и показано, что реализация сбора и обработки данных с использованием умных компонентов позволяет сократить время процесса более чем в 2 раза.

Ключевые слова: *умная медицинская палата, медицинская киберфизическая система, автоматизация в медицине, умное пространство, человеко-машинное взаимодействие*

Для цитирования:

Левоневский Д. К., Мотиенко А. И. Моделирование и автоматизация процессов сбора и обработки данных в умной медицинской палате // Программная инженерия. 2023. Том 14, № 10. С. 502—512. DOI: 10.17587/prin.14.502-512.

Введение

Умные медицинские палаты (также умные палаты, далее — УП), являясь по сути медицинскими киберфизическими системами, выглядят многообещающим решением для повышения качества оказания медицинской помощи в больницах, клиниках, интернатах, в домах престарелых и иных стационарных учреждениях. Они предназначены для интеграции передовых технологий, сбора и анализа данных и автоматизации процессов ухода за пациентами, повышения их безопасности и оптимизации рабочего процесса. Умные палаты включают ряд инновационных инструментальных средств. К их числу относятся: носимые устройства, датчики, электронные медицинские карты и системы мониторинга в режиме реального времени. Все это необходимо, чтобы предоставить медицинским специалистам исчерпывающую и актуальную информацию о состоянии пациентов. Такой подход позволяет персоналу быстрее и более информировано принимать решения [1].

В исследовании, результаты которого представлены в статье, разработана концептуальная модель УП, включающая спецификации ситуаций, поведения и компонентов киберфизического окру-

жения палаты. Создание такой модели является первым шагом к созданию УП. Модель описывает основные понятия предметной области и связи между ними. На этом этапе важно выделить и охарактеризовать основные типы сущностей, которыми будем оперировать, определить отношения между ними. Такой подход позволит в дальнейшем строить частные модели, описывающие решение тех или иных конкретных задач. В статье будет рассмотрен типовой сценарий сбора и обработки данных в УП, выполнено его моделирование и приведена оценка затрат времени для сбора информации о пациенте и оценки его состояния.

Существующие решения в области медицинских киберфизических систем

Рассмотрим существующие подходы, методы, модели и системы, используемые для построения УП.

В работе [2] авторы предлагают систему голосового управления для взаимодействия с оборудованием в палате, что позволяет улучшить качество и удобство медицинского обслуживания [3—5]. Система голосового управления подходит для всех госпитализированных и особенно полезна для послеоперационных пациентов и инвалидов [6]. Пациенты могут лежать на кровати и, произнося команды мобильным устройствам, управлять оборудованием в палате.

В работе [7] авторы предлагают бесконтактную систему телемедицины для эффективного мониторинга удаленных карантинных отделений COVID-19 в Индии с использованием ближней связи и системы обработки естественного языка (*Natural Language Processing,* NLP). Авторы предлагают систему, которая объединяет чип связи ближнего радиуса действия (*Near Field Communication,* NFC) и интеллектуальное облачное средство аналитики для обеспечения бесконтактной (или с минимальным контактом) связи с системой. Чип NFC позволяет врачу легко извлекать историю болезни пациента из облачной базы данных. Каждому пациенту выдается отдельный NFC-чип, который служит связующим звеном с базой данных, что минимизирует непосредственное взаимодействие медицинского работника с ним [8—10]. Благодаря этому снижается вероятность ошибочного диагноза и обеспечивается защита врача от контакта с инфицированным пациентом.

Средства интеллектуального структурирования данных на основе облачных технологий помогают медицинским специалистам с меньшими усилиями систематизировать данные и интегрировать их в существующие базы пациентов. Эти средства используют методы data mining и NLP для представления повествовательных клинических текстов в структурированном формате. Они имеют встроенное приложение оптического распознавания символов (*Optical Character Recognition,* OCR), которое помогает преобразовывать текст на изображениях (медицинские отчеты) в цифровой формат [11]. Традиционные медицинские карты доступны в виде печатных копий, что затрудняет интеграцию содержащихся в них данных с базой данных. Таким образом, система, объединяющая технологии NFC, NLP и OCR, благодаря простоте визуализации медицинских отчетов расширяет возможности принятия решений врачом. Она помогает отслеживать прогресс лечения пациентов на карантине и соответствующим образом назначать план лечения, поддерживая минимальное взаимодействие с людьми.

В работе [12] авторы предлагают инновационную интеллектуальную систему ухода за пациентами, основная функция которой — уведомление медсестер о том, что пациент нуждается в их помощи. Установлено, что среди всех возможных причин тревоги, таких как потребность в информации, обезболивание и помощь в туалете, менее одной трети всех вызовов медсестры считают серьезными или срочными [13—15]. Учитывая, что персоналу приходится выполнять множество клинических и административных задач, им трудно принимать решения относительно того, следует ли прерывать текущую задачу, чтобы отреагировать на тревожный сигнал. Имея ограниченную информацию о тревоге, предоставляемую системами ухода, медсестры, как правило, снижают приоритетность при реагировании на нее [16]. Авторы этих публикаций констатируют необходимость создания устойчивой системы ухода за пациентами, которая поможет персоналу расставлять приоритеты в своих реакциях на сигналы тревоги и справляться с ними, одновременно выполняя другие обязанности.

В интеллектуальной системе ухода за пациентами все сигналы могут передаваться на мобильные телефоны медсестер, что обеспечивает немедленную связь и трехэтапное оповещение о покидании постели, поскольку падение пациентов также является распространенной ситуацией [17, 18]. В исследовании был использован матрас, чувствительный к движению и позволяющий осуществлять трехэтапное оповещение о выходе из постели, чтобы медсестры получали предупреждение о намерении пациента покинуть свою постель сразу после того, как он в ней сядет. Кроме того,

персонал может попросить пациентов подождать помощи при вставании с постели с помощью функции мгновенной связи при невозможности помочь немедленно.

В существующих системах при вставании с постели часто сообщалось о ложных срабатываниях. Их чрезмерное число может привести к усталости от оповещений, при которой персонал иногда игнорирует сигналы тревоги. В данном исследовании авторы наблюдали 35,2 % ложных срабатываний в традиционной системе ухода за пациентами и 0 % в интеллектуальной.

В работе [19] предложен подход к организации умной палаты на основе нейрокомпьютерного интерфейса (НКИ) и методологии IoT (*Internet of Things,* Интернет вещей), использующий гибридные сигналы. Система содержит подсистемы гибридной асинхронной электроэнцефалографии (ЭЭГ), электроокулографии (ЭОГ) и управления НКИ на основе гироскопа, а также подсистему мониторинга и управления IoT. Она основана на способе управления графическим интерфейсом через НКИ. Графический интерфейс состоит из курсора и нескольких кнопок. Пользователь использует гироскоп для управления выделением области курсора и ЭОГ, связанную с морганием, для управления щелчком курсора. Он получает подсказки о движениях глаз (моргании) и одновременно записывает их. С помощью идентификации ЭЭГ- и ЭОГ-сигналов пользователя и выполняется синхронный выбор и передача управляющих команд. Добавление режима ЭЭГ позволяет эффективно сократить число ложных операций. Кроме того, носимые устройства и оборудование с камерами могут собирать физиологические сигналы и другие сигналы мониторинга для пациентов, чтобы осуществлять пассивный контроль посредством комплексной оценки.

В работе [20] предложено формирование верхнего уровня ИТ-архитектуры умной больницы и представление требований к ее комплексной архитектуре. Авторы этой публикации выделяют следующие ключевые компоненты, которые должна включать умная клиника: интеллектуальное здание; комплексная система автоматизации и диспетчеризации инженерных систем здания; комплексная система безопасности здания; мультимедийные системы; непосредственно УП. Построение архитектурной модели следует принципам постепенной детализации, согласованности уровней, независимости слоев, полноты, последовательности, отсутствия дублирования и непрерывной трансформации текущей архитектуры предприятия.

В работе [21] авторы предлагают систему контроля уровня жидкости во флаконах для внутривенных капельниц, которые могут использоваться в отделениях интенсивной терапии (ОИТ) и послеоперационных отделениях. В разработке использовалась система обмена сообщениями на базе GSM для оповещения персонала о том, что бутылка скоро опустеет. Используется тензодатчик для измерения веса жидкости, а микроконтроллер считывает данные и отправляет их в GSM-модуль [22]. Уровень жидкости в бутылке в режиме реального времени отображается в приложении на Android. На дисплее отображается соответствующий процент жидкости, оставшейся во флаконе. Когда этот уровень ниже 100 мл, выводится предупреждение и повторно отправляется SMS-сообщение дежурному с периодом в 10 с до тех пор, пока система не выключится. Одновременно с предупреждением звучит звуковой сигнал. Кроме того, команда медицинских работников может контролировать уровень жидкости в режиме реального времени на своем рабочем месте с помощью модуля IoT. В случае, если персонал/сопровождающий не может добраться до пациента по тревоге, отток жидкости автоматически перекрывается электромагнитным клапаном.

В работе [23] авторы предложили систему, основанную на IoT, которая измеряет показатели жизнедеятельности как пациентов, стоящих в очереди в амбулаторных отделениях, так и пациентов, находящихся в палатах медицинских учреждений в Малави. Разрабатываемая система регистрирует частоту пульса и насыщение кислородом у амбулаторных и стационарных пациентов. Датчики, используемые для сбора двух жизненно важных показателей, подключены к компьютерам и мобильным устройствам, к которым имеет доступ медицинский персонал. Частота пульса дает представление о температуре тела и кровяном давлении [24]. Насыщение кислородом также дает представление о частоте дыхания. Любые изменения, выходящие за рамки нормы, могут быть обнаружены быстро, и это позволит медицинскому персоналу вовремя оказать помощь пациенту.

В работе [25] представлена система мониторинга шейного отдела в медицинских палатах на основе носимых устройств. На основе облачной платформы IoT реализуется интеллектуальное управление и контроль оборудования в палатах. В соответствии с конкретным сценарием использования УП применяемая система мониторинга и управления IoT может интегрировать различные компоненты. Технология и система облачной платформы IoT имеют масштабируемое пространство

хранения данных и эффективные возможности их обработки. В этой работе анализируется текущая ситуация и медицинские перспективы интеллектуальных носимых устройств для профилактики и лечения шейного спондилеза у офисных сотрудников. В системе IoT спроектированы узел мониторинга и узел контроллера. Каждый узел включает в себя основной модуль управления, сетевой модуль Wi-Fi, модуль питания и периферийных схем, модуль датчиков и модуль контроллера. Реализованы средства управления сенсорными узлами и аналитики, позволяющие отображать динамику процесса.

В работе [26] рассмотрена система локализации пациентов. Медицинскому персоналу необходимо осуществлять наблюдение за пациентом во время госпитализации и оказывать своевременную помощь. Однако часто сложно эффективно выполнять свои обязанности, особенно в условиях необходимости соблюдения конфиденциальности. Достижения в технологиях локализации без устройств и развитие технологий машинного обучения сделали локализацию более точной. Авторы работы [26] используют легкодоступные сигналы Wi-Fi в палатах и выполняют локализацию пациентов с сохранением конфиденциальности, используя многомасштабные сверточные нейронные сети и модели долгой краткосрочной памяти. Результаты демонстрируют высокую точность локализации. Кроме того, система может быть расширена для обнаружения чрезвычайных ситуаций, что позволяет медицинскому персоналу реагировать быстро.

