ИСП РАН

ТРУДЫ ИНСТИТУТА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ РАН

PROCEEDINGS OF THE INSTITUTE FOR SYSTEM PROGRAMMING OF THE RAS

ISSN PRINT
2079-8156
ТОМ 31
ВЫПУСК 1

ISSN ONLINE
2220-6426
VOLUME 31
ISSUE 1

ИНСТИТУТ СИСТЕМНОГО ПРОГРАММИРОВАНИЯ
ИМ. В.П. ИВАННИКОВА РАН

МОСКВА, 2019

ISSN 2079-8156 (Print), ISSN 2220-6426 (Online)

Труды Института системного программирования РАН
Proceedings of the Institute for System Programming of the RAS

Труды ИСП РАН – это издание с двойной анонимной системой рецензирования, публикующее научные статьи, относящиеся ко всем областях системного программирования, технологий программирования и вычислительной техники. Целью издания является формирование научно-информационной среды в этих областях путем публикации высококачественных статей в открытом доступе.

Издание предназначено для исследователей, студентов и аспирантов, а также практиков. Оно охватывает широкий спектр тем, включая, в частности, следующие:

- операционные системы;
- компиляторные технологии;
- базы данных и информационные системы;
- параллельные и распределенные системы;
- автоматизированная разработка программ;
- верификация, валидация и тестирование;
- статический и динамический анализ;
- защита и обеспечение безопасности ПО;
- компьютерные алгоритмы;
- искусственный интеллект.

Журнал издается по одному тому в год, шесть выпусков в каждом томе.

Поддерживается открытый доступ к содержанию издания, обеспечивая доступность результатов исследований для общественности и поддерживая глобальный обмен знаниями.

Труды ИСП РАН реферируются и/или индексируются в:

Proceedings of ISP RAS are a double-blind peer-reviewed journal publishing scientific articles in the areas of system programming, software engineering, and computer science. The journal's goal is to develop a respected network of knowledge in the mentioned above areas by publishing high quality articles on open access.

The journal is intended for researchers, students, and practitioners. It covers a wide variety of topics including (but not limited to):

- Operating Systems.
- Compiler Technology.
- Databases and Information Systems.
- Parallel and Distributed Systems.
- Software Engineering.
- Software Modeling and Design Tools.
- Verification, Validation, and Testing.
- Static and Dynamic Analysis.
- Software Safety and Security.
- Computer Algorithms.
- Artificial Intelligence.

The journal is published one volume per year, six issues in each volume.

Open access to the journal content allows to provide public access to the research results and to support global exchange of knowledge. **Proceedings of ISP RAS** is abstracted and/or indexed in:

УДК004.45

Адрес: 109004, г. Москва, ул. А. Солженицына, дом 25.
Телефон: +7(495) 912-44-25
E-mail: info-isp@ispras.ru
Сайт: http://www.ispras.ru/proceedings/

Address: 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
Tel: +7(495) 912-44-25
E-mail: info-isp@ispras.ru
Web: http://www.ispras.ru/en/proceedings

С о д е р ж а н и е

Table of Contents

Моделирование прикладных и информационных систем из готовых сервисных ресурсов Интернет[1]

[1,2] *Лаврищева Е.М. <lavr@ispras.ru>*
[1] *Мутилин В.С. <mutilin@ispras.ru>*
[1,3] *Козин С.В. <kozyyy@yandex.ru>*
[1] *А.Г. Рыжов <ryzhov@ispras.ru>*

[1]*Институт системного программирования им. В.П. Иванникова РАН,*
109004, Россия, Москва, ул. А. Солженицына, д. 25
[2]*Московский физико–технический институт,*
141700, Россия, Московская обл., г. Долгопрудный, Институтский пер., д. 9.
[3]*Национальный исследовательский университет Высшая школа экономики,*
101000, Россия, г. Москва, ул. Мясницкая, д. 20

Аннотация. Рассматривается подход к созданию сложных (информационных и прикладных) систем из готовых ресурсов (модулей, компонентов, КПИ, сервисов, reuses и др.). В основе подхода создания систем, веб–систем лежит компонентная модель (КМ), включающая функциональные, системные, сервисные и интерфейсные готовые ресурсы (ГОР), и алгебра компонентов для выполнения разных операций над ГОР. Прикладные функции компонентов, КПИ описываются в языках программирования (ЯП), интерфейсы в языках IDL и WSDL, а сервисные компоненты создаются в SOA, SCA IBMSphere или выбираются из Интернет библиотек, как готовые. Исходные компоненты верифицируются и сохранятся в репозитории ГОР и интефейсов. Представлен метод сборки ГОР: Link в среде IBM, MS.VS; make в BSD, config в JavaEE; стандарт IEEE 828–96–2012 (Configuration), как итог всех сборок. ГОР тестируются на множестве тестовых данных, проверяется их правильность и надежность работы. На готовый сконфигурированный продукт формируется сертификат качества на основе стандартов ISO/IEC 9000 (1–4) quality. Даны перспективы развития средств обеспечения безопасности и качества веб–систем.

Ключевые слова: компонент; сервис; ресурс; система; веб–система; сборка; надежность; качество.

DOI: 10.15514/ISPRAS–2019–31(1)–1

Для цитирования: Лаврищева Е.М., Мутилин В.С., Козин С.В., Рыжов А.В. Моделирование прикладных и информационных систем из готовых сервисных ресурсов Интернет. Труды ИСП РАН, том 31, вып. 1, 2019 г., стр. 7–24. DOI: 10.15514/ISPRAS–2019–31(1)–1

1. Введение

В 70–80 годы прошлого столетия сформировался подход к сборке разнородных модулей в сложную систему с использованием интерфейсов при реализации комплексов программ специального и общего назначения в рамках ВПК (под руководством Липаева В.В.) и он получил название – сборочного программирования [1–3]. Метод сборки был представлен в ОС ЕС (IBM–360) и в других средах – Sun Microsystems, MS VS .Net, IBM WebSphere и др. По словам А.П. Ершова [4], «сборочное программирование обеспечивает построение уже существующих (проверенных на правильность) готовых отдельных фрагментов программ (типа reuses) в сложную структуру». Для описания интерфейсов использовался простой

[1] Работа выполняется по проекту РФФИ № 16-01-00352

язык MIL, затем после 90–х появились стандартные языки IDL, WSDL и др., а также средства сборки link и make BSD (1996), .Net, Java, SPAROL и др. [5–10]. Оператор make обеспечивают сборку исполняемых модулей из Filemake библиотек и задает им порядок связей друг с другом в средах ОС Linux, Java, .NET и др. Метод сборки получил широкое развитие на фабриках программ (Д. Гринфилда (Jack Greenfield) и Г. Ленц (Herbert Lenz) – потоковая сборка, 2007; К. Чернетски (Krzysztof Czarnecki) и А. Айзенекер (Ulrich Eisenecker) – мультисборка, 2005; И. Бей (Ying Bai) – взаимодействие разноязыковых программ в разных средах, 2010; К. Пол (Klaus Pohl) – конвейерная сборка на ProductLine/ProductFamily, 2004 и др.). Создана теория фабрик для производства программ из КПИ [20–23], основанная на методе сборки (на статью [20] приходят запросы из Китая, Таиланда, Южной Кореи и др.).

В 2009 году (потом в 2012) процесс сборки (интеграции) готовых reuses был стандартизирован (IEEE 828–96–2012, Configuration) и позволяет получить конфигурационную структуру любой программной и информационной системы, элементы которой можно менять, добавлять, заменять и удалять. Затем делать новый вариант конфигурационной структуры системы.

2. Подход к моделированию систем из компонентов

Моделирование систем проводится с помощью компонентной модели (КМ), как составной модели ОКМ [11]. Компонент (Component, Comp, C) определяется как некоторая абстракция, включающая в себя интерфейсный раздел и артефакт описания функции. Он может иметь несколько реализаций в зависимости от операционной среды, модели данных, СУБД и др. [12]. Интерфейс определяет данные для связи одного компонента с другими. Компонент может иметь несколько интерфейсов и наследуется в виде классов в модели компонента или каркаса на любом ЯП (C, C++, Ruby, Basic, Java и др.). Компоненты могут быть:

1) программные, сервисные, системные и служебные;
2) серверные, клиентские и веб–серверные и веб–клиентские;
3) контейерные, патерные, программные, математически–ориентированные и др.

Каждый из этих типов компонентов имеет спецификацию, требования, правила их взаимодействия с другими компонентами, заданную в интерфейсе. Для компонентов разработана компонентная алгебра (внешняя, внутренняя и эволюционная), которая включает разные действия над компонентами (добавление, удаление, замена, верификация, тестирование и др.).

2.1 Алгебра операций над компонентами

Операция добавления компонента С к компонентой среде обозначается $\oplus$ и выполняется согласно правил

$C \oplus CE_1 = CE_2$,

$NameSpace = \{C.CName\} \cup CE_1.NameSpace$,

$CE_2.InRep = \{C.(CIn_i, CName)\} \cup CE_1.InRep$,

$CE_2.ImRep = \{C.(C_j, CName)\} \cup CE_1.ImRep\}$.

Аксиома 1. Операция добавления компонента к компонентной среде коммутативна и ассоциативна:

$C \oplus CE = CE \oplus C$; $C_1 \oplus (CE \oplus C_2) = (C_1 \oplus CE) \oplus C_2$.

Операция удаления компонента из среды обозначается знаком $\Diamond$ и имеет вид: $CE_1 \Diamond C = CE_2$.

Теорема 2. Для любого компонента *C* в среде *CE* выполняется равенство

$(C_2 \oplus CE) \Diamond C = CE$.

Операция замены компонента задается знаком "–" и имеет вид:

$CE.NameSpace(C_1) - C_2 = (CE \lozenge C_1) \oplus C_2$.

Операция объединения ($\cup$) компонентных сред имеет вид:

$CE_1 \cup CE_2 = CE_3$,

Теорема 3. Операция объединения компонентных сред ассоциативна*:*

$(CE_1 \cup CE_2) \cup CE_3 = CE_1 \cup (CE_2 \cup CE_3)$.

Аксиома 3. Операция объединения компонентных сред коммутативна:

$CE_1 \cup CE_2 = CE_2 \cup CE_1$.

Утверждение 2. Для любых компонентных сред выполняется равенство:

$CE \cup FW = FW \cup CE$.

Утверждение 3. Для двух компонентных сред CE_1 , CE_2, и компонента *Comp* всегда выполняется:

$Comp \oplus (CE_1 \cup CE_2) = (Comp \oplus CE_1) \cup CE_2 = (Comp \oplus CE_2) \cup CE_1$.

Операция добавления интерфейса и реализации компонентов

Операция добавления *addImp Comp* нового входного интерфейса и реализации компонента Comp в среду или в модель:

NewComp = AddImp(OldComp, NewCImps, NewCIntOs) и

NewInt = OldInt $\cup$ *NewIntOs,* где

NewCImp = OldCImp $\cup$ {*NewImps*}($\exists$ *OldIntt* $\in$ *OldCIntI*) *Provide*(*OldInt*) $\subseteq$ *NewImps.* где

NewCImps – новая реализация, которая добавляется;

OldCInt – множество существующих интерфейсов *Int.*

Условие целостности компонента выполняется автоматически при условии использования прежних входных интерфейсов. Из целостности старого компонента вытекает целостность нового компонента.

2.2 Компонентная алгебра

Компонентная алгебра – это внешняя, внутренняя и эволюционная алгебры**:**

$\sum = \{\varphi_1, \varphi_2, \varphi_3\}$, где

$\varphi_1 = \{CSet, CESet, \Omega_1\}$ – внешняя алгебра;

$\varphi_2 = \{CSet, CESet, \Omega_2\}$ – внутренняя алгебра;

$\varphi_3 = \{Set, CESet, \Omega_3\}$ – алгебра эволюции.

Внешняя алгебра $\varphi_1 = \{CSet, CESet, \Omega_1\}$включает операции:

- инсталляция $CSet_2 = Cset \oplus CESet_1$;
- объединение компонентных сред $CESet_3 = CESet_1 \cup CESet_2$;
- удаление компонента из компонентной среды $CESet_2 = CESet_1 \setminus CSet$.

Внутренняя алгебра $\varphi_2 = \{CSet, CESet, \Omega_2\}$ включает операции:

- add imp добавление реализации;
- addInt – добавление интерфейса;
- replImp – замена
- реализации компонента;
- replInt – замена интерфейса компонента.

Алгебра эволюции $\varphi_3 = \{Set, CESet, \Omega_3\}$ включает операции:

- рефакторинга; реинженеринга с заменой;
- переименования компонентов и интерфейсов и добавления новых компонентов в ПС;
- реверсной инженерии – восстановления исходной структуры компонента по выходному коду ПС.

Приведенные алгебры позволяют гибко управлять компонентами в моделируемой системе.

3. Проектирование и верификация моделей систем

Построение архитектуры системы на компонентной основе проводится с помощью графовой модели метода ОКМ, создаваемой с помощью логико–математического аппарата на четырех уровнях:

- обобщающий уровень, который задает объект в виде денотата в соответствии с теории Фреге типа <имя объекта> <концепт>;
- структурный уровень для теоретико–множественного упорядочения объектов и представления их в виде графа; могут применяться операции объединения, пересечения, разности и т.п.;
- характеристический уровень для логико–алгебраического определения характеристик объектов и их свойств;
- поведенческий уровень для определения поведения и взаимосвязей между объектами графа ОМ.

3.1 Графовая модель

На первых двух уровнях выделяются объекты O = (O_i, O_2, …, O_n) или функции предметной области (ПрО) и создается граф G, в вершинах которого размещаются O_i, а дуги задают их связи (рис.1).