В работе [27] рассмотрена контекстно-ориентированная система ухода, которая способна эффективно улучшить качество медицинской помощи в больницах. Авторы предлагают новую «ситуационно-осведомленную» систему ухода для УП, где «ситуация» относится к новому типу контекста, извлекаемому механизмами логического вывода, основанными на знаниях, с учетом базовой контекстной информации. Кроме того, предлагаемая система построена на основе сети беспроводных датчиков окружающей среды для окончательного восприятия контекста, что отличается от предыдущих проектов, использующих носимые датчики (например, радиочастотные) или датчики, нарушающие конфиденциальность (например, камеры) для извлечения контекстов. Архитектура разработана таким образом, чтобы приложения, созданные на основе системы, были расширяемыми. Кроме того, система способна поддерживать иерархический вывод ситуаций из простых контекстов.

На основе предложенной системы были созданы приложения для уведомления лиц, осуществляющих уход, о предварительно настроенном тревожном событии (например, падении), и для создания отчетности, которая дает интегрированное представление о состоянии пациента.

В работе [28] представлена интеллектуальная система мониторинга здоровья для определения состояния пациента. Такие системы требуют сбора, передачи и обработки больших объемов мультимодальных медицинских данных, генерируемых различными типами датчиков и медицинских устройств, что является сложной задачей и может сделать некоторые из приложений удаленного мониторинга здоровья непрактичными. Поэтому перенос вычислительного интеллекта на конечные устройства (*edge computing*) является многообещающим подходом для непрерывного удаленного мониторинга. Авторы представляют механизм классификации, который позволяет обнаруживать эпилептические припадки с высокой точностью и низкими требованиями к вычислительным ресурсам. Предлагается схема выборочной передачи данных и надежная энергоэффективная система экстренного оповещения для обнаружения эпилептических припадков. Например, при нормальных состояниях пациента можно сэкономить значительное количество энергии, передавая должным образом сжатые данные или отправляя только наиболее репрезентативные характеристики ЭЭГ, которые имеют отношение к выявлению судорог.

Представленный выше краткий обзор позволяет сделать вывод о том, что существует довольно много решений в области УП, предназначенных для автоматизации тех или иных задач в учреждении здравоохранения. При этом указанные решения являются частными, используют различные технологии и протоколы и реализуют отдельные сценарии работы УП. В случае, когда необходимо реализовать комплексный подход к автоматизации ведения пациентов в УП, применение этих решений оказывается недостаточным. Причина в том, что при этом возникают вопросы совместимости при интеграции разнородных модулей, систем и протоколов, а также вопросы дублирования компонентов, что приводит к растущим затратам и снижению рентабельности. Кроме того, повышаются и риски нарушения информационной безопасности, которые неизбежны при усложнении комплекса УП.

Чтобы избежать перечисленных выше вопросов, весь комплекс УП необходимо реализовывать с использованием средств единого фреймворка

с типовыми компонентами, связями и сценариями функционирования. Для этого на первом этапе проекта разрабатывается концептуальная модель УП, позволяющая в дальнейшем создать интегрированные программные и аппаратные решения либо интегрировать существующие компоненты, выполняя комплектацию УП исходя из необходимого для конкретного подразделения набора задач.

Концептуальная модель умной палаты

Для того чтобы в наиболее общем виде описать УП, сформируем набор базисных множеств, которые описывают предметную область:

$$\Omega = <G,\ T,\ A,\ M,\ S,\ C,\ I>,$$

где G — множество целей; T — множество задач; A — множество акторов (субъектов действия); M — множество модулей; S — множество сценариев функционирования; C — множество интерфейсов; I — множество показателей эффективности достижения целей.

Определим основные отношения между элементами перечисленных множеств:

R_{GT} — отношение, устанавливающее, какие задачи выполняются для достижения заданной цели;

R_{TS} — отношение, устанавливающее, какие сценарии реализуются при выполнении задач;

R_{AS}, R_{MS} — отношения, устанавливающие, какие акторы и модули задействованы в сценариях;

R_{CM} — отношение, устанавливающее, какие интерфейсы используются при передаче данных между модулями;

R_{IG} — отношение, устанавливающее, какие показатели отражают эффективность достижения целей.

Таким образом, УП как сложная система может быть представлена в виде (рис. 1):

$$S = <\Omega,\ R_{GT},\ R_{TS},\ R_{AS},\ R_{MS},\ R_{CM},\ R_{IG}>.$$

Рассмотрим более подробно каждое базисное множество.

Цели. Создание УП преследует, как правило, одну или несколько целей, которые можно разбить на перечисленные далее группы.

1. Цели, связанные непосредственно с медицинской деятельностью: обеспечить оперативный мониторинг состояния здоровья пациента и выявление ситуаций, требующих повышенного внимания или вмешательства; обеспечить частичную автоматизацию процесса диагностики заболеваний и терапии.

Рис. 1. Обобщенная концептуальная модель умной палаты

2. Косвенные цели: повысить комфорт пациентов; повысить производительность персонала путем автоматизации рутинных задач; упростить коммуникацию между персоналом и пациентами, особенно в случаях, когда привычные способы общения невозможны или затруднительны для пациентов.

Задачи. К типовым задачам относят: снятие показателей и команд с датчиков и мониторов, сохранение измеренных данных в хранилищах; анализ данных и выработку рекомендаций; выработку, ввод и реализацию управляющих воздействий; интеграцию данных и компонентов; оповещение участников процесса; обеспечение конфиденциальности данных.

Акторы. Умные палаты включают в себя множество акторов. К основным категориям относятся перечисленные далее.

1. Пациенты: люди, которые получают медицинское обслуживание в УП. Они могут использовать различные устройства, такие как мониторы, пульты управления и т. д., чтобы помочь врачам и медсестрам собирать данные о своем здоровье, управлять функциональностью УП.

2. Врачи: медицинские специалисты, ответственные за лечение пациентов. Они могут использовать данные, собранные с помощью умных устройств, чтобы определить оптимальную тактику лечения для каждого пациента.

3. Медицинские сестры: медицинский персонал, ответственный за уход за пациентами. Они могут использовать умные устройства для сбора данных о пациентах и контроля за процессом лечения.

4. Инженеры: специалисты по разработке и обслуживанию технического оборудования и программного обеспечения для УП.

5. Аналитики: специалисты по обработке и анализу медицинских данных в УП.

Модули. Модули УП классифицируются, прежде всего, по их задачам и функциональности.

1. Модули сбора данных собирают данные о пациентах, такие как жизненные показатели, сведения о приеме лекарств и т. д., и передают их на сервера для обработки. Примеры: датчики здоровья, умные часы, тонометры.

2. Модули обработки и анализа данных могут использоваться для рекомендации способа лечения, выявления рисков и прогнозирования заболеваний на основе собранных данных. Примеры: экспертные системы, системы визуализации.

3. Модули коммуникации обеспечивают связь между пациентами, персоналом и устройствами УП. Они могут использоваться для передачи сообщений, напоминаний о приеме лекарств, дистанционной консультации и т. д. Примеры: устройства видеосвязи, пульты управления.

4. Модули управления лечебным процессом используются для автоматизации процесса терапии. Пример: робот доставки лекарств.

5. Модули управления комфортом могут использоваться для управления освещением, системами отопления и кондиционирования воздуха и т. д. Пример: система проветривания.

6. Модули защиты обеспечивают безопасность пациентов и персонала в УП. Они могут включать в себя системы мониторинга безопасности, средства контроля доступа, устройства тревожной сигнализации и т. д. Примеры: системы видеонаблюдения в коридорах, датчики движения, электронные замки и др.

7. Модули администрирования отвечают за мониторинг и управление инфраструктурой УП в целом, ее конфигурирование.

Существуют и другие критерии классификации модулей. По типу реализуемых процессов модули можно разделить на физические (исполнительные устройства), информационные (вычислительные устройства) и киберфизические (модули, для которых характерна интеграция информационных ресурсов и физических процессов). По области задач модули могут быть медицинскими, обслуживающими, вспомогательными, по роли в области обработки данных — источниками (датчики), обработчиками, получателями (визуализаторы, актуаторы). По мобильности модули могут быть стационарные, переносные, самодвижущиеся, по автономности — управляемые, автономные, по критичности — некритичные, заменяемые, критичные.

Интерфейсы. В общем случае они описываются с помощью протоколов и технологий на разных уровнях модели взаимодействия открытых систем — ISO OSI. Существует ряд типовых технологий для соединения модулей УП, в том числе перечисленные далее.

1. Беспроводные сети: технологии Wi-Fi, Bluetooth, Zigbee и Z-Wave могут использоваться для связи между стационарными и мобильными устройствами в УП.

2. Кабельная связь (Ethernet, USB), а также шины данных (например, CAN) могут использоваться для организации скоростной и надежной связи между стационарными устройствами.

3. Интернет вещей (IoT): протоколы связи, такие как MQTT и CoAP, могут использоваться для соединения медицинских приборов и датчиков.

4. Мобильная связь: мобильные сети, такие как 3G, 4G и 5G, также могут использоваться для соединения устройств в УП и передачи данных, для отправки уведомлений.

Существуют интегральные стандарты передачи информации в медицинской области, к примеру, стандарт HL7. Выбор интерфейса зависит от конкретных требований УП и используемых устройств.

Сценарии. Сценарии функционирования УП представляют собой типовые последовательности и алгоритмы функционирования палаты, предназначенные для решения ее задач. Типовые сценарии связаны с мониторингом состояния пациентов (медицинская сигнализация), его анализом, организацией процесса лечения, они рассмотрены ниже.

Показатели. Наиболее распространенные показатели эффективности УП могут отражать перечисленное далее.

1. Качество медицинской помощи: сокращение времени нахождения пациента в больнице, улучшение точности диагностики, снижение летальности и количества осложнений после операций и т. д.

2. Комфорт пациентов: снижение числа жалоб, опросы пациентов и т. д.

3. Оптимизацию процессов медицинского ухода: улучшение доступности медицинской информации, сокращение времени на выполнение рутинных процедур и т. д.

4. Затраты на здравоохранение: снижение числа процедур и препаратов, снижение трудозатрат, уменьшение числа повторных визитов пациентов в больницу, ошибок при медицинском обслуживании и т. д.

Раскроем более подробно отдельные составляющие концептуальной модели. Для этого построим вспомогательные модели. Начнем с характеристики УП в статике. Для этого применимы модели, отражающую структуру системы, — ее компоненты и связи между ними.

Поскольку УП представляет собой киберфизическую систему, в ней происходят как физические,

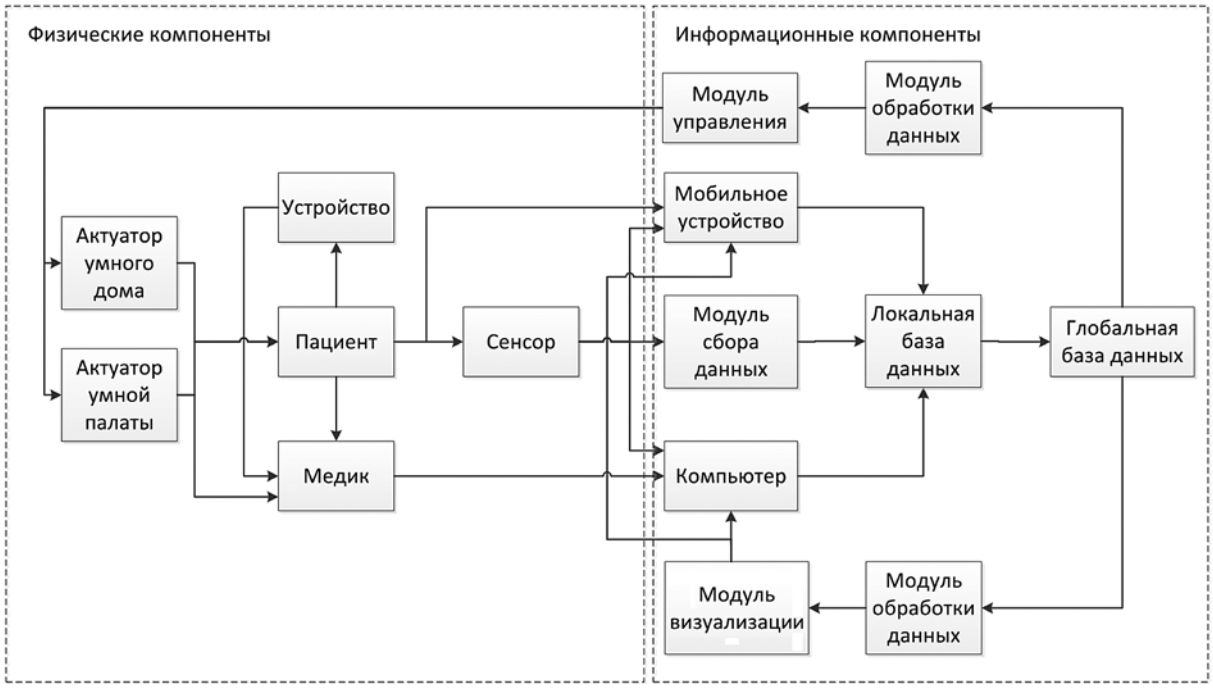


Рис. 2. Обобщенная структурная схема умной палаты

так и информационные процессы. Как информационная система палата содержит компоненты, которые собирают, хранят, обрабатывают, ищут, выдают информацию. Как физическая система палата содержит датчики (устройства, получающие информацию из окружающей среды) и актуаторы (устройства, реализующие физические воздействия). Общая структура УП, отражающая акторов, модули и связи между ними, может быть представлена в виде диаграммы потоков данных. Она представлена на рис. 2.

Организация физической структуры палаты зависит от набора используемых модулей и связей. Информационная составляющая УП описывается с помощью структур данных. Наиболее важные сущности и отношения, используемые в УП, рассмотрены в работах [29, 30].

Характеристика УП в динамике дается с помощью моделей, описывающих ее поведение. Поведение в общем случае складывается из одного или более типовых сценариев функционирования. Ряд сценариев взаимодействия пациента и врача с учетом использования датчиков физиологических показателей организма пациента и системы многомодального ввода информации для него представлен в работе [31].

Подобные модели, а также модели других сценариев и ситуаций могут быть использованы при проектировании УП и ее составных частей. В частности, для разработки программного обеспечения строятся диаграммы классов, а для проектирования систем — диаграммы компонентов, которые могут включать известные и предлагаемые решения. Они могут использоваться для разработки программного обеспечения и позволяют автоматически создавать каркасы программных модулей в системах автоматизированного проектирования.

Рассмотрим практический пример типового модуля, предназначенного для сбора и обработки медицинских данных в частной задаче диагностики синдрома пароксизмальной симпатической гиперактивности (ПСГА). Модуль реализован в виде мобильного приложения, но также может быть выполнен и в виде специализированного устройства. Он получает данные о клинических признаках, обрабатывает их по заданным методикам и формирует оценки для дальнейшего принятия решения врачом. Для этого используются методы и шкалы диагностики ПСГА. На основании суммы баллов принимается решение о возможности постановки диагноза. Использование такого модуля в качестве интегрированного интеллектуального информационного компонента УП обеспечивает ряд преимуществ, к числу которых относятся следующие.

1. Автоматизация ввода данных: сбор данных осуществляется автоматически из разных компонентов и баз данных УП.

2. Автоматизация представления данных: результаты передаются в хранилища данных, что позволяет обрабатывать и представлять их в совокупности с другими данными.

3. Интеграция данных: обеспечивается большая полнота и точность данных, снижается потребность в повторных измерениях, расширяются возможности анализа данных.

Моделирование

Рассмотрим процессы сбора обработки данных в УП на примере ПСГА согласно методике, описанной в работе [32]. Для диагностики и стратификации заболевания собирается ряд показателей, часть из которых измеряется с помощью различных приборов (температура, артериальное давление, частота дыхания, пульс), другие регистрируются путем визуального осмотра, занимающего около 5 мин. Алгоритм одинаков и приведен на рис. 3, однако длительность его шагов различается.

С учетом уровней автоматизации сбора и обработки данных в УП, выделенных в работе [30], возможны следующие варианты поведения системы. На нулевом уровне автоматизации измерение, ввод данных и расчеты выполняются вручную, для оценки времени сбора и обработки данных используются типовые значения времени измерения показателей пациента, а также визуального осмотра. На первом уровне выполняется автоматизация расчетов и оценки тяжести заболевания, измерение и ввод данных по-прежнему выполняются вручную. На втором уровне некоторые показатели измеряются и вводятся в систему автоматически с использованием умных устройств, в частности, предполагается сбор данных об артериальном давлении, пульсе и температуре с монитора. Третий уровень подразумевает непосредственное участие специалиста только в принятии решения, но в силу недостаточности средств автоматизации в этом исследовании он не рассматривается. Гистограммы времени сбора и обработки данных для уровней 0, 1 и 2 показаны на рис. 4, см. третью сторону обложки.

Среднее время сбора и обработки данных составляет соответственно 11,9 мин, 6 мин и 5 мин. Таким образом, автоматизация расчетов позволяет на 50 % сократить время реализации сценария, а использование умного монитора — еще дополнительно на 16 %.

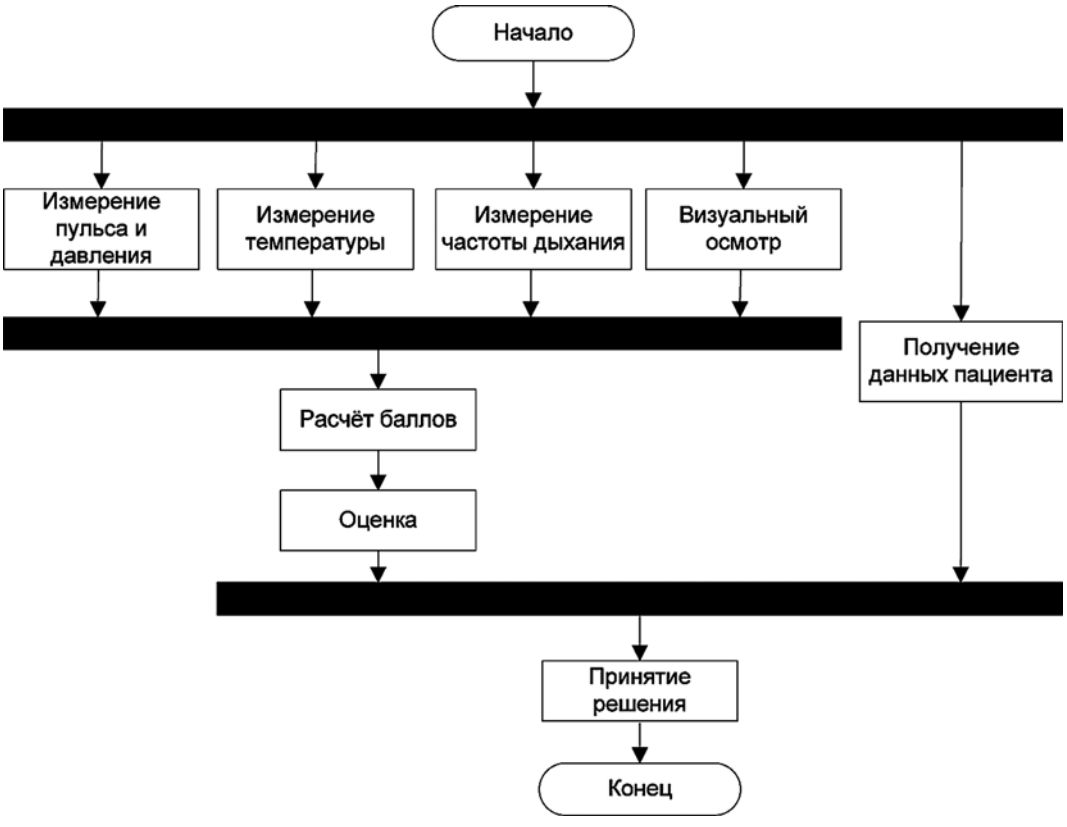


Рис. 3. Алгоритм диагностики и стратификации ПСГА

Заключение

Решения в области УП позволяют обеспечить повышенную безопасность пациентов, улучшить качество их лечения, оптимизировать рабочие процессы в стационарном учреждении, снизить затраты, повысить комфорт и реализовать комплексный «экологический» подход к автоматизации процесса ведения пациентов. Продуманные решения в области УП позволяют сделать процесс лечения более эффективным, экономичным и удобным для пациентов и медицинских специалистов. Реализовывать такие решения необходимо с использованием единого фреймворка с типовыми компонентами, связями, сценариями функционирования.

В данной статье приведены результаты, полученные на начальном этапе реализации такого проекта, а именно — концептуальная модель УП и сценарий сбора и обработки информации в этой палате, позволяющий решить типовую задачу диагностики и стратификации заболевания. Моделирование этого сценария на примере диагностики ПСГА показывает, что время работы персонала значительно (не менее, чем в 2 раза) сокращается за счет автоматизации сбора данных и расчетов по сравнению со сбором и обработкой данных вручную.

Исследование выполнено за счет гранта Российского научного фонда № 22-71-10092, https://rscf.ru/project/22-71-10092/.

Список литературы

1. **Huang P.-H.** The Application of Smart Medical Care in the Smart Ward-Take A Company as an Example. Ph.D. Thesis. College of Management (Executive Master in Business Administration), 2022. 85 p.

2. **Jian W. S., Wang J. Y., Rahmanti A. R.** et al. Voice-based control system for smart hospital wards: a pilot study of patient acceptance // BMC health services research. 2022. Vol. 22, No. 1. P. 287. DOI: 10.1186/s12913-022-07668-1.

3. **Brunete A., Gambao E., Hernando M., Cedazo R.** Smart assistive architecture for the integration of IoT devices, robotic systems, and multimodal interfaces in healthcare environments // Sensors. 2021. Vol. 21, No. 6. P. 2212. DOI: 10.3390/s21062212.

4. **Azhiimah A. N., Khotimah K., Sumbawati M. S., Santosa A. B.** Automatic Control Based on Voice Commands and Arduino. // International Joint Conference on Science and Engineering (IJCSE 2020). Atlantis Press, 2020. P. 29—34. DOI: 10.2991/aer.k.201124.006.

5. **Cronin S., Doherty G.** Touchless computer interfaces in hospitals: A review // Health informatics journal. 2019. Vol. 25, No. 4. P. 1325—1342. DOI: 10.1177/1460458217748342.

6. **Ali H., Cole A., Panos G.** Transforming patient hospital experience through smart technologies // Design, User Experience, and Usability. Case Studies in Public and Personal Interactive Systems: 9th International Conference, DUXU 2020, Held as Part of the 22nd HCI International Conference (HCII 2020). Springer International Publishing, 2020. P. 203—215. DOI: 10.1007/978-3-030-49757-6_14.