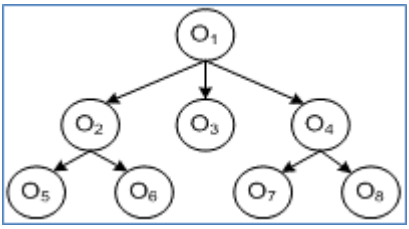


Рис.1. Граф из объектов ПрО
Fig. 1. Graph of domain objects

Свойства графа G:

- для каждой вершины Oi существует хотя бы одна связь (структурная) с другой вершиной графа (стрелки);
- существует лишь одна вершина O1 графа G, которая имеет статус множества объектов, отображающего ПрО в целом.

Графу G соответствует объектная модель ОМ = (O_1, O_2, ..O_7). Объекты модели проверяются с помощью model checking на соответствие спецификации требованиям и свойствам (рис.2).

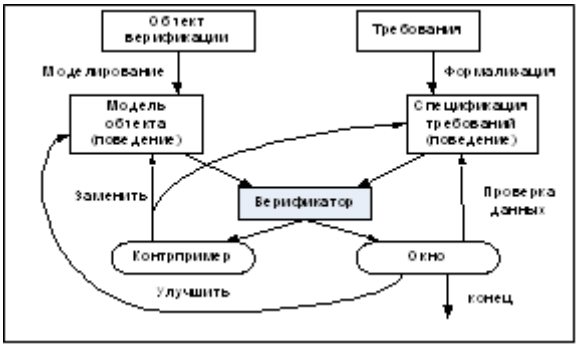


Рис.2. Верификация модели ОМ
Fig.2. Object model verification

Если соответствие удовлетворяется, то верификатор сообщает о правильности модели, в противном случае дается пояснение о возникшем несоответствии.

На следующих уровнях моделирования системы уточняется функции и интерфейсы. В результате в граф G добавляются интерфейсов связи объектов друг с другом $\boldsymbol{I}_{\text{Oi}} = \{Io_{n5}, I_{o6}, I_{o7}, I_{o8}\}$ (рис.3.).

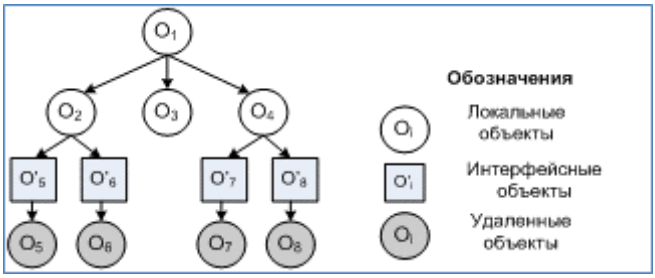


Рис. 3. Граф G с объектами и интерфейсами
Fig. 3. Graph G with objects and interfaces

В графе *G* заданы:

- O1, O2, O3, O4 , O5 , O6 , O7 , O8 – функциональные объекты;
- O′5, O′6, O′7, O′8, – интерфейсные объекты, которые размещаются в репозитории, а дуги соответствуют связям между всеми видами объектов.

Элементы графа $O_1 – O_8$ описываются в ЯП (Fortran, Basic, C, C++, Python и др.), а интерфейсные объекты $O'_5 – O'_8$ в языке IDL/ WSDL. По графу G можно построить программы $P_0 – P_5$ с использованием операторов сборки *link* :

$P_0 = (P_1 \cup P_2 \cup P_3 \cup P_4 \cup P_5)$.

$P_1 = O_2 \cup O_5$, *link* $P_1 = In\ O'_5\ (O_2 \cup O_5)$:

$P_2 = O_2 \cup O_6$, *link* $P_2 = In\ O'_6\ (O_2 \cup O_8)$:

P_3:

$P_4 = O_4 \cup O_7$, *link* $P_4 = In\ O'_7\ (O_4 \cup O_7)$:

$P_5 = O_4 \cup O_8$, *link* $P_4 = In\ O'_8\ (O_4 \cup O_8)$:

На основе этого графа определяется модель системы (sys) в виде:

- M_{sys} = < Mf, Ms, Mi, Md>, где
- M_f = (fO1, fO2,…, fOr) – множество функций объектов ПрО;
- M_s = (Sin, Sout, Sinoutt) – множество входных данных Sin, выходных данных Sout и промежуточных данных Sinoutt, определяемых на множестве системных сервисов (Common Facility Services) операционной среды;
- M_i = (Io1, Io2, …, Ion) – множество интерфейсно–компонентных элементов для функциональных элементов fOn входные in, выходные параметры out и inout из множества M_s;
- M_d = (Md1, Md2, …, Mdn) множество данных и метаданных ПрО, с которыми работает система.

Множества M_{sys}, M_s и M_i определяют ГОР, интерфейсы и общие данные. Они могут иметь пересечение по данным, которые относятся к *in, out, inout* и входят в состав внешних типов данных, которыми обмениваются элементы модели через сервер Application.

Модели проверяются и на их основе конфигурируются отдельные версии вариантов выходной системы. Для каждой пары отдельных функциональных и интерфейсных элементов может быть задана точка их изменения для замены более корректным или функционально правильным компонентом.

Системные компоненты управляют оборудованием, ОС и серверами. В состав системы входят клиент, сервер или Интернет браузер. На веб–сервере происходит обработка запросов на выполнение разных операций над программами и данными.

3.1 Модели системных и прикладных сервисов клиент–серверной архитектуры

Основу сервисов составляет клиент–серверная архитектура Интернет, которая первоначально была реализована в CORBA [13, 16, 17] и включает:

- брокер объектных запросов (*Object Request Broker* – ORB) для взаимодействия клиент–объектов с сервер–объектами на ЯП (Smalltalk, Cobol, Ada–95, Lisp, PL/1, C++, Python, Java, IDLScript и др.);
- общие объектные сервисы (*Common Object Services* – COS) для управления изменениями, реализациями, транзакциями, подпроцессами и т.п.;
- общие средства обслуживания (*Common Facilities* – CF) для объединения в различные конфигурации сервисных объектов;
- объектные приложения (*Application Objects* – AO), над которыми могут производиться операции – открыть, инсталлировать, переместить, поместить, выполнить.

Объект–клиент и объект–сервер обмениваются между собой с помощью запросов, каждый из которых обрабатывается брокером ORB на основе описания интерфейсов объектов для клиента, сервера и ядра ORB.

Интерфейс клиента (*Client Interface*) обеспечивает взаимодействие с объектом–сервером и состоит из трех базовых интерфейсов:

- stub–интерфейса, содержащего описание внешних параметров и операций объекта в IDL;
- интерфейса динамического вызова (Dynamic Invocation Interface – DII) объекта, определяемого во время выполнения программы клиента при поиске интерфейса в репозитории интерфейсов или в репозитории реализаций;
- интерфейса сервисов ORB (ORB Services Interface), содержащего набор сервисных функций, которые клиент запрашивает у сервера через брокер ORB.

Stub–интерфейс обеспечивает взаимосвязь клиента с ORB через *stub* и посылает параметры серверу в запросе. Схема взаимодействия клиента с объектом соответствует схеме вызова RPC удаленных процедур.

Интефейс DII обеспечивает доступ объектов и их интерфейсов во время выполнения. В каждом вызове указывается тип объекта, тип запроса и параметры. По этой информации извлекается соответствующая программа из репозитория интерфейсов и репозитория реализаций [18].

Описание интерфейса в IDL начинается с ключевого слова **interface**, за которым следует: имя интерфейса, описание типов параметров и операций вызова объектов. Описание типов данных начинается ключевым словом typedef, за которым следует базовый или конструируемый тип и его идентификатор. В качестве константы может быть некоторое значение типа данного или выражение, составленное из констант. Типы данных и константы описываются как фундаментальные типы данных ЯП: integer, boolean, string, float, char и др.

3.2.1 Генерация системных программ клиента и сервера

Генерируются два класса системных компонентов: клиент; сервер.

В клиент помещаются механизмы коммуникации между программными компонентами. Пользователь этого класса не должен ничего менять, а использовать в таком виде:

```
…
int param;
…
```

```
ClientInterface ci = new ClientInterface();
int result = ci.func(param);
…
```

Здесь `ClientInterface` – это сгенерированный класс, который содержит механизмы, необходимые для передачи данных на сервер, который предоставляет сервис `int func (int param)`.

Если сгенерированный класс – сервер, то кроме механизмов связи с клиентами в него помещают пустые функции, отвечающие задекларированным методам, описанным в интерфейсе. Для использования сервера пользователю необходимо реализовать конкретную логику этих методов следующим образом:

```
class ServerInterface {
  …
  public int func (int param){
    …//реализация метода
  }
…
void main(){
  …
  ServerInterface si = new ServerInterface();
  si.work();
  …}
```

3.2.2 Современная клиент–серверная архитектура в Интернет

В связи с большим количеством пользователей Интернет и большими объемами данных (типа Big Data), современная клиент–серверная архитектура включает больше серверов, в том числе и серверов Баз Данных. Они способствуют увеличению общей производительности веб–систем путем вертикального и горизонтального масштабирования. Вертикальное масштабирование увеличивает производительность за счет добавления аппаратных ресурсов серверов, а горизонтальное масштабирование способствует росту производительности систем за счет увеличения количества серверов и БД. Это масштабирование основываться на процессах: *Front–end* для клиентских систем и *Back–end* для обслуживания серверных частей приложений. Эти процессы должны обеспечивать безопасность, защиту и качественную работу в Интернет [28].

3.3 Средства тестирования систем

Тестирование системы, созданной методом сборки, проводится с помощью набора тестов для отдельных элементов и системы в целом. Тесты проверяют функциональные и интерфейсные компоненты. В качестве инструмента тестирования можно использовать фреймворк Visual Studio 2007 со средствами проверки правильности тестирования разных видов объектов. В него входит компонент *Test Manager*, который управляет средствами планирования процесса тестирования и выполнения тестовых сценариев. При обнаружении ошибок в процессе тестирования вносятся исправления в модели системы M_{sys}. Затем проводится повторная операция config для получения готового продукта с добавлением новых объектов или удаления старых.

4. Проектирование систем из сервисных компонентов

В настоящее время в Интернет накоплено огромное количество ГОР сервисно–компонентного типа, которые представляются средствами моделей SOA (Service Oriented Architecture) и SCA (Service Component Architecture). Интерфейс сервисных компонентов задается в языке WSDL [13 –15]. Элементы SOA задают описание некоторой функции (Function) с заданным качеством сервиса (Quality service) в IT–стандартах комитета W3C.

Средствами модели SCA задаются сервисные компоненты, как ГОР типа reusable. К ним относится EJB сервер приложений J2EE, сетевые сервисы, объекты планирования, доступа к БД и к системе. Элементы архитектуры SCA могут собираться в систему путем интеграции или конфигурационной сборки. Данные модели SOA и SCA используются нами при моделировании систем и веб–систем. При проектировании веб—систем создаются:

- модель системы M_{wsys};
- клиент–серверная архитектура с веб–сервером и веб–клиентом для выполнения запросов от M_{wsys};
- спецификации интерфейсов компонентов I_{on};
- схемы запросов к имени функции или компонента с помощью операторов вызова или протоколов связи стандарта ISO/IEC;
- сервисно–компонентные элементы для сред (J2EE, .Net, Appach, JAVA, SOAP, WSDL и др.) [15] и их накопление в библиотеках системных сред Microsoft VS .Net,, IBM, Intel, Linux. Semantic Web Интернет;
- данные для верификации и тестирования сервисных компонентов.

4.1 Создание сервисных компонентов в SOA

Модель SOA – это набор принципов и средств создания системного ПО и прикладных программных систем с помощью совместимых и унифицированных сервисов Интернет [13–15]. SOA задает реализацию сервисов на серверной стороне с открытым интерфейсом с описанием типов входных/выходных параметров в языке WSDL и портов обмена метаданными (Metadata Exchange Endpoints). WSDL–компилятор текст описания в этом языке готовит для сервера и клиента в виде прокси–классов. SOA обеспечивает согласованность, языковую независимость и интероперабельность серверной и клиентской частей системы. От разработчика требуется написать сервис средствами WCF (Window Communication Foundation) и использовать его в ЯП (Java, Python, Ruby и др.).

Фундаментом сетевых служб SOA являются:

- набор языков – XML, SOAP, UDDI, WSDL, BPREL, BPMN и др. для реализации базовых свойств сетевых сервисов, обеспечения их взаимодействия между собой в соответствующих средах (SOA, SCA и др.);
- поставщик услуги, который осуществляет ее реализацию в виде сетевой службы, прием и выполнение запросов пользователей, а также публикацию сведений о сервисе в соответствующем реестре;
- реестр (каталог) служб содержит библиотеку сервисов, а также средства их поиска и вызова с помощью запросов, которые поступают от поставщиков сервисов на получение сервисов.

Клиент осуществляет поиск и вызов необходимого сервиса из реестра описания сервисов в соответствии с заданным интерфейсом.

Посредником между этими сервисами и системой является *провайдер* клиент–серверной архитектуры, который обеспечивает взаимодействие между поставщиками и провайдерами.

4.2 Создание сервисно–компонентных ГОР в SCA (IBM WebSphere)

SCA [21] предназначена для работы с прикладными компонентами с разными спецификациями и включает: EJB сервер приложений J2EE компании Sun Microsystems, который работает с сетевыми сервисами, компонентами доступа к БД и к ИС предприятия (Enterprise Information System, EIS) и др. SCA обеспечивает доступ к сервисным компонентам и определяет зависимости между ними через аппарат ссылок. Компоненты SCA в IBM WebSphere Integration Developer (WebSphereID) упаковываются в модуль для выполнения сервисного модуля с WebSphere Process Server – эквивалентного EAR–файлу J2EE и некоторым другим. Подмодули J2EE и артефакты упаковываются с модулем SCA,

что позволяет запустить сервис и передавать данные для обработки и интеграции. Система Dynamic Profiles WebSphere Portlet Factory обеспечивает динамическую конфигурацию пользовательского интерфейса, безопасность и другие службы.