7. **Balasubramanian V., Vivekanandhan S., Mahadevan V.** Pandemic tele-smart: a contactless tele-health system for efficient monitoring of remotely located COVID-19 quarantine wards in India using near-field communication and natural language processing system // Medical & Biological Engineering & Computing. 2022. P. 1—19. DOI: 10.1007/s11517-021-02456-1.

8. **Gune A., Bhat A., Pradeep A.** Implementation of near field communication based healthcare management system // 2013 IEEE Symposium on Industrial Electronics & Applications. IEEE, 2013. P. 195—199. DOI: 10.1109/ISIEA.2013.6738993.

9. **Sujadevi V. G., Kumar T. H., Arunjith A. S.** et al. Effortless exchange of personal health records using near field communication // 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE. 2016. P. 1764—1769. DOI: 10.1109/ICACCI.2016.7732303.

10. **Supriya A., Ramgopal S., George S. M.** Near Field Communication based system for health monitoring // 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT). IEEE. 2017. P. 653—657. DOI: 10.1109/RTEICT.2017.8256678.

11. **Thompson P., McNaught J., Ananiadou S.** Customised OCR correction for historical medical text // 2015 digital heritage. IEEE. 2015. Vol. 1. P. 35—42. DOI: 10.1109/DigitalHeritage.2015.7413829.

12. **Wen M. H., Bai D., Lin S.** et al. Implementation and experience of an innovative smart patient care system: A cross-sectional study // BMC Health Services Research. 2022. Vol. 22, No. 1. P. 1—11. DOI: 10.1186/s12913-022-07511-7.

13. **Meade C. M., Bursell A. L., Ketelsen L.** Effects of nursing rounds: on patients' call light use, satisfaction, and safety // AJN The American Journal of Nursing. 2006. Vol. 106, No. 9. P. 58—70. DOI: 10.1097/00000446-200609000-00029.

14. **Tzeng H. M.** Perspectives of staff nurses of the reasons for and the nature of patient-initiated call lights: an exploratory survey study in four USA hospitals // BMC Health Services Research. 2010. Vol. 10, No. 1. P. 1—13. DOI: 10.1186/1472-6963-10-52.

15. **Torres S. M.** Rapid-cycle process reduces patient call bell use, improves patient satisfaction, and anticipates patient's needs // JONA: The Journal of Nursing Administration. 2007. Vol. 37, No. 11. P. 480—482. DOI: 10.1097/01.NNA.0000295609.94699.76.

16. **Klemets J., Evjemo T. E.** Technology-mediated awareness: facilitating the handling of (un) wanted interruptions in a hospital setting // International journal of medical informatics. 2014. Vol. 83, No. 9. P. 670—682. DOI: 10.1016/j.ijmedinf.2014.06.007.

17. **Bouldin E. D., Andresen E. M., Dunton N. E.** et al. Falls among adult patients hospitalized in the United States: prevalence and trends // Journal of patient safety. 2013. Vol. 9, No. 1. P. 13. DOI: 10.1097/PTS.0b013e3182699b64.

18. **Hempel S., Newberry S., Wang Z.** et al. Hospital fall prevention: a systematic review of implementation, components, adherence, and effectiveness // Journal of the American Geriatrics Society. 2013. Vol. 61, No. 4. P. 483—494. DOI: 10.1111/jgs.12169.

19. **Cai X., Pan J.** Toward a brain-computer interface-and internet of things-based smart ward collaborative system using hybrid signals // Journal of Healthcare Engineering. 2022. Vol. 2022. Article ID 6894392. DOI: 10.1155/2022/6894392.

20. **Levina A., Iliashenko V. M., Kalyazina S., Overes E.** Smart Hospital Architecture: IT and Digital Aspects // Algorithms and Solutions Based on Computer Technology: 5th Scientific International Online Conference Algorithms and Solutions based on Computer Technology (ASBC 2021), Cham: Springer International Publishing, 2022. P. 235—247. DOI: 10.1007/978-3-030-93872-7_20.

21. **Yadav S., Manohar M., Jeba Shiney O.** et al. A Smart System for Monitoring Flow in Drip Bottles in Healthcare // Futuristic Communication and Network Technologies: Select Proceedings of VICFCNT 2020. Springer Singapore, 2022. P. 663—669. DOI: 10.1007/978-981-16-4625-6_66.

22. **Gayathri S., Ganesh C. S. S.** Automatic indication system of glucose level in glucose trip bottle // International Jornal of Multidisciplinary Research and Modern Education. 2017. Vol. 3, No. 1. P. 148—151. DOI: 10.5281/zenodo.438622.

23. **Kadam'manja J., Mukamurenzi S., Umuhoza E.** Smart Vital Signs Monitoring System for Patient Triage: A case of Malawi // 2022 International Conference on Computer Communication

and Informatics (ICCCI). IEEE, 2022. P. 1—6. DOI: 10.1109/ICCCI54379.2022.9740751.

24. **Dalal J., Dasbiswas A., Sathyamurthy I.** et al. Heart rate in hypertension: review and expert opinion // International Journal of Hypertension. 2019. Vol. 2019, No. 2087064. P. 1—7. DOI: 10.1155/2019/2087064.

25. **Liu Q., Hou S., Wei L.** Design and implementation of intelligent monitoring system for head and neck surgery care based on internet of things (IoT) // Journal of Healthcare Engineering. 2022. Vol. 2022. Article ID 9806292. DOI: 10.1155/2023/9806292.

26. **Feng Y. S., Liu H. Y., Hsieh M. H.** et al. An RSSI-based device-free localization system for smart wards // 2021 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW). IEEE, 2021. P. 1—2. DOI: 10.1109/ICCE-TW52618.2021.9603249.

27. **Huang Y. C., Liao C. F., Yen Y. C.** et al. An extensible situation-aware caring system for real-world smart wards // Impact Analysis of Solutions for Chronic Disease Prevention and Management: 10th International Conference on Smart Homes and Health Telematics. — Springer Berlin Heidelberg, 2012. — P. 190—197. DOI: 10.1007/978-3-642-30779-9_24.

28. **Abdellatif A. A., Emam A., Chiasserini C. F.** et al. Edge-based compression and classification for smart healthcare systems: Concept, implementation and evaluation // Expert Systems with Applications. 2019. Vol. 117. P. 1—14. DOI: 10.1016/j.eswa.2018.09.019.

29. **Levonevskiy D., Motienko A., Terekhov I.** Automation of diagnosis, stratification, and treatment of the paroxysmal sympathetic hyperactivity syndrome in the smart ward environment // 2nd International Conference on Computer Applications for Management and Sustainable Development of Production and Industry (CMSD-II-2022). SPIE. 2023. Vol. 12564. P. 74—80. DOI: 10.1117/12.2669244.

30. **Levonevskiy D., Motienko A., Tsentsiper L., Terekhov I.** Automation of Data Processing for Patient Monitoring in the Smart Ward Environment // 12th Computer Science On-line Conference (CSOC 2023). 2023. Vol. I, LNNS 722. P. 746—756. DOI: 10.1007/978-3-031-35311-6_71.

31. **Левоневский Д. К., Мотиенко А. И.** Модели сценариев функционирования медицинской киберфизической системы в штатных и экстренных ситуациях // Программная инженерия. 2022. Том 13, № 8. С. 383—393. DOI: 10.17587/prin.13.383-393.

32. **Ценципер Л. М., Терехов И. С., Шевелев О. А.** и др. Синдром пароксизмальной симпатической гиперактивности (обзор) // Общая реаниматология. 2022. Том 18, № 4. С. 55— 67. DOI: 10.15360/1813-9779-2022-4-55-67.

Modeling and Automation of Data Collection and Processing in a Smart Medical Ward

D. K. Levonevskiy, PhD, Senior Researcher, levonevskij.d@iias.spb.su, **A. I. Motienko,** PhD, Senior Researcher, anna.gunchenko@gmail.com,
St. Petersburg Federal Research Center of the Russian Academy of Sciences (SPC RAS),
St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences,
St. Petersburg, 199178, Russian Federation

Corresponding author:

Dmitrii K. Levonevskiy, PhD, Senior Researcher,
St. Petersburg Federal Research Center of the Russian Academy of Sciences,
199178, Saint Petersburg, Russian Federation
E-mail: levonevskij.d@iias.spb.su

Received on July 07, 2023
Accepted on August 11, 2023

The study considers modeling and automating the processes of data collection and processing in a smart medical ward. We examined the existing solutions in the field of smart wards. Such solutions allow to increase patient safety, improve the quality of their treatment, optimize workflows in inpatient facilities, reduce costs and increase comfort of the patients. At the same time, these solutions are focused on particular tasks, use varied technologies and protocols, and implement single scenarios for the functioning of the ward. This is not enough to create an integrated approach to automating patient treatment in smart wards, as there are problems of compatibility and integration of heterogeneous modules, systems and protocols, which leads to growing costs and reduced profitability. To avoid these problems, the complex must be implemented within a single framework with typical components, connections, and operation scenarios. To do this, at the first stage of the work, a conceptual model of a smart room is proposed, a description of its components is given, and scenarios of their functioning are provided. An algorithm for collecting and processing data for diagnosis and stratification of diseases has also been developed, a simulation of the smart ward operation has been carried out, and it showed that the implementation of data collection and processing using smart components can reduce the processing time at least twice.

Keywords: *smart medical ward, medical cyber-physical system, automation in medicine, smart space, human-machine interaction*

Acknowledgements: *The study was supported by the Russian Science Foundation, project No. 22-71-10092, https://rscf.ru/project/22-71-10092/.*

For citation:

Levonevskiy D. K., Motienko A. I. Modeling and Automation of Data Collection and Processing in a Smart Medical Ward, *Programmnaya Ingeneria,* 2023, vol. 14, no. 10, pp. 502—512. DOI: 10.17587/prin.502-512.

References

1. **Huang P.-H.** The Application of Smart Medical Care in the Smart Ward-Take A Company as an Example, Ph.D. Thesis, College of Management (Executive Master in Business Administration), 2022. 85 p.

2. **Jian W. S., Wang J. Y., Rahmanti A. R.** et al. Voice-based control system for smart hospital wards: a pilot study of patient acceptance, *BMC health services research*, 2022, vol. 22, no. 1, pp. 287. DOI: 10.1186/s12913-022-07668-1.

3. **Brunete A., Gambao E., Hernando M., Cedazo R.** Smart assistive architecture for the integration of IoT devices, robotic systems, and multimodal interfaces in healthcare environments, *Sensors*, 2021, vol. 21, no. 6, pp. 2212. DOI: 10.3390/s21062212.

4. **Azhiimah A. N., Khotimah K., Sumbawati M. S., Santosa A. B.** Automatic Control Based on Voice Commands and Arduino, International Joint Conference on Science and Engineering (IJCSE 2020), *Atlantis Press*, 2020, pp. 29—34. DOI: 10.2991/aer.k.201124.006.

5. **Cronin S., Doherty G.** Touchless computer interfaces in hospitals: A review, *Health informatics journal,* 2019, vol. 25, no. 4, pp. 1325—1342. DOI: 10.1177/1460458217748342.

6. **Ali H., Cole A., Panos G.** Transforming patient hospital experience through smart technologies, *Design, User Experience, and Usability. Case Studies in Public and Personal Interactive Systems: 9th International Conference, DUXU 2020, Held as Part of the 22nd HCI International Conference (HCII 2020)*, Springer International Publishing, 2020, pp. 203—215. DOI: 10.1007/978-3-030-49757-6_14.

7. **Balasubramanian V., Vivekanandhan S., Mahadevan V.** Pandemic tele-smart: a contactless tele-health system for efficient monitoring of remotely located COVID-19 quarantine wards in India using near-field communication and natural language processing system, *Medical & Biological Engineering & Computing*, 2022, pp. 1—19. DOI: 10.1007/s11517-021-02456-1.

8. **Gune A., Bhat A., Pradeep A.** Implementation of near field communication based healthcare management system, *2013 IEEE Symposium on Industrial Electronics & Applications*, IEEE, 2013, pp. 195—199. DOI: 10.1109/ISIEA.2013.6738993.

9. **Sujadevi V. G., Kumar T. H., Arunjith A. S.** et al. Effortless exchange of personal health records using near field communication, *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2016. P. 1764—1769. DOI: 10.1109/ICACCI.2016.7732303.

10. **Supriya A., Ramgopal S., George S. M.** Near Field Communication based system for health monitoring, *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, IEEE, 2017, pp. 653-657. DOI: 10.1109/RTEICT.2017.8256678.

11. **Thompson P., McNaught J., Ananiadou S.** Customised OCR correction for historical medical text, 2015 digital heritage. IEEE, 2015. Vol. 1. P. 35—42. DOI: 10.1109/DigitalHeritage.2015.7413829.

12. **Wen M. H., Bai D., Lin S.,** et al. Implementation and experience of an innovative smart patient care system: A cross-sectional study, *BMC Health Services Research*, 2022, vol. 22, no. 1, pp. 1—11. DOI: 10.1186/s12913-022-07511-7.

13. **Meade C. M., Bursell A. L., Ketelsen L.** Effects of nursing rounds: on patients' call light use, satisfaction, and safety, *AJN The American Journal of Nursing*, 2006, vol. 106, no. 9, pp. 58—70. DOI: 10.1097/00000446-200609000-00029.

14. **Tzeng H. M.** Perspectives of staff nurses of the reasons for and the nature of patient-initiated call lights: an exploratory survey study in four USA hospitals, *BMC Health Services Research*, 2010, vol. 10, no. 1, pp. 1—13. DOI: 10.1186/1472-6963-10-52.

15. **Torres S. M.**Rapid-cycle process reduces patient call bell use, improves patient satisfaction, and anticipates patient's needs, *JONA: The Journal of Nursing Administration*, 2007, vol. 37, no. 11, pp. 480—482. DOI: 10.1097/01.NNA.0000295609.94699.76.

16. **Klemets J., Evjemo T. E.** Technology-mediated awareness: facilitating the handling of (un) wanted interruptions in a hospital setting, *International journal of medical informatics*, 2014, vol. 83, no. 9, pp. 670-682. DOI: 10.1016/j.ijmedinf.2014.06.007.

17. **Bouldin E. D., Andresen E. M., Dunton N. E.** et al. Falls among adult patients hospitalized in the United States: prevalence and trends, *Journal of patient safety*, 2013, vol. 9, no. 1, pp. 13. DOI: 10.1097/PTS.0b013e3182699b64.

18. **Hempel S., Newberry S., Wang Z.** et al. Hospital fall prevention: a systematic review of implementation, components, adherence, and effectiveness, *Journal of the American Geriatrics Society*, 2013, vol. 61, no. 4, pp. 483-494. DOI: 10.1111/jgs.12169.

19. **Cai X., Pan J.** Toward a brain-computer interface-and internet of things-based smart ward collaborative system using hybrid signals, *Journal of Healthcare Engineering*, 2022, vol. 2022, article ID 6894392. DOI: 10.1155/2022/6894392.

20. **Levina A., Iliashenko V. M., Kalyazina S., Overes E.** Smart Hospital Architecture: IT and Digital Aspects, *Algorithms and Solutions Based on Computer Technology: 5th Scientific International Online Conference Algorithms and Solutions based on Computer Technology (ASBC 2021)*, Cham: Springer International Publishing, 2022, pp. 235—247. DOI: 10.1007/978-3-030-93872-7_20.

21. **Yadav S., Manohar M., Jeba Shiney O.** et al. A Smart System for Monitoring Flow in Drip Bottles in Healthcare, *Futuristic Communication and Network Technologies: Select Proceedings of VICFCNT 2020*, Springer Singapore, 2022, pp. 663—669. DOI: 10.1007/978-981-16-4625-6_66.

22. **Gayathri S., Ganesh C. S. S.** Automatic indication system of glucose level in glucose trip bottle, I*nternational Jornal of Multidisciplinary Research and Modern Education*, 2017, vol. 3, no. 1, pp. 148-151. DOI: 10.5281/zenodo.438622.

23. **Kadam'manja J., Mukamurenzi S., Umuhoza E.** Smart Vital Signs Monitoring System for Patient Triage: A case of Malawi, *2022 International Conference on Computer Communication and Informatics (ICCCI)*, IEEE, 2022, pp. 1-6. DOI: 10.1109/ICCCI54379.2022.9740751.

24. **Dalal J., Dasbiswas A., Sathyamurthy I.** et al. Heart rate in hypertension: review and expert opinion, *International Journal of Hypertension*, 2019, vol. 2019, no. 2087064, pp. 1—7. DOI: 10.1155/2019/2087064.

25. **Liu Q., Hou S., Wei L.** Design and implementation of intelligent monitoring system for head and neck surgery care based on internet of things (IoT), *Journal of Healthcare Engineering*, 2022, vol. 2022, article ID 9806292. DOI: 10.1155/2023/9806292.

26. **Feng Y. S., Liu H. Y., Hsieh M. H.** et al. An RSSI-based device-free localization system for smart wards, *2021 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW),* IEEE, 2021, pp. 1—2. DOI: 10.1109/ICCE-TW52618.2021.9603249.

27. **Huang Y. C., Liao C. F., Yen Y. C.** et al. An extensible situation-aware caring system for real-world smart wards, *Impact Analysis of Solutions for Chronic Disease Prevention and Management: 10th International Conference on Smart Homes and Health Telematics*, Springer Berlin Heidelberg, 2012, pp. 190—197. DOI: 10.1007/978-3-642-30779-9_24.

28. **Abdellatif A. A., Emam A., Chiasserini C. F.** et al. Edge-based compression and classification for smart healthcare systems: Concept, implementation and evaluation, *Expert Systems with Applications*, 2019, vol. 117, pp. 1—14. DOI: 10.1016/j.eswa.2018.09.019.

29. **Levonevskiy D., Motienko A., Terekhov I.** Automation of diagnosis, stratification, and treatment of the paroxysmal sympathetic hyperactivity syndrome in the smart ward environment, *2nd International Conference on Computer Applications for Management and Sustainable Development of Production and Industry (CMSD-II-2022)*, SPIE, 2023, vol. 12564, pp. 74—80. DOI: 10.1117/12.2669244.

30. **Levonevskiy D., Motienko A., Tsentsiper L., Terekhov I.** Automation of Data Processing for Patient Monitoring in the Smart Ward Environment, *12th Computer Science On-line Conference (CSOC 2023)*, 2023, vol. I, LNNS 722, pp. 746—756. DOI: 10.1007/978-3-031-35311-6_71.

31. **Levonevskiy D. K., Motienko A. I.** Scenario Models for the Functioning of a Medical Cyber-Physical System in Normal and Emergency Situations, *Programmnaya Ingeneria*, 2022, vol. 13, no. 8, pp. 383—393. DOI: 10.17587/prin.13.383-393 (in Russian).

32. **Tsentsiper L. M., Terekhov I. S., Shevelev O. A.** et al. Paroxysmal Sympathetic Hyperactivity Syndrome (Review), *General Reanimatology,* 2022, vol. 18, no. 4, pp. 55—67. DOI: 10.15360/1813-9779-2022-4-55-67 (in Russian).

УДК 004.932
DOI: 10.17587/prin.14.513-519

Р. М. Хусаинов, аспирант, rumil_husainov98@mail.ru,
Н. Г. Талипов, канд. техн. наук, доц., nafis.talipov@mail.ru,
А. С. Катасёв, д-р техн. наук, проф., askatasev@kai.ru,
Д. В. Шалаева, магистрант, dvshalaeva@bk.ru,
Казанский национальный исследовательский технический университет
им. А. Н. Туполева-КАИ

Нейросетевая технология анализа транспортных потоков в автоматизированных системах управления дорожным движением

Поступила в редакцию 14.07.2023
Принята к публикации 09.08.2023

Представлены результаты решения задачи разработки нейросетевой технологии анализа транспортных потоков в автоматизированных системах управления дорожным движением. Описаны методы анализа транспортных потоков с применением различных технологий, существующие системы анализа транспортных потоков на основе нейросетевых технологий и технологии обнаружения и отслеживания объектов на видеопотоке в целях обеспечения безопасности дорожного движения. Приведены алгоритмы, используемые в работе нейросетевой технологии, включающие этапы обнаружения объектов, отслеживания обнаруженных объектов, выявления инцидентов, автоматического сбора информации из видеопотоков, сбора статистики. По результатам апробации (тестирования) реализованной нейросетевой технологии на собственных (подготовленных) и используемых видеоданных, загруженных из сети Интернет, достоверность результатов исследования видеокадров (верного распознавания объектов транспортного потока) в нейросетевой системе управления дорожным движением составила 85...90 %.

Ключевые слова: *транспортные потоки, распознавание объектов, обнаружение объектов, инцидент, нейронная сеть, дорожно-транспортные происшествия, транспортное средство, дорожное движение, видеопоток, нейросетевая технология*

Для цитирования:

Хусаинов Р. М., **Талипов Н. Г.**, **Катасёв А. С.**, **Шалаева Д. В.** Нейросетевая технология анализа транспортных потоков в автоматизированных системах управления дорожным движением // Программная инженерия. 2023. Том. 14, № 10. С. 513—519. DOI: 10.17587/prin.14.513-519

Введение

По официальным данным государственной инспекции безопасности дорожного движения, в Российской Федерации за 2022 г. совершено 103 912 дорожно-транспортных происшествий (ДТП), в которых погиб 11 501 человек, а 130 698 человек получили травмы различной степени тяжести. По Республике Татарстан произошло 2760 ДТП, в которых 256 человек погибли, 3386 человек были ранены [1].

При решении задач по регулированию дорожного движения и управлению транспортными системами широко применяются интеллектуальные транспортные системы, способные эффективно управлять улично-дорожной сетью с учетом ее плотности и пропускной способности [2—4]. Основными элементами улично-дорожной сети являются улицы, проспекты, пешеходные и велосипедные дорожки, тротуары и др. Особенностью таких систем является автоматическое формирование управляющих воздействий в режиме реального

времени на объекты транспортной инфраструктуры с частичным участием оператора. Важными элементами в составе интеллектуальной транспортной инфраструктуры являются подсистемы на основе нейросетевых технологий.

Существуют программное обеспечение с открытым исходным кодом (Open Source) и коммерческие решения анализа транспортных потоков, прогнозирования аномалий и инцидентов. Среди популярных коммерческих решений можно выделить аппаратно-программный комплекс SecurOS Soffit, систему идентификации и раннего оповещения ДТП Crash AI и автоматизированную систему управления дорожным движением (АСУДД) «Мегаполис» [5—7].