Сервер каталогов Tivoli (Tivoli Directory Server) обеспечивает протокол доступа к каталогам LDAP (Lightweight Directory Access Protocol) для управления идентификацией. Реестр WebSphere Service Registry and Repository позволяет провайдерам регистрироваться, а клиентам – выбирать сервисы.

Сервисно–компонетная модель SCM представляет собой обобщение объектно–компонентной модели продуктов (СПП, [18]). В ней каждый программный элемент содержит удаленные компоненты типа reuses – КПИ, которые обмениваются гетерогенными данными и обеспечивают выполнение системы. При этом используются механизмы сервисных объектов данных SDO и сервисы доступа DAS.

Для описания сетевых сервисов язык WSDL предоставляет следующие виды описаний:

- строка (xsd:string),
- целые числа (xsd:int, xsd:long, xsd:short, xsd:integer, xsd:decimal),
- числа с плавающей запятой (xsd:float, xsd:double),
- логический тип (xsd:boolean),
- последовательность байтов (xsd:base64Binary, xsd:hexBinary),
- дата и время (xsd:time, xsd:date, xsd:g),
- объекты (xsd:anySimpleType).

Протокол Contract WorkFlow задает контракт–протокол для взаимодействия клиента и сервира. WCF содержат три вида контрактов:

1) сервисов для описания функциональных операций, реализованных сервисом. Внутри контракта сервиса имеются контракты об операциях, как отдельные операции сервиса, которые реализуют функции;
2) данных, определяющих формат данных, которыми будут обмениваться сервисы. Это относится как к запросу на сервис, так и к октету сервиса.
3) сообщений, как тип контракта, который используется для того, чтобы получить контроль над заголовком SOAP.

Пример описания сообщения в языке XML в WCF:

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="
http://www.cbsystematics.com">
<!Конверт протокола SOAP-> <env:Header>
<!-Заглавие протокола SOAP-> </env:Header> <env:Body>
<!-Тело протокола SOAP-> </env:Body> </env:Envelope>
```

При написании контрактов WCF атрибутами будут [ServiceContract], [OperationContract], [FaultContract], [MessageContract] и [DataContract].

На этапе выполнения клиента вызывается метод, определенный в интерфейсе сервиса, WCF сериализует типы CLR и вызов метода в формат XML и посылает сообщение в сеть для привязки к схеме кодировки в WSDL. Со стороны XML задается XSD–описание структуры данных и сообщение осуществляется лишь после того, как будет создан экземпляр XML (XML Instance). Со стороны .NET имеется тип CLR, который определяет структуры данных и функциональные возможности после того, как создан объект такого типа.

4.3 Средства Семантик Веб для разработки ГОР и систем

Для описания ГОР имеются такие средства [24, 25]:

- **RDF** стандарта W3C (2004) для описания сетевых, семантических ресурсов и метаданных (данные о данных). Служит каркасом для создания отдельных компонентов семантической паутины. RDFS (англ. *RDF Schema*) – это надстройка над RDF, которая позволяет создавать классы и свойства объектов.

- **OWL** (Web Ontology Language) построен на форматах RDF и RDFS, предназначен для описания онтологий, логики и согласуется с современными сетевыми стандартами.
- **SPARQL** (Protocol And RDF Query Language) – язык запросов для быстрого доступа к данным RDF для получения необходимой информации из сети.
- **RIF** – формат обмена правилами (Rule Interchange Format) и др.
- **WSDL** – язык описания входных и выходных данных для описания запросов Интернет на сервисы и включает языки:
  - **WSCI** (Web Services Choreography Interface [23]),
  - **WSCL** (Web Services Conversation Language [24]),
  - **BPMN** (Business process and model and notation [25]),
  - **BPEL** (Business Process Execution Language for Web Services [26]) и другие.

В качестве адреса объектов в сети используются универсальные идентификаторы ресурсов URI (Uniform Resource Identifier) и интерфейс, задаваемый для управления связями с другими сервисами через XML–документы.

4.4 Сборка сервисных ГОР веб–системы в Интернет

Для сборки сервисов используется инструмент – ***Jopera for Eclipse*** (http://www.jopera.ethz.ch/), который обеспечивает [18, 23– 25]:

- композицию сервисов (типа Agile) и визуальный мониторинг отладки композиций сервисов;
- управление изменением интерфейсов сервиса с помощью сообщений об изменениях в сервисе Jopera;
- масштабируемость и автономное исполнение процесса запуска систем с помощью сообщений Jopera.

Jopera предоставляет набор Eclipse–плагинов для связи различных программных элементов и допускает итеративную композицию сервисов (через маршрутизаторы SOAP и RESTful Web–сервис, Grid–сервисы, Java snipets и др.), а также путем моделирования и исполнения процессов в сети. Для поиска сервисов по их семантическим описаниям используются *Feta Client* и *Feta Engine*.

Feta Client – это GUI–плагин системы Интернет Taverna, используемый для описания сервиса, а *Feta Engine* для задания Web–сервиса.

Подключаясь к *Feta Engine*, плагин *Taverna Feta* позволяет:

- конструировать ориентированные на заданную предметную область семантические запросы к нужным сервисам, которые затем отсылаются на *Feta Engine*;
- отображать информацию о результатах выполнения запроса на поиск сервисов;
- интегрировать результирующие сервисы в систему WorkFlow.

Критериями поиска сервисов являются:

- сервис, на входе которого находится элемент семантического или общего типа X;
- сервис, который производится на выходе системы и выдает элемент семантического типа Y;
- сервис, который решает задачу X или еще более конкретную;
- сервис, который использует метод X и более конкретный;
- сервис процессора WSDL и др.
- сетевые сервисы стандартной модели OSI, SOA, SCA, как инструменты представления и обработки ресурсов в сети Интернет для реализации деловых, финансовых, экономических и других услуг при решении разных прикладных задач.

4.5 Конфигурация ресурсов веб–систем

Под *конфигурацией* системы понимается структура некоторой ее версии, включающая функции, объединенные между собой операциями связи с параметрами, задающими

режимы функционирования системы [16–18]. Версия или конфигурация системы согласно IEEE Standard 828–2012 (Configuration) включает:

- базис конфигурации – BC (Configuration Baseline);
- элементы конфигурации (Configuration Item);
- компоненты, ГОР, входящие в описание моделей Msys, Mwsys;

Управление конфигурацией (Configuration Management) заключается в наблюдении за модификацией параметров конфигурации и компонентов системы, а также в проведении систематического контроля, учета и аудита внесенных изменений, поддержки целостности и работоспособности системы.

Управление конфигурацией исходя из стандарта состоит в выполнении следующих задач:

1) идентификация конфигурации (Configuration Identification).
2) контроль конфигурации (Configuration Control).
3) учет статуса конфигурации (Configuration Status Accounting).
4) фудит конфигурации аудит (Configuration Audit).
5) трассировка изменений конфигурации на этапах сопровождения и эксплуатации системы;
6) верификация компонентных сервисов по моделями M_{sys}, M_{wsys};
7) доказательство изоморфного отображения данных компонентных сервисов с типами данных стандарта ISO/IEC 11404 Object Data Types –2007.

При конфигурационной сборке ГОР используется модели систем M_{sys}, M_{wsys} и модель характеристик MF (Model Feature). ГОР и КПИ накапливаются в репозиториях или библиотеках системы. Они отбираются, адаптируются и интегрируются в единую систему. Основную роль в этих процессах выполняет конфигуратор ИТК (http://7dragons.ru/ru). Он обеспечивает сборку разнородных ГОР и их интерфейсов с вариантами отдельных готовых продуктов, которые находятся в репозиториях.

М*одель среды конфигуратора* включает:

- описание графовой схемы системы из ГОР;
- модели вариантов системы;
- конфигурационную модель ГОР и КПИ;
- операцию конфигурационной сборки КПИ и ГОР;
- аудит конфигурации;
- верификатор моделей и ГОР.
- оценку качества ГОР.

Конфигуратор собирает требуемые ГОР и КПИ по моделям в веб–систему с учетом их интерфейсов по заданным моделям и формирует конфигурационный файл системы, который выполняется в соответствующей операционной среде.

5. Моделирование варианта ОС для прикладных систем

Выше рассмотрены подходы к созданию систем и веб–систем из ГОР, сделанных другими разработчиками для выполнения разных функций математического, системного и сервисного типа. В данном разделе рассматриваются подход к процессам создания варианта ОС Linux, который будет управлять некоторой новой прикладной областью (например, медицина, биология, геология и др.). Разработан процесс создания варианта ОС в виде экспериментального варианта ядра ОС [19, 30–33]. Ниже дается описание.

5.1 Процесс определения варианта ядра ОС Linux

Для определения варианта ОС необходимо выполнить 7 процессов следующего содержания [19, 30–33].

1) **Конфигурирование.** Во время сборки Linux проверяет файл конфигурации kconfig на наличие рекурсивных зависимостей и несуществующих переменных в коде

2) **Поиск «мертвого кода»,** т.е. такого кода, в котором контроль не передается ни при каких обстоятельствах.
3) **Препроцессирование** включает предварительную обработку элемента функции из ядра ОС непосредственно перед компиляцией и получения кода для окончательной версии программы. Если находятся ошибки, то выдается информация о них.
4) **Компиляция** состоит в выдаче кода или случайных ошибок при обнаружении необъявленной переменной / функции, отсутствие пунктуации в коде и т. д.
5) **Связывание** отдельных элементов ядра выполняет оператор Link. Он находит ошибки задания связи компонентов в интерфейсе или ошибки вызова компонентов из внешних библиотек.
6) **Обработка** ошибочных ситуаций.
7) **Тестирование** собранного варианта ОС. Обработка ошибок проводится с помощью LDV и CPAchecker. Инструмент LDV ставит метки кода в соответствии с заданными правилами, CPAchecker проверяет доступность меток и правильность выполнимости компонентов.

5.2 Конфигурация варианта ядра ОС Linux

На основе анализа ядра ОС Linux установлено, что оно содержит более 10 000 переменных и большое количество функциональных и системных компонентов для обработки разного рода заданий по функционированию любых прикладных систем. Создание некоторого варианта ОС для класса прикладных систем (например, для медицины, биологии и др.) требует выбора из множества компонентов ОС наиболее подходящих для оперативного управления прикладными системами. Из выбранных компонентов ОС создается модель *MF* с базовыми характеристиками компонентов ОС и модель системы *Msys* варианта ОС, включающая множество функциональных M_f, интерфейсных M_{io} и компонентов M_d работы с данными. Эти компоненты тестируются на правильность их идентификации с помощью наборов тестов и операций установления связей с соответствующими компонентами других множеств. После тестирования проводится операция *config* (M_{fi}, MI_{oi}, M_{di}) для получения конфигурационного файла варианта ОС.

Процесс формирования варианта ОС включает [4–8] определение загружаемых модулей из библиотек сервисов, драйверов устройств и файловых систем и настройки параметров безопасности, защиты и криптографии.

Конфигурация ядра варианта ОС проводится с помощью этих параметров с учетом особенностей прикладной системы. Например, ядро может включать в себя множество опций безопасности исходя из стека SELinux Национального агентства по безопасности NSA, ориентированных на безопасность функциональных компонентов ОС.

Перед конфигурацией варианта системы ОС проверяется файл /etc /udev /rules.d/70–persistent–net.rules и определяются имена сетевых устройств, а также схема именования правилами Udev. Выходной файла начинается с блока комментариев, за которым следуют две строки для каждого сетевого адаптера. Первая строка сетевой карты – это описание и комментарии, показывающие идентификаторы оборудования и устройств согласно карты PCI и драйверов.

При задании имени интерфейса идентификатор аппаратного обеспечения не используются. Вторая строка – это правило Udev, которое соответствует сетевой карте с фактически присвоенным именем.

Некоторое программы ПО ОС могут устанавливаться позже с помощью символических ссылок /dev /cdrom и /dev/dvd для устройства CD–ROM или DVD–ROM. Кроме того, могут

помещаться символические ссылки в /etc/fstab. Udev в зависимости от возможностей каждого устройства.

Сценарий может работать в режиме «by–path» по умолчанию для устройств USB и FireWire, создаваемые правила которых зависят от физического пути к устройству CD или DVD. Сценарий может работать в режиме «по–id» (по умолчанию для устройств IDE и SCSI) и зависит от правил идентификационных строк, хранящихся на устройстве CD или DVD. Физический путь к устройству (порты и / или слоты) изменяются, например, при планировании перемещения диска в другой порт IDE или другой разъем USB режима «by–id».

Для каждого устройства находится соответствующий каталог в разделе / sys / class или / sys / block и для видеоустройств в разделе / sys / class / video4linux / videoX.

Интерфейсы сетевых сценариев задаются в файле / etc / sysconfig /. При этом файл ifconfig содержит настраиваемый интерфейс ifconfig.xyz в сетевой карте с именем eth0. Внутри этого файла содержатся атрибуты интерфейса IP–адрес, маски подсети и т. д. (Пример описания в сконфигурированного файла см. в [28].)

6. Обеспечение качества систем

Качество – это совокупность свойств (показателей качества) ПО, которые обеспечивают его способность удовлетворять потребности заказчика, в соответствии с его назначением. Оно задано стандарте ГОСТ 2844–98 с помощью модели качества и его показателей. Стандарт ISO/IEC 12207 определил основные процессы ЖЦ разработки ПС, но и организационные и дополнительные процессы, которые регламентируют планирование, управление качеством и оценку затрат на проект. На этапах ЖЦ проводится анализ качества ПО, ориентированные на:

- достижение качества ПО в соответствии с требованиями и критериями;
- верификацию и аттестацию (валидацию) промежуточных результатов ПО на этапах ЖЦ и измерение степени достижения отдельных его показателей;
- тестирование готовой ПС, сбор данных об отказах, дефектах и др. ошибках в системе и оценивание надежности по соответствующим моделям надежности.

Модель качества ПО согласно стандарту задает шесть показателей (характеристик) q_1–q_6 (q–quality) качества:

q_1 – функциональность (functionality),
q_2 – надежность (realibility),
q_3 – удобство (usability),
q_4 – эффективность (efficiency),
q_5 – сопровождаемость (maitainnability),
q_6 – переносимость (portability).

Каждая характеристика q_i рассчитывается по специальным формулам и метрикам стандарта. Надежность оценивается согласно полученных на процессе тестирования ошибок, дефектов и отказов в ПО и по разным моделям надежности (оценочным, измерительным и др.).

Данные по всем показателям качества q_1–q_6 оцениваются по окончательной формуле:

$$q_i = \sum_{j=1}^{k_i} a_{i_j} m_{i_j} w_{i_j}$$

где a_{i_j} – атрибуты каждого показателя качества; m_{i_j} – метрики каждого атрибута качества; w_{i_j} – вес каждого атрибута показателя качества системы. Полученные данные по характеристикам (показателям) модели качества входят в сертификат качества.

7. Заключение

Данный подход к сборке систем и веб–систем реализован в рамках проекта РФФИ №16–01–00352 «Теория и методы разработки изменяемых программных систем» [14]. В работе рассмотрены базовые понятия – ГОР, модели систем и конфигурационной сборки. Они специфицируются в языках (C++, JAVA, Python, Basic и др.). Описан компонентный метод ОКМ, основу которого составляет логико–математический аппарат проектирования и формирования модели системы и модели характеристик MF. Сформулированы модели характеристик масштабированных клиент–серверных архитектур при создании современных веб–систем [28]. Приведено описание новых открытых моделей SOA и SCA для описания сервисов, серверов и клиентов. Дано описание разных методов сборки систем из ГОР (link, make, config и др.). Приведен вариант веб–системы для прикладных систем. Обоснованы перспективы развития технологии программирования прикладных систем с обеспечением безопасности, надежности и качества в клиент–серверной архитектуре [30–33].

Список литературы

[1] Лаврищева Е.М., Грищенко В.Н. Сборочное программирование. Киев, Наук. думка, 1991, 236 стр.

[2] Липаев В.В., Позин Б.А., Штрик А.А. Технология сборочного программирования М.: Радио и связь, 1992, 324 стр.

[3] Лаврищева Е.М., Грищенко В.Н. Сборочное программирование. Основы индустрии программных продуктов. Киев, Наук. думка, 2009, 371 стр.

[4] Ершов А.П. Опыт интегрального подхода к актуальной проблеме ПО. Кибернетика, № 3, 1984, стр.11–21.

[5] Лаврищева Е. М., Карпов Л. Е., Томилин А. Н. Подходы к представлению научных знаний в Интернет науке. Сб. трудов XIX Всероссийский научной конференции «Научный сервис в сети Интернет», Новороссийск, 18–23 сентября 2017, стр. 310–326.

[6] Лаврищева Е. М., Карпов Л. Е., Томилин А. Н. Семантические ресурсы для разработки онтологии научной и инженерной предметных областей. Сб. трудов XVIII Всероссийской научной конференции «Научный сервис в сети Интернет» Новороссийск, 19–24 сентября 2016 г., стр. 126–138.

[7] SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). W3C Recommendation 27, April 2007, URL: http://www.w3.org/TR/SOAP12–part1.

[8] Web Services Description Language (WSDL) 1.1. W3C Note 15 March 2001, URL: http://www.w3.org/TR/wsdl.

[9] Reference Model for Service Oriented Architecture 1.0. URL: **Ошибка! Недопустимый объект гиперссылки.**

[10] Web Services Resource 1.2 (WS–Resource). URL: **Ошибка! Недопустимый объект гиперссылки.**.

[11] Лаврищева Е.М. Теория объектно–компонентного моделирования программных систем. Препринт 29 ИСП РАН, 2016, 48 стр. ISBN 078–5–91474–025–9.

[12] Лаврищева Е.М. Компонентная теория и коллекция технологий для разработки индустриальных приложений из готовых ресурсов. Труды научно–практической конференции «Актуальные проблемы системной и программной инженерии», АПСПИ–2015, 20–21мая 2015, стр. 101–119.

[13] Лаврищева Е.М. Программная инженерия. Тема 1. Теория программирования, 50 стр.; Тема 2. Технология программирования, 48 стр.; Тема 3. Базовые основы программной инженерии, 52 стр. Методические пособия, Москва, МФТИ, 2016.

[14] Лаврищева Е.М., Петренко А.К. Моделирование семейств программных систем. Труды ИСП РАН, том 28. вып. 6, 2016 г,, стр. 180 –190. DOI: 10.15514/ISPRAS-2016-28(6)-4.