Интеллектуальная система SecurOS Soffit визуального (светового) сопровождения людей используется для предотвращения непредумышленного наезда на пешеходов на нерегулируемых пешеходных переходах. Система Crash AI реализована на основе нейронной сети для определения и анализа тяжести аварий. Система «Мегаполис» позволяет наблюдать дорожную обстановку вместе с получением аналитических численных показателей. Преимуществом коммерческих систем является высокое качество конечного продукта, высокий класс защиты комплекса от внешних воздействий и наличие технической поддержки. Однако данные системы имеют недостатки [8, 9]. Сравнительная оценка упомянутых систем представлена в табл. 1.

Таким образом, актуальным для решения прикладных задач является использование нейросетевых технологий и методов создания нейросетевых систем. Для обнаружения объектов на изображениях транспортной инфраструктуры целесообразно использовать сверточные нейронные сети [10—12]. Нейросетевые системы хорошо извлекают признаки из изображений и эффективно используются в таких задачах, как классификация, распознавание образов, сегментация и анализ изображений [13, 14].

В данной работе для распознавания объектов на видеоизображениях использована наиболее популярная архитектура сверточной нейронной сети YOLO. Она не требует привлечения больших аппаратных ресурсов, благодаря чему ее можно реализовать на различных платформах. Различные архитектуры сверточных сетей из класса YOLO обеспечивают высокую точность и скорость детектирования объектов на изображениях [15—17].

Кроме распознавания объектов на видеоизображениях в анализе транспортных потоков необходимо решать задачу слежения за движущимися объектами. Широкое распространение получили интеллектуальные системы видеонаблюдения, в состав которых входят специальный программный модуль анализа потока видеоданных, поступающего с цифровой видеокамеры, и автоматический трекинг (отслеживание) передвигающегося в поле зрения камеры объекта. Для отслеживания объектов выбран детектор DeepSort [18—20], совмещающий такие качества, как простота и высокая точность детектирования.

Для комплексного решения перечисленных задач актуальна разработка нейросетевой технологии анализа транспортных потоков в целях обеспечения безопасности дорожного движения. Для эффективной работы АСУДД необходимы видеоданные с камер уличного видеонаблюдения, установленные на участках дорожного движения. Видеоданные должны разделяться на отдельные кадры и передаваться в нейронную сеть, решающую задачи распознавания и слежения за движущимися объектами. На основе результатов работы нейронной сети с помощью специальных алгоритмов можно осуществлять построение траекторий движения, вычисление векторов движения объектов дорожной инфраструктуры, определять аномалии и инциденты, такие как незапланированная остановка транспортного средства, пробка, ДТП и т. д.

Таблица 1

Сравнительная оценка систем анализа транспортных потоков

Решение	Достоинства	Недостатки
SecurOS Soffit	Техническая поддержка, качество	Высокая стоимость, необходимость доработок
Crash AI	Техническая поддержка, качество	Высокая стоимость, необходимость доработок
АСУДД «Мегаполис»	Полнота решения, качество	Высокая стоимость
Open Source	Доступность	Низкое качество

Постановка задачи применения нейросетевой технологии анализа транспортных потоков

Для решения задачи анализа транспортных потоков необходимо использовать нейросетевую технологию, включающую следующие этапы: обнаружение объектов; отслеживание обнаруженных объектов; выявление инцидентов; автоматический сбор информации из видеопотоков; сбор статистики. Для этого требуется разработать следующие алгоритмы: подготовки видеокадров; распознавания движущихся объектов; слежения за движущимися объектами; сбора статистики и формирования выходного видеопотока. Рассмотрим решение этих задач более подробно.

Разработка алгоритмов подготовки видеокадров, распознавания и слежения за движущимися объектами, сбора статистики и формирования выходного видеопотока

Алгоритм подготовки видеокадров для передачи на вход программной реализации нейронной сети должен включать решение следующих задач: загрузка входящего видеопотока; нормализация и масштабирование видеопотока; получение готовых кадров с необходимым разрешением и масштабом. В нейронную сеть исходный видеопоток может поступать напрямую из видеокамер дорожного наблюдения или путем загрузки сохраненных записей движения транспортных потоков. В ходе проведения исследования для обучения нейронной сети YOLOv3 использовались собственные (подготовленные) кадры и кадры из видеопотока (записи камер дорожного наблюдения), загруженные из сети Интернет[1]. Схема алгоритма подготовки видеокадров представлена на рис. 1.

Кадры входного видеопотока для предварительно обученной нейронной сети YOLOv3 выбраны с разрешением 1280×720 пикселей и форматом 16:9, могут использоваться и другие параметры[2].

После того как входящий видеопоток обработан, полученные нормализованные кадры передаются в нейронную сеть YOLOv3, которая обрабатывает каждый кадр, распознает объекты, определяет классы и положение объектов в кадре. Результатом выполнения процесса данного блока являются распознанные объекты в виде прямоугольников вокруг объектов. Схема алгоритма сбора статистики и формирования выходного видеопотока показана на рис. 2.

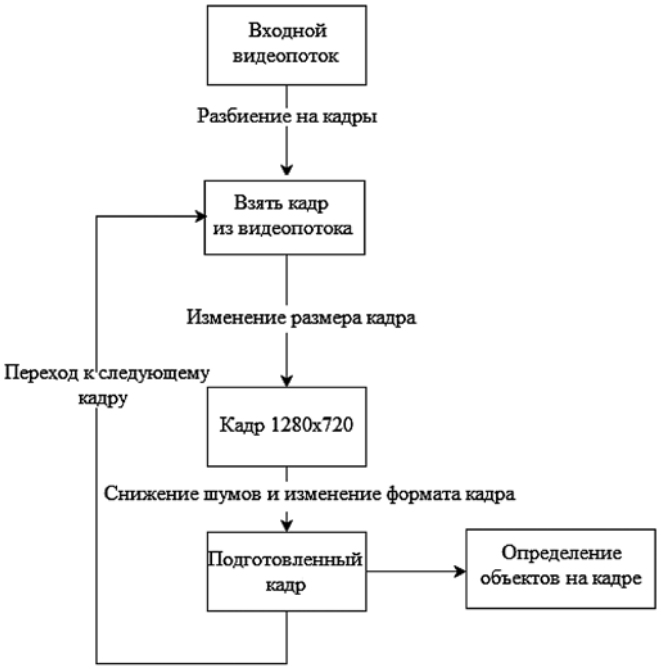


Рис. 1. Схема алгоритма подготовки видеокадров

Рис. 2. Схема алгоритма сбора статистики и формирования выходного видеопотока

[1] https://dzen.ru/a/X2365GOyXQTNVy34
[2] https://habr.com/ru/articles/514450/

После того как YOLOv3 провела распознавание объектов, координаты и размеры полученных обнаруженных прямоугольников вокруг объектов передаются объекту глубокой сортировки для дальнейшего отслеживания.

Для отслеживания объектов используется детектор DeepSort, который осуществляет трекинг распознанных на предыдущем этапе нейронной сетью YOLOv3 объектов от кадра к кадру и анализ трека объектов для распознавания инцидентов (остановка, затор, ДТП). Результаты решения задачи отслеживания объектов с помощью детектора DeepSort передаются разработанным алгоритмам в виде координат и параметров трека объектов.

На основе алгоритма сбора статистики и формирования выходного видеопотока вычисляется и строится траектория движения объектов, вычисляется вектор движения каждого обнаруженного транспортного средства, определяются различные аномалии и инциденты. Алгоритмы имеют следующие особенности:

- вектор движения вычисляется как отрезок, построенный на основе двух предыдущих положений объекта в кадре, а длина отрезка зависит от длины предыдущей траектории и размера объекта (дальние объекты имеют меньший вектор);
- инцидент «остановка» определяется в случае, если координаты центра объекта не изменяются в течение некоторого времени, т. е. можно говорить о том, что объект не двигается;
- определение инцидента «затор» происходит в том случае, если фиксируется нахождение большого количества транспортных средств;
- определение ДТП происходит в случае, если векторы обнаруженных транспортных средств пересекаются (это значит, что транспортные средства двигаются слишком быстро и высока вероятность ДТП).

В результате работы предыдущих алгоритмов собирается, аккумулируется и выводится информация в виде статистики и графиков по инцидентам. По результатам расчетов с применением библиотеки OpenCV формируется выходной видеопоток с отображенными на нем обнаруженными объектами, траекториями их движения и векторами, а также зафиксированными аномалиями и инцидентами.

Реализация и апробация нейросетевой технологии анализа транспортных потоков

Для анализа транспортных потоков в ходе проведения исследования используется программная реализация YOLOv3, необходимо подготовить соответствующее виртуальное окружение, расширяющее возможности уже существующей программной реализации сети под задачу анализа транспортных потоков. Для этого принято решение использовать среду Anaconda [21], позволяющую инкапсулировать все необходимые библиотеки в одном месте.

Перед началом работы в виртуальное окружение установлена библиотека OpenCV для работы с изображениями [22]. Также установлены библиотека Tensorflow и API Keras для работы с нейронной сетью YOLOv3 и детектором DeepSort [23—25]. В результате реализации нейросетевой технологии YOLOv3 появилась возможность эффективно распознавать следующие объекты: автомобиль, грузовик, мотоцикл, велосипед, пешеход. Кроме того, распознаются такие инциденты, как ДТП, остановка, скопление машин, пробка.

Результаты выполнения задач анализа транспортных потоков представлены на рис. 3—6, см. третью и четвертую стороны обложки.

По результатам апробации реализованной нейросетевой технологии на наборе собственных (подготовленных) и используемых видеоданных, загруженных из сети Интернет, достоверность результатов исследования видеокадров (верного распознавания объектов транспортного потока) составила 85...90 %. Ошибки возникали при наличии большого числа объектов на кадрах видеоизображений и плохого качества видеопотока. При мониторинге параметров транспортного потока обеспечивались сбор, анализ и предоставление агрегированных данных о транспортной обстановке на участке дорожного движения.

Формирование собранных статистических данных реализовано в виде json-файлов. Эти данные передаются в собранном виде, массив информации дополнительно анализируется для выявления потенциально интересных для дальнейшей оптимизации потока фактов и событий.

Статистика представляет собой два типа файлов:

- «сырые» данные о времени, типе и секторе кадра, в котором произошел инцидент (вся информация, которая была собрана за время исследования, до начала чистки, редактирования и статистической обработки);
- агрегированная информация о числе инцидентов за промежуток времени.

Полученные в ходе выполнения исследования статистические данные можно использовать в автоматизированных системах управления дорожным движением для анализа транспортных потоков и в других системах управления и приня-

Таблица 2

Выявленные ограничения, влияющие на работу системы

Ограничения	Влияние на результат	Решение
Разный угол установки камер	Среднее	Измерение соответствующих параметров и настройка коэффициентов на территории
Посторонние объекты	Сильное	Согласование с соответствующими городскими службами
Погодные условия	Среднее	Дополнительное обучение моделей машинного обучения
Число объектов	Среднее	Увеличение вычислительных мощностей, применение специальных устройств
Качество видео	Сильное	Качественная работа с ответственными городскими службами

тия решения для повышения оперативности реагирования на возникновение инцидентов. В ходе выполнения исследования также определены проблемы существующей инфраструктуры, преимущества и недостатки предложенного подхода, а также пути возможного их решения. Выявленные ограничения, влияющие на работу системы, приведены в табл. 2.

Полученные результаты могут быть применены на практике при создании нейросетевых систем для решения прикладных задач, в том числе для своевременного реагирования соответствующими городскими службами на возникающие проблемные ситуации на дорогах города в режиме реального времени, для использования их в автоматизированных системах управления дорожным движением, в работе ситуационных центров. Однако развитие описанной технологии и вывод ее на этап коммерциализации в виде конечного продукта возможны только после выполнения определенных доработок, учитывающих выявленные проблемы и недостатки.

Заключение

В статье описана нейросетевая технология анализа транспортных потоков и выявления таких инцидентов, как ДТП, остановка и пробка на дорогах муниципальных образований для получения более точной статистики и прогнозирования аномалий и инцидентов. Апробация технологии на наборе собственных (подготовленных) и используемых видеоданных, загруженных из сети Интернет, а также практическое использование для анализа транспортных потоков, прогнозирования и выявления инцидентов показала ее эффективность и практическую пригодность к решению поставленных прикладных задач. В перспективе целесообразно совершенствование технологии с учетом трудностей и недостатков, выявленных в ходе ее апробации.

Список литературы

1. **Показатели** состояния безопасности дорожного движения. URL: http://stat.gibdd.ru/ (дата обращения 04.07.2023).
2. **Ермишина Е. В., Шарыпова Т. Н.** Критическая информационная инфраструктура в интеллектуальной транспортной системе // Colloquim-Journal. 2022. № 2-1 (125). С. 20—21.
3. **Соколянский В. В., Булатов А. Б., Булатов Р. Б., Дудаев Б. В.** Интеллектуальные транспортные системы в городской инфраструктуре // Вопросы экономических наук. 2015. № 4 (74). С. 75—81.
4. **Низяева Ю. Д., Слободчиков Н. А.** Опыт внедрения интеллектуальных транспортных систем в городскую инфраструктуру // Системный анализ и логистика. 2022. № 2 (32). С. 50—55. DOI: 10.31799/2077-5687-2022-2-50-55.
5. **Интеллектуальная** система предотвращения ДТП**.** URL: https://iss.ru/products/securos-soffit (дата обращения 26.07.2023).
6. **Российская** ИИ-система Crash AI предупредит о риске возникновения ДТП. URL: https://3dnews.ru/981314 (дата обращения 26.07.2023).
7. **АСУДД** «Мегаполис». URL: https://asudd.pro/products/asudd/ (дата обращения: 26.07.2023).
8. **ПО** с открытым исходным кодом: почему сегодня это особенно опасно и как защититься? URL: https://www.tadviser.ru/index.php (дата обращения 26.07.2023).
9. **Слабое** звено Open Source**.** URL: https://innostage-group.ru/press/blog/review/slaboe-zveno-open-source (дата обращения 26.07.2023).
10. **Lecun Y., Boser B., Denker J. S., Henderson D., Howard R. E., Hubbard W., Jackel L. D.** Backpropagation applied to handwritten zip code recognition // Neural Computation. 1989. P. 541—551.
11. **Shleymovich M. P., Dagaeva M. V., Katasev A. S.** et al. The analysis of images in control systems of unmanned automobiles on the base of energy features model // Computer Research and Modeling. 2018. No. 10 (3). P. 369—376. DOI: 10.20537/2076-7633-2018-10-3-369-376.
12. **Хусаинов Р. М., Талипов Н. Г.** Анализ алгоритмов и систем распознавания знаков дорожного движения // Вестник технологического университета. 2022. Т. 25, № 3. С. 72—77.
13. **Друки А. А.** Применение сверточных нейронных сетей для выделения и распознавания автомобильных номерных знаков на изображениях со сложным фоном // Известия

Томского политехнического университета. 2014. Т. 324, № 5. С. 85—92.

14. **Хомоненко А. Д., Яковлев Е. Л.** Обоснование архитектуры сверточной нейронной сети для автономного распознавания объектов на изображениях бортовой вычислительной системой // Наукоемкие технологии в космических исследованиях Земли. 2018. Т. 10, № 6. С. 86—93.

15. **Боков П. А., Кравченя П. Д.** Экспериментальный анализ точности и производительности разновидностей архитектур Yolo для задач компьютерного зрения // Программные продукты и системы. 2020. № 4. С. 635—640. DOI: 10.15827/0236-235X.132.635-640.

16. **Руденький А. О.** Применение архитектуры Yolo для выявления нарушения правил, остановки, стоянки транспортных средств в неположенном месте и езды по тротуарам // Международная научно-техническая конференция молодых ученых БГТУ им. В. Г. Шухова. Материалы конференции. 2021. С. 3779—3788.

17. **Шнайдер А. С.** Детектирование объектов на изображениях на примере модели Yolo // Труды молодых ученых Алтайского государственного университета. 2018. № 15. С. 358—361.

18. **Jung H., Choi M.-K., Jung J.** et al. ResNet-Based Vehicle Classificaton an Localization in Traffic Surveillance Systems // IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2017. P. 934—940. **DOI:** 10.1109/CVPRW.2017.129.

19. **Багиров М. Б.** Разработка и исследование алгоритмов сопровождения объектов на видеопотоке // Информационные системы и технологии. Сб. материалов XXVIII Международной научно-технической конференции. 2022. С. 350—360.

20. **Wojke N., Bewley A., Paulus D.** Simple Online and Realtime Tracking with a Deep Association Metric // IEEE International Conference on Image Processing. 2017. P. 1—5. DOI: 10.1109/ICIP.2017.8296962.

21. **Официальный** сайт Anaconda. URL: https://www.anaconda.com/products/individual (дата обращения 04.07.2023).

22. **Официальный** сайт разработчика Джозефа Чет Редмона**.** URL: https://pjreddie.com/media/files/yolov3.weights (дата обращения 04.07.2023).

23. **Сети.** Google-Диск. URL: https://drive.google.com/drive/folders/1m2ebLHB2JThZC8vWGDYEKGsevLssSkjo (дата обращения 04.07.2023).

24. **Официальный** репозиторий разработчика github. URL: https://github.com/nwojke/deep_sort (дата обращения 04.07.2023).

25. **Официальный** репозиторий разработчика github. URL: https://github.com/qqwweee/keras-yolo3 (дата обращения 04.07.2023).

Neural Network Technology for Traffic Flow Analysis in Automated Traffic Control Systems

R. M. Khusainov, Postgraduate Student, rumil_husainov98@mail.ru,
N. G. Talipov, PhD (technical science), Associate Professor, nafis.talipov@mail.ru,
A. S. Katasev, Dr. Sci. (technical science), Professor, ASKatasev@kai.ru,
D. V. Shalaeva, Master Student, dvshalaeva@bk.ru,
Kazan National Research Technical University named after A. N. Tupolev, Kazan, 420111, Russian Federation

Corresponding author:

Nafis G. Talipov, PhD (technical science), Associate Professor,
Kazan National Research Technical University named after A. N. Tupolev, Kazan, 420111, Russian Federation
E-mail: nafis.talipov@mail.ru

Received on July 14, 2023
Accepted on August 09, 2023

The article presents the results of solving the problem of developing a neural network technology for analyzing traffic flows in automated traffic control systems. Methods for analyzing traffic flows using various technologies, existing systems for analyzing traffic flows based on neural network technologies and technologies for detecting and tracking objects on a video stream in order to ensure traffic safety are described. The algorithms used in the operation of neural network technology are described, including the stages of object detection; tracking of detected objects; identifying incidents; automatic collection of information from video streams; collecting statistics. According to the results of approbation (testing) of the implemented neural network technology on own (prepared) and used video data downloaded from the Internet, the reliability of the results of the study of video frames (correct recognition of traffic flow objects) in the neural network traffic control system was 85—90 %. Errors occurred in the presence of a large number of objects on a video frames and poor quality of the video stream. When monitoring the parameters of the traffic flow, the collection, analysis and provision of aggregated data on the traffic situation on the road section were ensured. Automatic collection of information from video files is implemented up to the file time without reference to real time. The statistical data obtained during the study can be used in automated traffic control systems to analyse traffic flows and in other control and decision-making systems to increase the responsiveness to incidents.

The results obtained can be applied in traffic flow studies to obtain accurate statistics and forecast anomalies and incidents (traffic accidents, traffic jams) in order to ensure traffic safety.

Keywords: *traffic flows, object recognition, object detection, incident, neural network, traffic accidents, vehicle, traffic, video stream, neural network technology*

For citation:

Khusainov R. M., **Talipov N. G.**, **Katasev A. S.**, **Shalaeva D. V**. Neural Network Technology for Traffic Flow Analysis in Automated Traffic Control Systems, *Programmnaya Ingeneria,* 2023, vol. 14, no. 10, pp. 513—519. DOI: 10.17587/prin.14.513-519.

References

1. **Indicators** of the state of road safety, available at: http://stat.gibdd.ru/ (date of access 04.07.2023).

2. **Yermishina E. V., Sharypova T. N.** Critical information infrastructure in the intelligent transport system, *Colloquim-Journal,* 2022, no. 2-1 (125), pp. 20—21 (in Russian).

3. **Sokolyansky V. V, Bulatov A. B., Bulatov R. B., Dudayev B. V.** Intelligent transport systems in urban, *Voprosy ekonomicheskikh nauk*, 2015, no. 4 (74), pp. 75—81 (in Russian).

4. **Nizyaeva Yu. D., Slobodchikov N. A.** Experience in implementing intelligent transport systems in urban infrastructure, *Sistemnyy analiz i logistika*. 2022, no. 2 (32), pp. 50—55. DOI: 10.31799/2077-5687-2022-2-50-55 (in Russian).

5. **Intelligent** accident prevention system, available at: https://iss.ru/products/securos-soffit (date of access 26.07.2023).

6. **The Russian** AI system Crash AI will warn of the risk of an accident, available at: https://3dnews.ru/981314 (date of access 26.07.2023).

7. **ASUDD** "Megapolis", available at: https://asudd.pro/products/asudd/ (date of access 26.07.2023).

8. **Open** source software: why is it especially dangerous today and how to protect yourself, available at: https://www.tadviser.ru/index.php (date of access 26.07.2023).

9. **The weak** link of Open Source, available at: https://innostage-group.ru/press/blog/review/slaboe-zveno-open-source (date of access 26.07.2023).

10. **Lecun Y., Boser B., Denker J. S., Henderson D., Howard R. E., Hubbard W., Jackel L. D.** Backpropagation applied to handwritten zip code recognition, *Neural Computation*, 1989. pp. 541—551.

11. **Shleymovich M. P., Dagaeva M. V., Katasev A. S**. et al. The analysis of images in control systems of unmanned automobiles on the base of energy features model, *Computer Research and Modeling,* 2018, no. 10 (3), pp. 369—376. DOI: 10.20537/2076-7633-2018-10-3-369-376 (in Russian).

12. **Khusainov R. M., Talipov N. G.** Analysis of algorithms and systems for recognizing traffic signs, *Vestnik tekhnologicheskogo universiteta,* 2022, vol. 25, no. 3, pp. 72—77 (in Russian).

13. **Druki A. A.** The use of convolutional neural networks for the extraction and recognition of license plates in images with a complex background, *Izvestiya Tomskogo politekhnicheskogo universiteta*, 2014, vol. 324, no. 5, pp. 85—92 (in Russian).

14. **Homonenko A. D., Yakovlev E. L.** Substantiation of the architecture of a convolutional neural network for autonomous recognition of objects in images by an onboard computer system, *Naukoyemkiye tekhnologii v kosmicheskikh issledovaniyakh Zemli*, 2018, vol. 10, no. 6, pp. 86—93 (in Russian).

15. **Bokov P. A., Kravcheniya P. D.** Experimental analysis of the accuracy and performance of varieties of Yolo architectures for computer vision tasks, *Programmnyye produktyi sistemy,* 2020, no. 4, pp. 635—640. DOI: 10.15827/0236-235X.132.635-640 (in Russian).