[15] Кулямин В.В., Лаврищева Е.М., Мутилин В.С., Петренко А.К. Верификация и анализ вариабельных операционных систем. Труды ИСП РАН, том 28, вып.3, 2016 г., стр. 189–209. DOI: 10.15514/ISPRAS-2016-28(3)-12.

[16] Лаврищева Е.М. Программная инженерия. Парадигмы, Технологии, CASE–средства программирования. 2 изд. М.: Юрайт, 2016, 280 стр.

[17] Лавришева Е.М., Грищенко В.Н. Связь разноязыковых модулей в ОС ЕС. М., Финансы и статистика, 1982, 136 стр.

[18] Лаврищева Е.М. Программная инженерия и технология разработки программных систем. М., Юрайт, 2017, 431 стр.

[19] Островский А.И. Подход к обеспечению взаимодействия программных сред JAVA и MS.Net. Проблемы программирования, № 2, 2011 г., стр. 37–44.

[20] Лаврищева Е.М. Теория и практика фабрик программ. Кибернетика и системный анализ, том 47, no. 6, 2011 г., стр. 145–158.

[21] Лавріщева К.М., Колесник А.Л., Стеняшин А.Ю. Об'єктне-компонентне проектування(ОКП) програмних систем. Теоретичні і практичні питання. Вісник КНУ, серия физ.–мат. наук, спецвипуск, 2013 г., стр. 150–164 (на украинском).

[22] Ekaterina M. Lavrischeva. Assemblling Paradigms of Programming in Software Engineering. Journal of Software Engineering and Applications, vol. 9, no. 6, 2016, pp. 296–317, DOI: 10.4236/jsea.2016.96021.

[23] Lavrischeva Ekaterina. Ontological Approach to the Formal Specification of the Standard Life Cycle. In Proc. of the 2015 Science and Information Conference (SAI), Jule 28–30, London, pp. 965–972.

[24] John Hebeler, Matthew Fisher, Ryan Blace, Andrew Perez-Lopez. Semantic Web programming. Willey, 2009, 652 p.

[25] Semantic Web. Representation of data on the World Wide Web based on the RDF standards. URL: http://www.w3.org/2001/sw/.

[26] John D. McGregor, David A. Sykes. Practical Guide to testing of Object–oriented software. Addison–Wesley Professional, 2001, 416 p.

[27] Sayyad A. S., Ingram J., Menzies T., Ammar H. Scalable product line configuration: a straw to break the camel's back. In Proc. of the IEEE/ACM 28th International Conference on Automated Software Engineering (ASE 2013), 2013, pp. 465–474.

[28] Лаврищева Е.М., Рыжов А.В. Подход к моделированию систем и сайтов из готовых ресурсов. В сб. трудов XX Всероссийской научной конференции «Научный сервис в сети Интернет», Новороссийск, 17–22 сентября 2018 г. CEUR Workshop Proceedings, vol. 2260, pp. 321–345.

[29] Козин С.В. Конфигурационная сборка варианта ядра Linux для прикладных систем. Труды ИСП РАН, том 29, вып.3, 2018, стр. 161-170. DOI: 10.15514/ISPRAS-2018-30(6)-9.

[30] E.M. Lavrischeva, A.K. Petrenko. Informatics: Formation of computer software and technologies of software systems. ISP RAN/Proc. ISP RAS, 2018, vol. 30, issue 5, pp. 7-30. DOI: 10.15514/ISPRAS-2018-30(5)-1.

[31] Лаврищева Е.М. Информатика и ЭВМ-70. Анализ и аспекты развития. Доклад на Открытой конференции ИСП РАН им. В.П. Иванникова, 2018.

[32] E. M. Lavrischeva. The Scientific Basis of Software Engineering. International Journal of Applied and Natural Sciences (IJANS), vol. 7, issue 5, 2018, pp. 15–32.

[33] Лаврищева Е.М., Пакулин Н.В., Рыжов А.Г., Зеленов С.В. Анализ методов оценки надежности оборудования и систем. Практика применения методов. Труды ИСП РАН, том 30, вып. 4., 2018, стр. 99-120. DOI: 10.15514/ISPRAS-2018-30(3)-8.

Modeling of application and information systems from ready–made Internet service resources

[1,2] *Lavrischeva E.M. <lavr@ispras.ru>*
[1] *Mutilin V.S. <mutilin@ispras.ru>*
[1,3] *Kozin S. V. <kozyyy@yandex.ru>*
[1] *Ryzhov A.V. <ryzhov@ispras.ru>*
[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[2] *Moscow Institute of physics and technology, 9, Institutsky per., Dolgoprudny, Moscow region, 141700, Russia*
[3] *National research University Higher School of Economics, 20, Myasnitskaya st., Moscow, 101000, Russia*

Annotation. The approach to creation of complex (information and applied) systems from ready resources (modules, components, reusable components, services, reuses, etc.) is considered. At the heart of the approach of creating systems, Web systems is a component model (CM), which includes functional, system, service and interface resources (GOR), and the algebra of components to perform various operations on the GOR. Application functions components of the KPI are described in the programming languages (PL), interfaces in languages IDL and WSDL, and service components are created in SOA, SCA IBM Sphere, or get out of the Internet libraries, as a ready. The initial components are verified and stored in the repository of GOR and interfaces. The method of assembling GOR: Link in IBM and MS.VS environment are presented; make in BSD, config in JavaEE; IEEE standard 828–96–2012 (Configuration). GOR tested on many test data, verified their accuracy and reliability. A quality certificate based by ISO/IEC 9000 (1–4) is generated for the finished configured product. The prospects of development of security and quality of web systems are given.

Keywords: component; service; resource; system; web–system; Assembly; reliability; quality.

DOI: 10.15514/ISPRAS-2019-31(1)-1

For citation: Lavrischeva E.M., Mutilin V.S., Kozin S.V., Ryzhov A.V. Modeling of application and information systems from ready–made Internet service resources. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 1, 2019. pp. 7-24 (in Russian). DOI: 10.15514/ISPRAS-2019-31(1)-1

References

[1] Lavrischeva E.M., Grishchenko V.N. Assembly programming. Kyiv, Nauk. Dumka, 1991, 236 p. (in Russian).
[2] Lipaev V.V., Posin B.A., Strick A.A. Technology of Assembly programming. M., Radio and communication, 1992, 324 p. (in Russian).
[3] Lavrischeva E. M., Grishchenko V. N. Assembly programming. The basics of the software product industry. Kiev, Nauk. Dumka, 2009, 371 p. (in Russian).
[4] Ershov A.P. Experience of integral approach to the actual problem of Software. Cybernetics, № 3, 1984, pp. 11–21 (in Russian).
[5] Lavrischeva E.M., Karpov L.E., Tomilin A.N. Approaches to the representation of scientific knowledge in Internet science. In Proc. of the XIX All–Russian scientific conference "Scientific service on the Internet", Novorossiysk, 18–23 September 2017, pp. 310–326 (in Russian).
[6] Lavrischeva E.M., Karpov L.E., Tomilin A.N. Semantic resources for development of ontology of scientific and engineering subject areas. In Proc. of the XVIII All–Russian scientific conference "Scientific service on the Internet", Novorossiysk, September 19–24, 2016, pp. 126–138 (in Russian).
[7] SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). W3C Recommendation 27, April 2007, URL: http://www.w3.org/TR/SOAP12–part1.
[8] Web Services Description Language (WSDL) 1.1. W3C Note 15 March 2001, URL: http://www.w3.org/TR/wsdl.
[9] Reference Model for Service Oriented Architecture 1.0. URL: **Ошибка! Недопустимый объект гиперссылки.**.
[10] Web Services Resource 1.2 (WS–Resource). URL: **Ошибка! Недопустимый объект гиперссылки.**.

[11] Lavrischeva E.M. Theory of object–component modeling software systems. Preprint 29, ISP RAS, 2016, 48 p., ISBN 078–5–91474– 025–9 (in Russian).
[12] Lavrischeva E.M. Component theory and collection of technologies to develop industrial applications from ready–made resources. In Proc. of the 4th scientific–practical conference "Actual problems systems and software engineering", APSSE–2015, 20–21 may 2015, pp. 101–119 (in Russian).
[13] Lavrischeva E.M. Software engineering. Topic 1.Theory Programming, 50 p.; Topic 2. Programming technology, 48 p.; Topic 3. Fundamentals of software engineering, 52 p. Methodical manuals, Moscow, MIPT, 2016 (in Russian).
[14] Lavrischeva E.M., Petrenko A.K. Software Product Lines Modeling. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 6, 2016, pp. 40-64. DOI: 10.15514/ISPRAS-2016-28(6)-4 (in Russian).
[15] Kulyamin, V.V., Lavrischeva E.M., Mutilin V.S., Petrenko A.K. Verification, and analysis of various operating systems. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 3, pp. 189–208. DOI: 10.15514/ISPRAS-2016-28(3)-12 (in Russian).
[16] Lavrischeva E.M. Software engineering. Paradigms, Technologies, CASE-Software. 2nd ed. Moscow, Yurayt, 2016, 280 p. (in Russian).
[17] Lavrischeva E.M., Grishchenko V.N. Communication of multilingual modules in the EU OS. M.: Finance and statistics, 1982, 136 p. (in Russian)
[18] Lavrischeva E.M. Software engineering and technology for development of software systems. Moscow, Yurayt, 2017, 431 p. (in Russian)
[19] Ostrovsky A.I. Approach to software interaction JAVA environments and MS .Net. The problems of programming, №2, 2011, pp. 37–44 (in Russian).
[20] Lavrischeva E. M. Theory and practice of software factories. Cybernetics and Systems Analysis, volume 47, issue 6, November 2011, pp 961–972. DOI: 10.1007/s10559-011-9376-5.
[21] Lavrischeva E.M., Kolesnik A.L., A.Yu. Stenyashin. Object–component design of software systems. Theoretical and applied questions. Bulletin of Taras Shevchenko National University of Kyiv, Series Physics &Mathematics, Kiev, 2013, special issue, pp. 103–117 (in Ukrainian).
[22] Ekaterina M. Lavrischeva. Assemblling Paradigms of Programming in Software Engineering. Journal of Software Engineering and Applications, vol. 9, no. 6, 2016, pp. 296–317, DOI: 10.4236/jsea.2016.96021.
[23] Lavrischeva Ekaterina. Ontological Approach to the Formal Specification of the Standard Life Cycle. In Proc. of the 2015 Science and Information Conference (SAI), Jule 28–30, London, pp. 965–972.
[24] John Hebeler, Matthew Fisher, Ryan Blace, Andrew Perez-Lopez. Semantic Web programming. Willey, 2009, 652 p.
[25] Semantic Web. Representation of data on the World Wide Web based on the RDF standards. URL: http://www.w3.org/2001/sw/.
[26] John D. McGregor, David A. Sykes. Practical Guide to testing of Object–oriented software. Addison–Wesley Professional, 2001, 416 p.
[27] Sayyad A. S., Ingram J., Menzies T., Ammar H. Scalable product line configuration: a straw to break the camel's back. In Proc. of the IEEE/ACM 28th International Conference on Automated Software Engineering (ASE 2013), 2013, pp. 465–474.
[28] Lavrischeva E.M., Ryzhov A.G. The approach to the creation of systems and sites of ready–made resources. In Proc. of the XX All–Russian scientific conference "Scientific service on the Internet", Novorossiysk, 17–22 September 2018, CEUR Workshop Proceedings, vol. 2260. pp. 321–345 (in Russian).
[29] Kozin S.V. Configuration build of the Linux kernel variant for application systems. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue3, pp. 161-170 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-9.
[30] E.M. Lavrischeva, A.K. Petrenko. Informatics: Formation of computer software and technologies of software systems. ISP RAN/Proc. ISP RAS, vol. 30, issue 5, 2018, pp. 7-30. DOI: 10.15514/ISPRAS-2018-30(5)-1.
[31] Lavrishcheva E. M. Informatics and Computer-70. Analysis and development aspects. Conference paper, Ivannikov ISP RAS Open Conference, 2018.
[32] E. M. Lavrischeva. The Scientific Basis of Software Engineering. International Journal of Applied and Natural Sciences (IJANS), vol. 7, issue 5, 2018, pp. 15–32.
[33] Lavrischeva E. M., Pakulin N.V., Ryzhov A.G., Zelenov S.V. Analysis of methods for assessing the reliability of equipment and systems. Practice of methods. ISP RAN/Proc. ISP RAS, vol. 30, issue 3, 2018, pp. 99-120 (in Russian). DOI: 10.15514/ISPRAS-2018-30(3)-8.

Управление требованиями к ответственным системам. Обзор решений

[1] *Н. К. Горелиц <nkgorelits@2100.gosniias.ru>*
[2] *Д. С. Кильдишев <kildishev@ispras.ru>*
[2,3,4,5] *А. В. Хорошилов <khoroshilov@ispras.ru>*

[1]*Государственный научно-исследовательский институт авиационных систем,*
Россия, 125319, г. Москва, ул. Викторенко, 7
[2]*Институт системного программирования им. В.П. Иванникова РАН,*
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25
[3] *Московский физико-технический институт,*
141700, Московская область, г. Долгопрудный, Институтский пер., 9
[4] *Московский государственный университет имени М. В. Ломоносова*
Москва, 119991, ГСП-1, Ленинские горы, д. 1
[5] *НИУ "Высшая школа экономики",*
101000, Россия, г. Москва, ул. Мясницкая, д. 20

Аннотация. Требования являются неотъемлемой частью любого процесса разработки программных и аппаратных систем. Особенно тщательно относятся к требованиям при работе над ответственными системами, использование которых связано с риском для человеческой жизни. Разработка таких систем, как правило, контролируется сертифицирующими органами, требующими применения лучших практик с целью обеспечения безопасности разрабатываемого продукта. В статье рассматривается один из подходов к организации работы с требованиями, который сформировался на основе опыта разработки бортового оборудования гражданских воздушных судов и получил распространение в других отраслях. Приводится набор типовых задач, возникающих при таком подходе. Отталкиваясь от выделенного набора типовых задач формируется методика рассмотрения и оценки инструментов управления требованиями. Предложенная методика применяется для анализа ряда коммерческих и свободно распространяемых инструментов и в заключении формулируются выводы относительно их применения для управления требованиями в проектах по разработке ответственных систем.

Ключевые слова: управление требованиями; ответственные системы; инструменты для управления требованиями; управление изменениями; трассируемость

DOI: 10.15514/ISPRAS-2019-31(1)-2

Для цитирования: Горелиц Н. К., Кильдишев Д. С, Хорошилов А. В. Управление требованиями к ответственным системам. Обзор решений. Труды ИСП РАН, том 31, вып. 1, 2019 г., стр. 25-48. DOI: 10.15514/ISPRAS-2019-31(1)-2

1. Введение

Требования составляют фундамент любого проекта. С работы с требованиями начинается процесс реализации идеи, превращения потребностей и пожеланий в результат - законченный продукт или услугу. Требования и порожденные ими артефакты позволяют сформировать у команды разработчиков единое представление о целях проекта, о разрабатываемом продукте и используемых методах.

По определению международного стандарта по разработке требований ISO/IEC/IEEE 29148 требование - это утверждение, транслирующее или выражающее потребность и связанные с ней ограничения и условия [1].

Требования могут быть представлены в виде текстов на естественном языке, UML диаграмм или текстов на специализированных предметно-ориентированных языках. И даже если требования не зафиксированы документально, они все равно присутствуют в виде представлений о разрабатываемой системе и ее функциях.

Эффект от четкой формулировки требований и последующей аккуратной работы с ними особо заметен при разработке распределенной командой сложных систем с большим количеством связанных элементов. Чем масштабнее проект и больше количество разработчиков, тем сложнее удерживать разработку в рамках, преодолевать разнообразные риски и в итоге достичь удовлетворительного результата.

Связи требований с другими артефактами разработки, в том числе и с производными от требований, помогают осуществлять успешное взаимодействие всех членов и процессов проекта.

Тем не менее часто встречаются проекты со слабо формализованными требованиями, постановки задач с высокой степенью неопределенности. Недостаточная детализация требований может привести к усложнению разработки, нарушению взаимопонимания в коллективе, срыву сроков и реализации других рисков, провоцирующих дополнительные затраты.

В ежегодном выпуске Pulse of profession 2018 от PMI (Project Management Institute) приведены результатов опроса 4,5 тысяч специалистов по управлению проектами на предмет причин, по которым проекты, стартовавшие в организациях за последние 12 месяцев, потерпели неудачи. В 35% случаев респонденты указали в качестве основной причины неудач проектов проблемы с требованиями – с их сбором, анализом, управлением. Проблемы с требованиями занимают в результатах опроса третье место после изменения приоритетов организации и изменения целей проекта. Для сравнения, в Pulse of profession 2017 проблемы с требованиями указаны в качестве основной причины неудач проектов в 39% случаев (второе место), в 2016 – 37% (третье место), в 2015 – 38% (второе место).

Стоимость исправления ошибки, допущенной на этапе разработки требований заметно выше, чем стоимость ошибок на последующих этапах [2]. По данным Счетной палаты США [3][4] изменения в требованиях приводят к росту затрат на проекты более чем в 3 раза, к задержкам в сроках выполнения проектов более чем в 2 раза.

Для эффективной работы с требованиями согласно большинству источников [1][5][6] необходимо, чтобы требования обладали следующими характеристиками.

- Характеристики, касающиеся связи требований с предметной областью.