16. **Rudenkyy A. O.** Application of architecture to detect violations of rules, stopping, parking vehicles in the wrong place and driving on sidewalks, *Mezhdunarodnaya nauchno-tekhnicheskaya konferentsiya molodykh uchenykh BGTU named V. G. Shukhova*, Materialy konferentsii, 2021, pp. 3779—3788 (in Russian).

17. **Schneider A. S.** Detection of objects in images on the example of the Yolo model, *Trudy molodykh uchenykh Altayskogo gosudarstvennogo universiteta*, 2018, no. 15, pp. 358—361 (in Russian).

18. **Jung H. Choi M.-K., Jung J.** et al. ResNet-Based Vehicle Classificaton an Localization in Traffic Surveillance Systems, *IEEE Conference on Computer Vision and Pattern Recognition Workshops,* 2017, pp. 934—940. DOI: 10.1109/CVPRW.2017.129.

19. **Bagirov M. B.** Development and research of algorithms for tracking objects on a video stream, *Informatsionnyye sistemy i tekhnologii. Sbornik materialov XXVIII Mezhdunarodnoy nauchno-tekhnicheskoy konferentsii*, 2022, pp. 350—360 (in Russian).

20. **Wojke N., Bewley A., Paulus D.** Simple Online and Realtime Tracking with a Deep Association Metric, *IEEE International Conference on Image Processing,* 2017, pp. 1—5. DOI: 10.1109/ICIP.2017.8296962.

21. **Anaconda** official site, available at: https://www.anaconda.com/products/individual (date of access 04.07.2023).

22. **Official** site of the developer Joseph Chet Redmon, available at: https://pjreddie.com/media/files/yolov3.weights (date of access 04.07.2023).

23. **Networks.** Google-Disk, available at: https://drive.google.com/drive/folders/1m2ebLHB2JThZC8vWGDYEKGsevLssSkjo (date of access 04.07.2023).

24. **Developer's** official github repository, available at: https://github.com/nwojke/deep_sort (date of access 04.07.2023).

25. **Developer's** official github repository, available at: https://github.com/qqwweee/keras-yolo3 (date of access 04.07.2023).

ООО "Издательство "Новые технологии". 107076, Москва, ул. Матросская Тишина, д. 23, стр. 2
Технический редактор *Е. В. Конова*. Корректор *А. В. Чугунова*.

Сдано в набор 18.08.2023 г. Подписано в печать 28.09.2023 г. Формат 60×88 1/8. Заказ PI1023
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1. Сайт: **www.aov.ru**

29—30 ноября 2023 г. в ГосНИИАС состоится

V Всероссийская научно-техническая конференция

«МОДЕЛИРОВАНИЕ АВИАЦИОННЫХ СИСТЕМ»

Достижение высокой эффективности и надежности авиационных систем требует системного подхода в вопросах интеграции инновационных решений и прогрессивных технологий. При этом актуальным инструментом становится технология моделирования, которая позволяет проводить исследования и разработки в области создания перспективных авиационных систем, определять научные концепции, оценивать эффективность и прогнозировать результаты внедрения критических технологий. Государственный научный центр РФ Государственный научно-исследовательский институт авиационных систем (ГосНИИАС) является ведущим научным центром в области разработки перспективных авиационных систем и обладает уникальными компетенциями и опытно-экспериментальной базой.

Секции конференции:

- МОДЕЛИРОВАНИЕ ПРИ ОПРЕДЕЛЕНИИ НАУЧНЫХ КОНЦЕПЦИЙ И ОЦЕНКЕ ЭФФЕКТИВНОСТИ АВИАЦИОННЫХ СИСТЕМ
- МАТЕМАТИЧЕСКОЕ И ПОЛУНАТУРНОЕ МОДЕЛИРОВАНИЕ АВИАЦИОННЫХ КОМПЛЕКСОВ
- МОДЕЛИРОВАНИЕ СИСТЕМ НАВИГАЦИИ, НАВЕДЕНИЯ И УПРАВЛЕНИЯ ЛЕТАТЕЛЬНЫМИ АППАРАТАМИ
- МОДЕЛИРОВАНИЕ АВИАЦИОННОГО ВООРУЖЕНИЯ
- ОПЕРАЦИОННОЕ МОДЕЛИРОВАНИЕ АВИАЦИОННЫХ СИСТЕМ
- МОДЕЛИРОВАНИЕ БОРТОВЫХ СИСТЕМ ЛЕТАТЕЛЬНЫХ АППАРАТОВ
- ФИЗИЧЕСКОЕ МОДЕЛИРОВАНИЕ ВНЕШНИХ ФАКТОРОВ ДЛЯ РЕШЕНИЯ АВИАЦИОННЫХ ЗАДАЧ НАДЕЖНОСТИ, ПРОЧНОСТИ И БЕЗОПАСНОСТИ
- МОДЕЛИРОВАНИЕ АВИАЦИОННЫХ СИСТЕМ ТЕХНИЧЕСКОГО ЗРЕНИЯ
- МОДЕЛИРОВАНИЕ БОРТОВЫХ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ
- МОДЕЛИРОВАНИЕ И ЛЕТНЫЕ ИСПЫТАНИЯ АВИАЦИОННЫХ СИСТЕМ
- МОДЕЛИРОВАНИЕ ИНЕРЦИАЛЬНЫХ ДАТЧИКОВ И СИСТЕМ

Контактная информация:

- Сергеев Сергей Александрович, тел.: (499) 157-73-26, e-mail: ssa@gosniias.ru
- Бабиченко Андрей Викторович, тел.: (496) 46-16-0-16, e-mail: ABabichehko@rpkb.ru
- Люшинский Анатолий Владимирович, тел.: (496) 46-3-47-52, e-mail: ALushinskiy@rpkb.ru

Рисунок к статье **Д. К. Левоневского, А. И. Мотиенко**
«МОДЕЛИРОВАНИЕ И АВТОМАТИЗАЦИЯ ПРОЦЕССОВ СБОРА И ОБРАБОТКИ ДАННЫХ В УМНОЙ МЕДИЦИНСКОЙ ПАЛАТЕ»

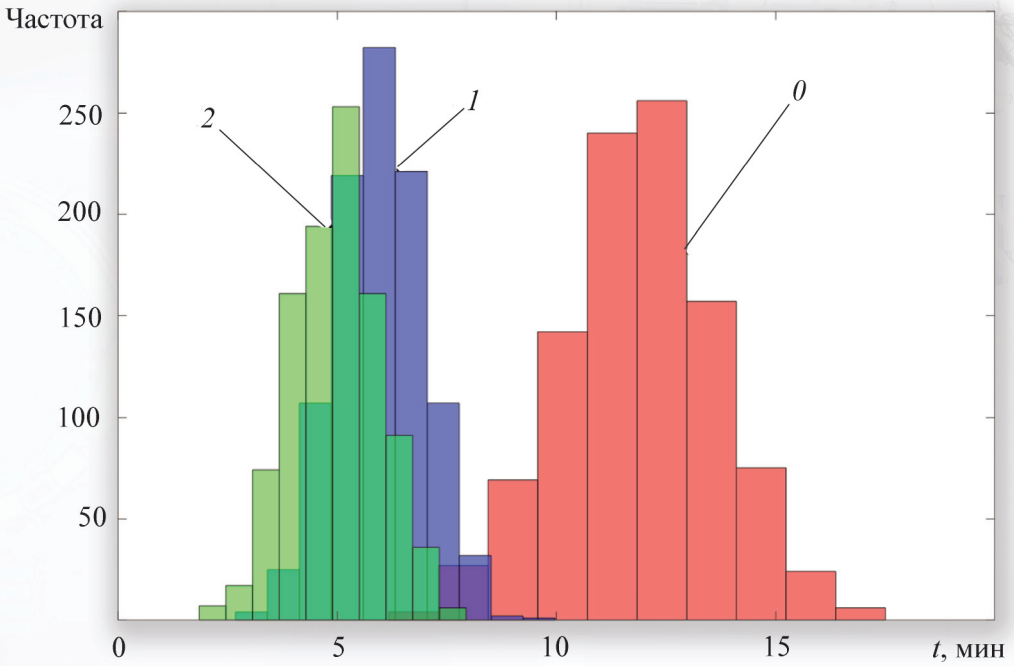


Рис. 4. Оценка времени сбора и обработки данных при разных уровнях автоматизации

Рисунок к статье **Р. М. Хусаинова, Н. Г. Талипова, А. С. Катасёва, Д. В. Шалаевой**
«НЕЙРОСЕТЕВАЯ ТЕХНОЛОГИЯ АНАЛИЗА ТРАНСПОРТНЫХ ПОТОКОВ В АВТОМАТИЗИРОВАННЫХ СИСТЕМАХ УПРАВЛЕНИЯ ДОРОЖНЫМ ДВИЖЕНИЕМ»

Рис. 3. Результаты выполнения задач обнаружения и отслеживания объектов

Рисунки к статье **Р. М. Хусаинова, Н. Г. Талипова, А. С. Катасёва, Д. В. Шалаевой**
«НЕЙРОСЕТЕВАЯ ТЕХНОЛОГИЯ АНАЛИЗА ТРАНСПОРТНЫХ ПОТОКОВ В АВТОМАТИЗИРОВАННЫХ СИСТЕМАХ УПРАВЛЕНИЯ ДОРОЖНЫМ ДВИЖЕНИЕМ»

Рис. 4. Результат выполнения задачи определения инцидента ДТП

Рис. 5. Результат выполнения задачи определения инцидента остановка

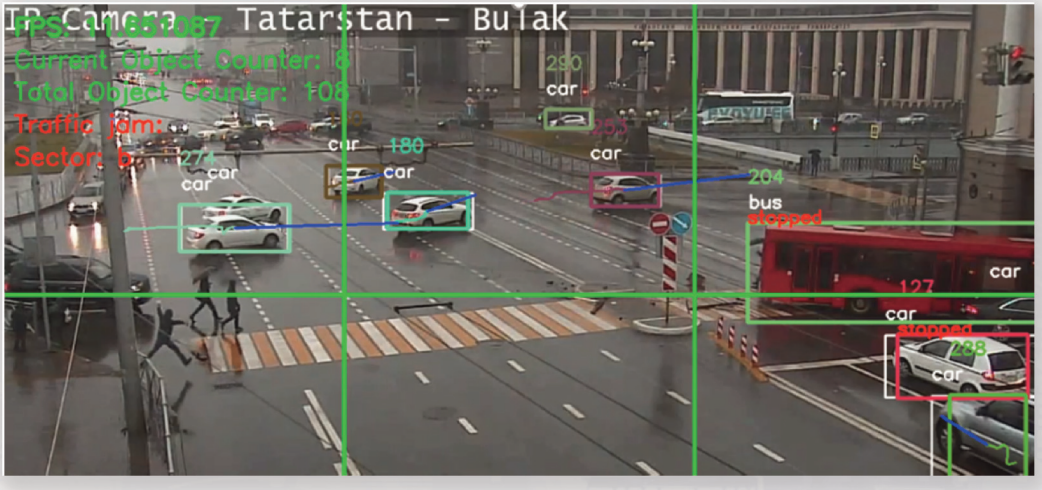


Рис. 6. Результат выполнения задачи определения инцидента пробка

ISSN 2220-3397. Программная инженерия. 2023. Том 14. № 10. 469–520. Подписной индекс по Объединенному каталогу «Пресса России» – 22765