  - **Адекватность (adequacy)** – соответствие сформулированных требований всем аспектам потребностей и ожиданий пользователей, а также интересам всех остальных заинтересованных лиц.
  - **Выполнимость (feasibility)** – возможность реализовать требование в рамках заданных условий и ограничений.
- Характеристики представления требований самих по себе (внутренние).
  - **Однозначность (unambiguous)** – одинаковость понимания формулировок требований экспертами в соответствующей предметной области.
  - **Внутренняя полнота (completeness)** – охваченность в описании требований всех аспектов и ситуаций, возможных в рамках описанного контекста работы системы.
  - **Непротиворечивость (consistency)** – согласованность описаний требований друг с другом, отсутствие противоречий и расхождений между ними.
  - **Минимальность (minimality)** – невыводимость одних требований из других на основе формальной логики, однократная формулировка каждого нужного ограничения и отсутствие смысловых пересечений между разными требованиями.

- **Простота (singularity)** – отсутствие необходимости подразбиения требования на составные части.
- **Отсутствие деталей реализации (implementation freedom)** – формулировка требований, а не возможных способов их реализации.
- **Систематичность (systematicness)** – представление в виде системы с четко выделенными атрибутами и ясным описанием взаимосвязей и зависимостей между ними.

- Характеристики, касающиеся использования требований в процессе разработки (внешние).
  - **Проверяемость (verifiability)** – возможность для человека, обладающего определенными навыками, однозначно установить в каждой затрагиваемой требованием ситуации, выполнено оно или нарушено.
  - **Прослеживаемость (traceability)** – возможность установления связей между требованиями и их источникам, с одной стороны, а также с разделами и элементами возникающих при разработке текстов программ, документов и моделей, с другой стороны.
  - **Модифицируемость (modifiability)** – возможность удобного и эффективного внесения изменений в сформулированные требования в ходе процесса разработки, включая поддержку различных их версий и конфигураций, а также управление запросами на изменения.

Для того чтобы в результате предпроектной деятельности и мероприятий самого проекта требования получились качественными, управляемыми и позволяющими эффективно пройти все остальные стадии жизненного цикла, предусмотрены разные виды работы с требованиями, соответствующие разным стадиям и процессам проекта. Они включают в себя активности следующих видов:

- **Выделение требований (requirements elicitation)**, состоящее из определения источников требований, извлечения требований и их согласования.
- **Систематизация требований (requirements analysis)** с целью построения целостного набора требований и определения всех существенных взаимоотношений и связей между ними.
- **Описание требований (requirements specification)** в виде документов или моделей.
- **Валидация требований (requirements validation)**, направленная на проверку характеристик, касающихся связи требований с предметной областью (адекватности и выполнимости).
- **Верификация требований (requirements verification)**, нацеленная на проверку внутренних характеристик требований.
- **Управление требованиями (requirements management)**, включающее контролируемое внесение модификаций в различные представления требований и их атрибуты, управление взаимосвязями требований с другими артефактами, предоставление аналитической и иной информации о наборе требований.

В рамках настоящей работы основное внимание уделяется вопросам инструментальной поддержки управления требованиями, а также активностей, связанных с созданием, модификацией и анализом различных представлений требований и их атрибутов в контексте разработки ответственных систем.

Таким образом, из вышеприведенного списка активностей за рамками рассмотрения остаются только вопросы выделения требований.

2. Требования при разработке ответственных систем

Управление требованиями должно осуществляться с учетом целей и задач проекта. Наиболее аккуратное отношение к требованиям традиционно присутствует при разработке ответственных систем, таких как системы управления в гражданской авиации, железнодорожном транспорте, автомобилестроении, судостроении и атомной энергетике.

Основной особенностью ответственных систем является то, что дефекты в конечном продукте могут повлечь за собой риск для жизни и здоровья человека. Также стоит обратить внимание на обычно весьма длительный срок полезного использования для практически всех типов сложных критически важных по безопасности систем, будь то самолет, судно или атомная станция. На протяжении десятков лет система должна выполнять свои функции с высоким уровнем доверия к безопасности и надежности. В связи с этим, эксплуатация таких систем возможна только после прохождения процедуры сертификации. Так, новая модель пассажирского авиалайнера может приступить к коммерческим рейсам только после получения сертификата типа воздушного судна, которое в России на момент написания данной статьи выдается Росавиацией, а, например, в США – Федеральным управлением гражданской авиации (FAA).

Большинство современных регламентов сертификации ответственных систем предъявляют требования не только к конечному продукту, но и к процессам его разработки [7]. Например, в гражданской авиации процесс разработки комплекса бортового оборудования воздушного судна регламентируется требованиями руководства по разработке воздушных судов гражданской авиации и систем Р-4754А [8], гармонизированного с международным аналогом ARP-4754A [9], а процессы разработки программного и аппаратного обеспечения – требованиями КТ-178С/DO-178C [10][11] и КТ-254/DO-254 [12][13], соответственно. Внедрение этих стандартов в гражданской авиации обеспечило достаточно низкий уровень происшествий, случившихся в результате проявления дефектов в бортовом оборудовании, и многие идеи, заложенные в них, в настоящее время распространились по регламентирующим документам в других отраслях.

2.1 Основные задачи разработки ответственных систем

Рассмотрим основные задачи, упоминающиеся в стандартах на разработку ответственных систем, которые относятся к работе с требованиями.

- **Регламенты.** В рамках проекта должны быть сформированы регламенты, в которых зафиксированы правила оформления требований каждого вида.

  Например, согласно КТ-178С регламент на оформление требований к программному обеспечению (ПО) высокого уровня, как правило, называется «стандартом на разработку требований к ПО», а регламент на оформление требований к ПО низкого уровня – «стандартом на проектирование ПО».

- **Оформление.** Все требования к разрабатываемой системе должны быть задокументированы согласно правилам соответствующего регламента.

- **Идентификация.** Каждое элементарное требование должно получить уникальный идентификатор.

- **Источник.** Для каждого требования должен быть указан источник или обоснование его появления (обычно источником являются требования более высокого уровня к системе, а обоснованием – принятые проектные решения).

- **Формальная инспекция.** Должен быть проведен анализ каждого требования на предмет правильности оформления, а также на предмет отсутствия неясностей, неопределенных условий, противоречий и возможности его проверки

(верифицируемости). При этом, как правило, требуется, чтобы анализ проводился группой инспекторов, независимых от авторов требований.

- **Базовая версия.** Для разделов требований, которые были утверждены (т. е. их проанализировали и анализ не выявил проблем), должна быть установлена базовая версия.

  Согласно КТ-178С, базовая версия (Baseline) – это утвержденная зарегистрированная конфигурация одной или нескольких утвержденных единиц конфигурации (например, элементарных требований или каталога требований), используемая в качестве базы для дальнейшей разработки и изменяемая только через процедуры управления изменениями.

- **Трассируемость.** Должны быть установлены и задокументированы связи между требованиями и артефактами, полученными на их основе, такими как исходный код ПО, тесты и прочие.

- **Управление изменениями.** Внесение изменений в требования, для которых была установлена базовая версия, должно выполняться по решению Совета по управлению изменениями, принимаемому по результатам рассмотрения явно сформулированного сообщения о проблеме, а вслед за внесением изменения должны быть проведены процедуры повторного анализа (формальной инспекции) модифицированных требований призванные определить последствия влияния внесенных изменений на существующий набор объектов.

- **Анализ последствий изменения.** Также по результатам изменения в требовании должна быть проведена оценка влияния этого изменения на источник требования, а также оценка его влияния на артефакты, полученные на его основе. Результатом оценки является указание изменений, которые необходимо внести в другие артефакты, а также список других действий, которые необходимо выполнить для приведения всего набора артефактов в согласованное состояние.

- **Данные трассировки.** Должны быть подготовлены данные трассировки (отчетные материалы), демонстрирующие двустороннюю связь между требованиями разных уровней, между требованиями и всеми артефактами, созданными на базе этих требований. Данные трассировки предназначены для:
  - обеспечения верификации полноты трансформации требований в требования более низкого уровня и другие артефакты;
  - наглядной демонстрации требований, которые не трассируются на требования более высокого уровня;
  - обеспечения верификации того, что нигде в исходном коде или оборудовании не реализованы недокументированные функции.

- **История изменений.** История изменений требований должна сохраняться как минимум на уровне базовых версий и быть доступна для анализа не только в ходе разработки системы, но и в течение всего срока ее эксплуатации [14].

Решить перечисленные задачи потенциально возможно и без применения специализированных систем, например, при помощи использования офисных пакетов [14][15]. Также на предприятиях часто можно столкнуться с непрофильным использованием имеющегося в активе программного обеспечения – по разным причинам (политическим, финансовым и др.) не приобретаются программные продукты, созданные для достижения преследуемых целей, зато создаются сложные «конструкторы» из ранее закупленных предприятием продуктов, изначально не предназначенных для автоматизации возлагаемых на них функций в явном виде. Подобные сложные комбинации инструменты обычно весьма тяжелы в поддержке и интеграции как внутри комбинации, так и при добавлении дополнительных продуктов в связку.

Специализированные системы управления требованиями обладают потенциалом для существенного снижения затрат на разработку и снижения риска возникновения ошибок.

2.2 Функциональные возможности систем управления требованиями

Рассмотрим, какие функциональные возможности систем управления требованиями могут потребоваться при разработке требований к ответственным системам.

I. Структурирование и хранение требований.

I.1. Элементы каталога требований и их свойства.

Как минимум, системы управления требованиями должны поддерживать хранение элементарных требований, но кроме того, при написании требований возникает необходимость в наличии других вспомогательных объектов, таких как:

- определения терминов, использующихся в требованиях;
- примечания, содержащие дополнительные комментарии;
- обоснования, поясняющие причины появления производных требований.

Второй вопрос – это номенклатура атрибутов у элемента каталога. Является ли она фиксированной? Может ли она быть дополнена пользователем? Поддерживается ли типизация атрибутов и ограничения на возможные значения атрибутов?

И наконец, основной вопрос относительно формирования содержательной части требований – какой формат используется для представления текста и других типов информации? Поддерживается ли в нем сложное форматирование, таблицы, изображения, диаграммы, математические формулы?

I.2. Идентификация элементарных объектов.

Для возможности ссылаться на отдельные объекты каталога требований, каждый из них должен иметь уникальный идентификатор. Так как с идентификатором приходится работать не только инструментам, но и людям (например, в ходе проведения формальных инспекций и обсуждения их результатов), то желательно, чтобы хотя бы один из способов идентификации был удобен для восприятия человеком.

Относительно способов формирования идентификаторов существует несколько часто встречающихся решений.

Первый из них – это автоматически генерируемые число-буквенные комбинации, такие как, универсальный уникальный идентификатор (universally unique identifier, UUID). Подобные идентификаторы удовлетворяют требованиям уникальности и используются на практике инструментами для внутренних целей. Но для работы с пользователем подобные идентификаторы не подходят, так как работа с ними затруднена – размер идентификатора сравнительно велик, его сложно запомнить, а значение не дает информации о соответствующем объекте.

Другой подход предполагает сопоставление каждому объекту уникального числа. Если такое число уникально в рамках отдельного одного каталога требований, то для однозначной идентификации в пределах базы требуется использовать составные идентификаторы, включающие идентификатор каталога. Числовые идентификаторы являются более компактными, их можно запомнить и достаточно удобно использовать в человеческом общении, но при этом они не содержат информации об объекте.

Третий вариант предусматривает использование идентификаторов, уникальных в пределах поддерева каталога требований. При таком подходе можно рассчитывать на присвоение объектам более-менее семантически значимых идентификаторов, но для глобальной идентификации потребуется использовать составные идентификаторы, включающие идентификаторы родительских элементов.

Например, «/Требования к ПО/Требования к компоненту MemoryManager/001» может являться составным идентификатором требования с номером 001 в некотором каталоге. При этом в идентификатор входит информация как о местоположении объекта в каталоге, так и о самом объекте. Полученные идентификаторы могут оказаться длиннее UUID, но при аккуратно организованном процессе присвоения идентификаторов они могут оказаться удобнее для восприятия человеком.

Еще один важный аспект поддержки идентификации элементов каталога – это работа с идентификаторами удаленных объектов. Если идентификатор однозначно обозначает определенный объект, то при удалении этого объекта такой идентификатор не должен быть использован повторно, чтобы избежать риска путаницы со ссылками на старый и на новый объекты, в том числе, для случаев, когда эти ссылки хранятся вне системы управления требованиями.

I.3. Структура каталога требований.

Каталог требований, как правило, содержит требования некоторого уровня к определенной целевой системе. Также в каталоге часто находятся вспомогательные элементы: определения, примечания, обоснования. Наиболее распространенный случай организации этих элементов – это поддержка иерархической структуры. На эту структуру можно смотреть как на дерево папок, в которых находятся листовые элементы, или же как на иерархию разделов одного документа. Если иерархия поддерживается, то интересен вопрос, насколько нелистовые элементы каталога (условно папки или разделы) отличаются по свойствам и возможностям от листовых.

I.4. Средства редактирования текста требований.

Одна из ключевых задач разработчика требований – это собственно написание текста требований. Относительно этого основной вопрос к инструментам заключается в том, какие средства для редактирования текста они предоставляют. Наиболее распространенные варианты: поддержка HTML-редакторов в браузере, возможность редактирования требований в традиционных офисных пакетах или во встроенных текстовых редакторах.

Применение интеграции с текстовыми редакторами позволяет использовать опыт широкого круга пользователей по работе в привычной среде для разработки требований. При этом возникает несколько проблем. Первая из них связана с преобразованием требований в офисный формат и обратно, а также с использованием разных версий офисных программ и разных версий установленного у разработчиков требований ПО для управления требованиями, что по причине ограниченной совместимости версий может вызывать коллизии при преобразованиях данных между форматами. Вторая заключается в необходимости поддержки слияния различных версий каталога требований. Подобная ситуация может возникнуть, когда после выгрузки требований в документ в каталог требований были внесены изменения.

Использование браузера в качестве средства просмотра или редактирования требований позволяет обеспечить работу с требованиями на различных устройствах и операционных системах. При этом недостатком является необходимость в поддержке различных представлений требований, если формат описаний и значений свойств требований в инструменте отличается от HTML. Ряд инструментов представляют собой веб платформы и, соответственно, для них HTML редакторы можно считать встроенными.

Встроенные редакторы поддерживаются большинством инструментов и позволяют редактировать требования средствами самого инструментов. Из минусов можно отметить потенциальную необходимость в обучении пользователя работе с инструментом.

II. Поддержка связей.

Под связью понимается отношение элемента каталога требований либо с другим элементом каталога, либо с некоторой внешней сущностью. Связи могут быть различных видов, например, «тест проверяет требование», «требование к компоненту реализует требование к системе» или «требование использует термин, определенный в данном элементе каталога требований». Отдельно стоит выделить такой вид связи как отношение родитель – ребенок.

Принято выделять различные виды связей по количеству связанных узлов – один к одному, один ко многим, многие ко многим. Примером связи один ко многим может послужить отношение родитель – ребенок. Одному родителю соответствует множество детей, одному ребенку – один родитель.

Отдельные связи могут иметь вид многие ко многим – например, требование может уточняться множеством других требований. Требование может уточнять несколько других требований.

В общем случае, связи могут обладать произвольными атрибутами подобно элементам каталога требований.

Задание связей может быть явным, когда указываются идентификаторы обоих элементов отношения, или неявным, когда связь вычисляется, например, на основании значений атрибутов. Второй вариант может быть удобен для таких видов связей как «определение-использование», когда вместо явного указания элемента каталога, определяющего термин, можно перечислить список использованных терминов, а инструмент найдет объект с определением и установит связь автоматически.

Наличие хорошо проработанных связей между данными проекта – один из элементов процесса управления конфигурацией, который должен выполняться на протяжении всего жизненного цикла разработки и тесно связан с активностями остальных процессов жизненного цикла. Наличие связей позволяет проводить различные виды анализа данных и качества процессов и получать информативные результаты о состоянии работы в текущий момент. Это особенно важно в контексте управления проектами [16] – связи обеспечивают поддержание информационной базы для управления проектом в актуальном, полном, целостном состоянии, что в свою очередь помогает руководителю видеть текущую картину разработки и принимать эффективные, своевременные и обоснованные решения на всех этапах проекта.

Таким образом, для каждого класса связей, перечисленного ниже, интересны ответы на следующие вопросы:

- Как задаются и хранятся связи?
- Поддерживаются ли различные виды связей?
- Поддерживаются ли атрибуты связей?
- Какие способы визуализации связей поддерживаются?

II.1 Поддержка связей между элементами каталога требований.

II.2 Поддержка связей между требованиями и их внешним источником.

II.3 Поддержка связей между требованиями и внешними артефактами разработки, например, тестами или исходным кодом.

Если поддержка такого вида связей отсутствует, то на практике применяется обходной маневр, в рамках которого для каждого внешнего артефакта заводится «прокси» элемент каталога требований и используется механизм внутренних связей (п. 2.1). Это позволяет решить задачи организации трассируемости, но при этом приходится решать проблему синхронизации между «прокси» элементами и внешними артефактами.

II.4 Поддержка генерации данных трассировки.

Хотя данные трассировки являются частным случаем визуализации связей, ввиду их важности с точки зрения выполнения требований КТ-178С и других

нормативных документов, авторами статьи было принято решение вынести их в отдельный пункт. В соответствии с целями использования данных трассировки могут поддерживаться специализированные представления для анализа покрытия некоторого множества сущностей связями определенного вида, например, покрытия требований исходным кодом или тестами.

Существует несколько используемых на практике форм представления данных трассировки. Среди них матрица связанности, список связанности и граф связанности.

Рассмотрим два множества элементов каталога требований N и M. Распределение элементов по множествам зависит от задач анализа. В качестве примера можно привести построение матрицы связанности объектов различных типов, например, требований и тестов (с целью решения задачи анализа покрытия требований тестами). Матрица связанности множеств будет представлять собой матрицу размера |N| на |M|. При этом в ячейке (i, j) будет содержаться информация о наличии связи между i-м элементом N и j-м элементом M. В списке связанности для каждого элемента в N будет приведен список всех связанных с ним элементов из M. Граф связанности будет представлять собой направленный граф. В качестве вершин будут включены все элементы N и M, для которых есть входящая или исходящая связь, а сами связи будут отображаться в виде ребер.

III. История изменений и управление изменениями.

III.1 История изменений каталога требований.

Поддержка регистрации истории изменений каталога требований является необходимым условием применимости инструмента для разработки ответственных систем. Работа с историей изменений основывается на понятии версионирования, рассматривая которое, следует выделить следующие моменты.

III.1.1 Базовый объект версионирования.

В качестве таких объектов могут выступать:

- каталог с требованиями в целом;
- произвольное поддерево каталога;
- отдельный объект.

Версионирования только на уровне объектов недостаточно для решения всех задач, так как по версиям отдельных объектов нельзя получить согласованное состояние всего каталога и для этого потребуется создание отдельного указателя конфигурации. И наоборот, версионирования на уровне всего каталога вполне достаточно, так как по версии каталога и идентификатору объекта можно однозначно восстановить состояние отдельного объекта в нужный момент времени.

III.1.2 Идентификатор версии:

- автоматически генерируемый
  - читабельный,
  - нечитабельный;
- определяемый пользователем.

III.1.3 Поддерживаемый набор атрибутов версии:

- дата и время изменения;
- автор изменения;
- комментарий, связанный с изменением.

III.1.4 Подход к сохранению изменений:

- сохранения выполняются пользователем:
  - сессия сохранения может быть прервана и продолжена после перезапуска

инструмента;

- сохранение должно быть произведено в пределах сессии работы с инструментом;
- сохранение должно быть произведено при редактировании отдельного объекта;

• Сохранения выполняются автоматически:

- сохранение происходит при редактировании отдельного объекта;
- сохранение происходит по завершению сессии редактирования.

III.1.5 Поддержка операций над версиями:

• визуализация истории изменений;

• визуализация отличий между двумя выбранными версиями;

• возможность восстановления состояния объекта до определенной версии

III.2 Поддержка статуса утвержденности.

Поскольку разработка требований ведется поэтапно, то часть разделов каталога требований может быть уже утверждена, тогда как другая часть может еще находиться в процессе разработки. Инструмент управления требованиями может предоставлять средства для хранения статуса утвержденности и его визуализации.

III.3 Анализ последствий изменений.

При внесении изменений в требования возникает вопрос о том, как эти изменения повлияют на другие требования и артефакты разработки. Ответ на этот вопрос получают в рамках оценки влияния изменения.

Одно из инструментальных решений для поддержки подобного анализа - это поддержка специального статуса у связей, для которых необходимо провести анализ влияния произошедших изменений на другую сторону связи. Этот статус получил название «подозрительных связей» (suspect link). После проведения анализа влияния статус «подозрительности» снимается.

IV. Организация совместной работы.

IV.1 Разрешение конфликтов.

При совместной работе над общим каталогом требований возникает задача организации совместного редактирования требований. Существует два основных подхода к решению этой задачи. Первый их них предполагает блокировку доступа на время редактирования. При этом пользователь может отметить, что он редактирует определенный набор требований (раздел), и этот набор никто другой редактировать не сможет.

Второй подход заключается в апостериорном слиянии изменений в случае конфликта. При этом оба пользователя смогут редактировать некоторый набор требований локально, но при внесении изменений в общий репозиторий возникнет конфликт, который потребуется разрешить одному из этих пользователей. Обычно разрешением конфликта должен заниматься тот из пользователей, кто вносит изменения позднее. Он должен проанализировать оба набора изменений и вручную совместить их корректным образом. Инструменты могут предоставлять средства упрощения этих действий, но стоит отметить, что автоматическое слияние запрещается многими регламентирующими документами (например, КТ-178С) во избежание неконтролируемых изменений.

IV.2 Поддержка ролей и разграничения доступа.

В процесс работы с требованиями вовлечены люди, решающие разные задачи. Можно выделить разработчиков требований, экспертов, проводящих формальные инспекции, и разработчиков иных артефактов, использующих требования в качестве исходных данных. В зависимости от роли могут различаться права доступа

к каталогу требований, объектам и даже свойствам объектов. Инструменты управления требованиями могут поддерживать такие ограничения прав. Также инструменты могут поддерживать настройку специфических форм представления требований в зависимости от роли пользователя.

В дополнение к базовому набору функциональных возможностей мы также рассмотрим ряд дополнительных возможностей:

V. Обмен требованиями с другими инструментами.

В процесс работы с требованиями оказываются вовлечены разные люди и организации, при этом обладающие различным опытом и использующие различные инструменты. Дополнительные сложности возникают при отсутствии возможности обмена данными между инструментами напрямую в связи с ограничениями по безопасности.

Кроме того, от современных инструментов управления требованиями ожидается способность интегрироваться с другими инструментами, автоматизирующими смежные процессы жизненного цикла продукта, включая не только разработку, но и пред- и околопроектные активности, эксплуатацию, техническое обслуживание и вывод из эксплуатации. Если подобная интеграция отсутствует, появляется большой объем работы по связыванию и консолидации данных жизненного цикла из разрозненных инструментов, которую приходится выполнять вручную. В результате формирование целостного набора данных жизненного цикла оказывается затруднено и возникают дополнительные риски – как в вопросах ресурсов проекта (дополнительные трудозатраты и сдвиг сроков), так и в вопросах качества продукта (риски человеческого фактора).

Для обмена требованиями между различными инструментами применяются механизмы импорта-экспорта каталога требований в стандартизированные представления, такие как файлы специального формата, например ReqIF [17], или стандартизованные программные интерфейсы, такие как OSLC Requirements Management Specification [18].

Следует отметить, что стандарт ReqIF на данный момент имеет три стабильные версии, начиная с 1.0.1. В стандарте кроме подхода к описанию элементов каталога и их свойств также описываются дополнительные механизмы, такие как задание связей, добавление изображений и вложений. При этом многие инструменты на практике могут отходить от описанных в стандарте представлений. Отдельные инструменты также поддерживают только обмен иерархией и свойствами артефактов. Стандарт позволяет обмениваться форматированным текстом с возможностью добавления изображений и произвольных вложений за счет использования подмножества объектов языка XHTML 1.0. При этом накладывается запрет на использование свойства тегов class и ограничения на использование стилей в style.

OSLC представляет собой семейство стандартов, описывающих интерфейс веб-сервисов, предназначенных для интеграции различных инструментов поддержки разработки, включая инструменты управления требованиями и управления изменениями. Инструменты при этом могут выступать в качестве поставщиков соответствующих сервисов или их потребителей.

Применительно к системам управления требованиям OSLC Requirements Management Specification описывает интерфейс веб-сервисов, позволяющий запрашивать отдельные требования и их подмножества, создавать новые требования и редактировать свойства уже существующих элементов каталога.

VI. Поддержка шаблонов и повторное использование.

При разработке требований достаточно часто возникает ситуация, когда можно выделить группы схожих требований или схожих разделов каталога требований, отличающихся в небольшом количестве деталей. В таких ситуациях могут оказаться полезны механизмы шаблонизации и повторного использования.

Первый из них предполагает возможность задания шаблонов группы требований и

добавления в каталог требований несколько копий шаблона, раскрытых с разными параметрами. Если поддержка шаблонов существует только в редакторе требований, то при необходимости внесения изменений в шаблон придется вручную отредактировать все созданные на его основе копии требований. Если же шаблон сохраняется в каталоге и его раскрытие выполняется автоматически, то при необходимости внесения изменений в шаблон все его вариации будут обновлены автоматически. Второй подход получил название механизма повторного использования.

VII. Поддержка генерации отчетов.
При управлении требованиями могут возникнуть ситуации, когда необходимо вывести определенное представление требований или некоторую дополнительную статистическую информацию о каталоге. Примером может послужить вывод печатного документа, содержащего все требования в заданном формате или построение отчета о покрытии требований тестами на основе внешних связей.
Для решения этих задач многие инструменты поддерживают генерацию различных представлений каталога требований, а также механизм генерации документов по задаваемым пользователем шаблонам.

VIII. Средства проактивного информирования об изменениях.
В дополнение к средствам сравнения версий требований и анализа последствий изменения некоторые инструменты предоставляют возможность автоматического информирования о появлении изменений в интересующих пользователя разделах каталога требований. В качестве механизмов для уведомления применяются как внешние средства передачи информации (например, электронная почта или мессенджеры), так и средства, встроенные в инструменты работы с требованиями.
Проактивное информирование используется на практике для различных задач. Одно из встречаемых применений – отслеживание комментариев и дискуссий, связанных с каталогом требований. Это могут быть замечания автору элемента, обсуждения каталога или его фрагментов, в том числе в процессе формальной инспекции. Кроме этого встречается отслеживание структуры и содержимого каталога. При этом обычно формируется и распространяется список внесенных за определенный промежуток времени изменений.

2.3 Инструменты для управления требованиями

В рамках настоящего обзора рассматриваются возможности наиболее распространенных коммерческих программных решений, декларирующих поддержку управления требованиями в контексте разработки ответственных систем, а также ряд свободно распространяемых инструментов. В первую группу входят продукты семейства IBM Rational (RequisitePro, DOORS и DOORS Next Generation), а также инструменты ReqView, Jama и Polarion. Во вторую группу включены инструменты RMTOO, aNimble Platform, Eclipse ProR и разрабатываемый в ИСП РАН инструмент Requality. Последние два инструмента развиваются по модели открытого ядра («open core»), в соответствии с которой базовая функциональность распространяется под свободной лицензией, а дополнительные возможности доступны на коммерческой основе. Для Eclipse ProR коммерческий вариант называется ReqIF Studio, а для Requality – это дополнительные плагины, входящие в состав АРМ ПТ, разрабатываемого совместно с ГосНИИАС.

2.3.1 Методика анализа инструментов

Оценка инструментов будет проводиться с точки зрения поддержки функциональных возможностей, необходимых для работы с требованиями при разработке ответственных систем, которые были сформулированы в разделе .II.1. Для проведения оценки

используется следующая методика. Для каждого инструмента изучаются публично доступные актуальные версии документации и учебно-методических материалов и описываются обнаруженные возможности по каждому из пунктов. Для свободно распространяемых инструментов также анализируется их исходный код. На основе собранной информации проводится экспертная оценка полноты поддержки каждой возможности, в результате чего каждому инструменту присваивается числовая оценка по каждой из следующих групп возможностей:

- структурирование и хранение требований;
- поддержка связей;
- поддержка управления изменениями на уровне каталога;
- поддержка управления изменениями на уровне отдельного объекта;
- поддержка статуса утвержденности и оценки влияния последствий изменения;
- поддержка совместной работы;
- обмен требованиями с другими инструментами;
- другие возможности.

Недостатком данного подхода является субъективность экспертных оценок и возможность получения недостоверных оценок ввиду неполноты документации или недостаточной аккуратности ее изучения. Тем не менее, методика позволит получить первичную оценку, которая может быть уточнена в дальнейшем по мере получения дополнительной информации.

2.3.2 Объекты анализа

Для анализа использовались следующие версии инструментов и материалы.

- IBM Rational RequisitePro 7.1.5. По причине окончания поддержки были использованы данные последней версии документации [21], рекламных материалов [22] и встроенной документации.
- IBM Rational DOORS 9.4. Официальная документация [23].
- IBM Rational DOORS Next Generation 6.0.6. Официальная документация [25].
- ReqView 2.4.0. Официальная документация [26].
- Jama Connect версии 8.30. Официальная документация [27].
- Polarion REQUIREMENTS 3.18. Официальная документация [28].
- rmToo 24.0. Официальная документация [29].
- aNimble Platform 0.4. Официальная документация [30].
- Eclipse ProR 0.13. Официальная документация [31].
- Requality 1.0. Официальная документация [32].

2.3.3 Краткое описание результатов анализа

В настоящем разделе представлено краткое описание результатов проведенного анализа. Более подробная информация, включающая детали рассмотрения по каждому инструменту, представлена в [34].

Базовая функциональность по работе с объектами каталога требований, их идентификации и иерархической организации, а также средства редактирования текста присутствуют практически у всех рассмотренных инструментов (табл. 1). Исключением стал RMTOO, у которого отсутствует возможность не допустить переиспользования идентификаторов

после удаления объекта. Также RMTOO оказался единственным инструментом, не поддерживающим добавления к объектам каталога атрибутов, определяемых пользователем, и не поддерживающим отношения родитель-ребенок между объектами каталога. Следует отметить отсутствие поддержки читабельных идентификаторов, не подлежащих повторному использованию у ProR и Requality, хотя для Requality такая возможность доступна при помощи дополнительного плагина.

Относительно возможностей по оформлению текста требований наиболее ограниченным оказался RequisitePro, у которого описание требования может содержать только текст без какого-либо форматирования. В ReqView и ProR отсутствует возможность использовать в тексте требований таблицы. В DOORS существует поддержка таблиц специального вида, у которых каждая ячейка является отдельным требованием. Также в DOORS таблицы можно вставить в текст требования как OLE-объекты. Но в этом случае их можно редактировать только с помощью внешних редакторов в настольном клиенте, а в веб-интерфейсе такие таблицы представлены как картинки. Это не позволяет считать поддержку таблиц в тексте требований в DOORS полноценной. Также следует отметить неожиданное отсутствие в DOORS поддержки выделения фрагментов текста моноширинным шрифтом.

Табл. 1. Структурирование и хранение требований
Table 1. Structuring and storage of requirements

	1.1 Типы	*1.2 ID*	*1.3 Иерархия*	*1.4 Редакторы*	
				ВР	*ТП*
RequisitePro	П	ЧРД	Р	П	+
DOORS	П	ЧОД	РМ	ПРК	
		ИД			
DOORS NG	П	ЧИД	Р	ПРКТ	
ReqView	П	ЧОД	РМ	ПРК	
Jama	П	ЧИД	Р	ПРКТФ	
		ЧРД			
Polarion	П	ЧРД	Р	ПРКТ	
RMTOO	Ф	ЧР	М	ПРКТФ	
aNimble	П	ЧИД	Р	ПРКТ	
ProR	П	РД	Р	ПРК	
Requality	П	ЧР	Р	ПРКТ	+
		РД			

1.1 Типы – элементы каталога требований и их свойства
П/Ф – расширяемый/фиксированный набор типов.
1.2. ID – Идентификация элементарных объектов
Ч – идентификатор читабелен;
И/Р/О – идентификатор уникален в пределах инструмента/проекта/поддерева;
Д – идентификатор удаленного узла не будет повторно использован.
1.3. Иерархия – Структура каталога требований
Р – отношение родитель-ребенок;
М – средства группировки объектов в папки/модули.
1.4. Редакторы – Средства редактирования текста требований.
ВР – возможности встроенного редактора:
П/Р/К/Т/Ф – поддержка обычного текста/форматирования изображений/таблиц/формул;
ТП – интеграция с текстовыми процессорами.

Поддержка связей между объектами каталога требований и генерация данных трассировки присутствует у всех инструментов (табл. 2). При этом практически все инструменты поддерживают задание связей, настраиваемых пользователем. Исключением оказались RequisitePro, RMTOO и aNimble. Поддержка связей с внешними объектами не столь распространена. Возможность, по крайней мере, генерации отчетов о покрытии требований элементами, получающимися на их основе, такими как тесты или исходный код, доступна в RequisitePro, DOORS, DOORS NG, Jama, Polarion и Requality. А возможность установить связь с источником требований во внешних документах в явном виде присутствует только в RequisitePro и Requality.

Табл. 2. Поддержка связей между объектами и с внешними сущностями
Table 2. Support of relationships between objects and with external entities

	2.1. Внутр.	*2.2. Источник*	*2.3. Потребитель*	*2.4 Трассировка*
RequisitePro	1	+	+	1
DOORS	3		+	2
DOORS NG	3		+	2
ReqView	3			1
Jama	3		+	1
Polarion	3		+	1
RMTOO	2			1
aNimble	1			1
ProR	3			1
Requality	3	+	+	1

2.1 Внутр. – поддержка связей между объектами каталога:
1 – только один тип связей;
2 – поддержка множества типов связей;
3 – поддержка типов связей, задаваемых пользователем.
2.2 Источник – поддержка связей между требованиями и их источником во внешних системах.
2.3 Потребитель – поддержка связей с внешними артефактами разработки, разрабатываемыми на основе требований.
2.4 Трассировка – поддержка генерации данных трассировки:
1 – матрицы и/или списки связанности
2 – матрицы и/или списки связанности+графические представления

Управление историей изменений на уровне всего каталога (табл. 3) отсутствует только в ReqView и aNimble, что существенно сокращают возможность их применения для разработки ответственных систем, поскольку для решения задач конфигурационного управления требованиями потребуется привлекать дополнительный инструментарий. ProR в обязательном порядке формирует очередную версию при завершении редактирования отдельного объекта, а RequisitePro – при завершении очередного сеанса работы с инструментом. Оба варианта являются не самым лучшим решением с точки зрения удобства использования. Кроме того, ProR вместе с RMTOO и Requality формируют нечитабельный идентификатор версии. Для ProR ситуация усугубляется отсутствием возможности указать комментарий к версии. Все инструменты поддерживают базовые операции по визуализации истории изменений и сравнению версий. Также все кроме Polarion позволяют восстановить состояние каталога, соответствующее выбранной версии.

Табл. 3. История изменений на уровне каталога
Table 3. History of changes at the directory level

	3.1 ID	3.2 Атрибуты	3.3 Сохранение	3.4 Операции
RequisitePro	АЧ	ДВЗ	П2	ИРВ
DOORS	П	ДВЗ	П3	ИДВ
DOORS NG	П	ДВЗ	П3	ИДВ
ReqView	-	-	-	-
Jama	П	ДВЗ	П3	ИДВ
Polarion	АЧ	ДВ	П2	ИМ
	П	ДВЗ	П3	ИД
RMTOO	АН	ДВЗ	П3	ИДВ
aNimble	-	-	-	-
ProR	АН	ДВ	А1	ИДВ
Requality	АН	ДВЗ	П3	ИДВ

3.1 ID – Идентификатор версии:
АЧ – автоматический, читаемый;
АН – автоматический, нечитаемый;
П – задаваемый пользователем.
3.2 Атрибуты – поддерживаемый набор атрибутов версии:
Д – дата и время изменения;
В – автор изменения;
З – комментарий.
3.3 Сохранение – подход к сохранению изменений:
П/А – сохранение выполняется пользователем/автоматически:
сохранение при редактировании отдельного объекта;
сохранение в пределах сессии работы с инструментом;
работа может быть прервана и продолжена после перезагрузки.
3.4 Операции – поддержка операций над версиями:
И – поддержка визуализации истории изменений.
Поддержка сравнения двух версий:
Р – рабочей и выбранной;
Д – дух выбранных.
М – выбранной и состояния до изменения.
В – восстановление состояния объекта выбранной версии.

Управление историей изменений на уровне отдельных объектов (табл. 4) дополняет историю изменений на уровне каталога более удобными средствами работы в контексте одного объекта. Для ReqView и aNimble это единственная возможность отслеживать историю разработки требований, а Polarion, RMTOO и ProR, наоборот, работают с историей изменений только на уровне всего каталога. При этом они предоставляют возможность просмотра истории изменений выбранного объекта, которая автоматически извлекается из истории версий проекта. Все остальные инструменты формируют историю изменений объектов, в рамках которой версии объектов обладают собственным набором атрибутов и

собственными операциями по просмотру, сравнению и восстановлению. В подавляющем большинстве случаев формирование версии происходит независимо от формирования версии каталога в целом. Исключением является Requality, у которого новая версия объекта может быть сформирована только при сохранении всего каталога, но при этом версии объекта управляются по собственным правилам и обладают независимым идентификатором.

Табл. 4. История изменений на уровне объектов
Table 4. History of changes at the object level

	4.1 ID	4.2 Атрибуты	4.3 Сохранение	4.4 Операции
RequisitePro	АЧ	ДВ	НП2	ИР
DOORS	АН	ДВ	НП2	ИМВ
DOORS NG	АН	ДВ	НА1	И
ReqView	АН	ДВ	НА1	ИМ
Jama	П(АН)	ДВ	НП2 (НА1)	ИМ
Polarion	-	-	-	ИД
RMTOO	-	-	-	ИД
aNimble	АЧ	ДВ	НП1	ИРВ
ProR	-	-	-	ИДВ
Requality	АЧ	ДВ	П3	ИР

4.1 ID – идентификатор версии:
АЧ – автоматический, читаемый;
АН – автоматический, нечитаемый;
П – задаваемый пользователем.
4.2 Атрибуты – поддерживаемый набор атрибутов версии:
Д – дата и время изменения;
В – автор изменения.
4.3 Сохранение – подход к сохранению изменений:
Н – сохранение выполняется независимо от версии проекта.
П/А – сохранение выполняется пользователем/автоматически:
1 – сохранение при редактировании отдельного объекта.
2 – сохранение в пределах сессии работы с инструментом.
3 – работа может быть прервана и продолжена после перезагрузки.
4.4 Операции – поддержка операций над версиями:
И – Поддержка визуализации истории изменений;
Поддержка сравнения двух версий:
Р – рабочей и выбранной;
Д – двух выбранных;
М – выбранной и состояния до изменения;
В – восстановление состояния выбранной версии.

Все инструменты кроме aNimble и ProR поддерживают процесс анализа требований, отслеживая статус утвержденности требований (табл. 5). Эта пара вместе с Requality также не предоставляет специализированных возможностей по анализу влияние последствий изменений по связям между объектами.

Табл. 5. Процессные аспекты и организация совместной работы
Table 5. Process Aspects and Collaboration

	5.1 Статус	*5.2 Изменения*	*5.3 Разрешение конфликтов*			
			Слияние	*Блокировки*	*Доступ*	*ПР*
RequisitePro	+	+		К, П	Т	
DOORS	+	+		П, О	К, Т, С	+
DOORS NG	+	+		П, О	К, Т, С	+
ReqView	+	+	+			
Jama	+	+		О	К, Т	+
Polarion	+	+		П, О	К, Т	+
RMTOO	+	+	+*		К*	
aNimble						
ProR			+*		К*	
Requality	+		+*		К*	

* - *с использованием системы управления версиями GIT*

5.1 Статус – поддержка статуса утвержденности.
5.2 Изменения – анализ последствий изменений.
5.3 Разрешение конфликтов:
Слияние – слияние изменений.
Блокировки:
К – каталога целиком;
П – поддерева каталога;
О – отдельных элементов или свойств.
Доступ – поддержка ограничений прав доступа:
К – ограничения уровня проекта;
Т – ограничения на типы объектов;
С – ограничения уровня свойств.
ПР – Поддержка представлений для различных ролей.

Относительно поддержки командной работы инструменты делятся на три группы. Первая группа инструментов, включающая RequisitePro, DOORS, DOORS NG, Jama и Polarion, предоставляют средства блокировки части требований на время редактирования. Вторая группа (ReqView, RMTOO, ProR и Requality) придерживается оптимистичного подхода к разрешению конфликтов, предоставляя возможность свободно редактировать любые части каталога и проверяя наличие несовместимых модификаций только в момент сохранения изменений. Если такие модификации обнаружены, то пользователю приходится проанализировать несовместимые изменения и сформировать новую версию, в которой должны быть учтены все конфликтующие модификации. В третьей группе находится только aNimble, который при одновременном редактировании одного объекта, просто записывает последнюю сохраняемую версию, тем самым теряя изменения, выполненные параллельно. В результате для предотвращения проблем, возникающих при одновременной работе над проектом aNimble необходимо применять исключительно организационные меры защиты.

Из приятных дополнений можно отметить возможность настройки специализированных представлений каталога требований, привязываемых к роли, которую выполняет пользователь, работая с данным каталогом. Такая возможность поддерживается в инструментах DOORS, DOORS NG, Jama и Polarion.

В табл. 6 представлена информация о поддержке инструментами дополнительных функциональных возможностей. Импорт-экспорт каталога в той или иной форме поддерживают все инструменты за исключением aNimble, причем наблюдается достаточно широкая поддержка формата ReqIF. К сожалению, недостаточная детальность спецификации формата не позволяет рассчитывать на беспроблемность обмена данными с его помощью. Также следует отметить растущую популярность предоставления доступа к содержимому каталога требований при помощи веб-сервисов. Хотя шаблоны проектов или отдельных требований встречаются в большинстве инструментов, более богатые возможности переиспользования требований, в рамках которых вместо принципа копирования-вставки поддерживается сохранение связи с шаблоном с возможностью автоматического распространения вносимых в него модификаций, доступны только в DOORS NG, Jama, Polarion и Requality. Генерация отчетов на основе каталога требований присутствует практически во всех инструментах, только в ProR для этого необходимо применять какие-то внешние инструменты. Проактивное информирование пользователей об изменениях в каталоге получило широкое распространение в коммерческих инструментах и отсутствует в свободно распространяемых.

Табл. 6. Дополнительные функциональные возможности
Table 6. Additional functionality

	6.1		*6.2*		*6.3*	*6.4*
	WebSvc	*ИЭ*	*Шабл.*	*ПИ*	*Отчеты*	*Информирование*
RequisitePro	A	H	+		+	+
DOORS	A	ReqIF2	+		+	+
DOORS NG	T	ReqIF2	+	+	+	+
ReqView		ReqIF1			+	
Jama	A	ReqIF2	+	+	+	+
Polarion	A	ReqIF2	+	+	+	+
RMTOO		ReqIF1	+		+	
aNimble			+		+	
ProR		ReqIF2				
Requality	C	ReqIF2	+	+	+	

6.1 Обмен требованиями с другими инструментами:
WebSvc - поддержка интеграции при помощи web-сервисов:
T – встроенная поддержка OSLC;
A – поддержка OSLC при помощи дополнений;
C – поддержка нестандартного интерфейса.
ИЭ – поддержка операций импорта/экспорта каталога:
ReqIF1 – только импорт ReqIF документов;
ReqIF2 – импорт/экспорт ReqIF документов;
H – нестандартный формат импорта и экспорта.
6.2 Поддержка шаблонов и повторное использование:
Шабл. – поддержка шаблонов;
ПИ – поддержка повторного использования.
6.3 Отчеты – Поддержка генерации отчетов.
6.4 Информирование – Проактивное информирование об изменениях.

В соответствии с методикой для каждого инструменты была проведена экспертная оценка полноты поддержки возможностей по каждой из восьми категорий, описанных в подразделе

3.1. Результаты этой оценки в графической форме представлены на рис. 1. Детальное описание алгоритма выставления оценок и сами оценки по каждой категории представлены в [34].

3.4 Выводы

Рассматривая пригодность инструментов для использования в разработке ответственных систем, следует отметить следующие значимые недостатки:

- В aNimble отсутствует управление историей изменений на уровне всего каталога, поддержка совместной работы и операции импорта-экспорта.
- В ReqView отсутствует управление историей изменений на уровне всего каталога.
- В RMTOO отсутствует возможность не допустить переиспользования идентификаторов после удаления объекта.
- ProR поддерживает историю изменений на уровне всего каталога, но только посредством автоматического сохранения объекта при завершении его редактирования, без читабельного идентификатора версии и без возможности добавить к версии комментарий. Также в ProR отсутствуют читабельные идентификаторы требований и встроенные средства генерации отчетов.
- RequisitePro содержит необходимую функциональность, но уже не поддерживается и устарел по многим параметрам.

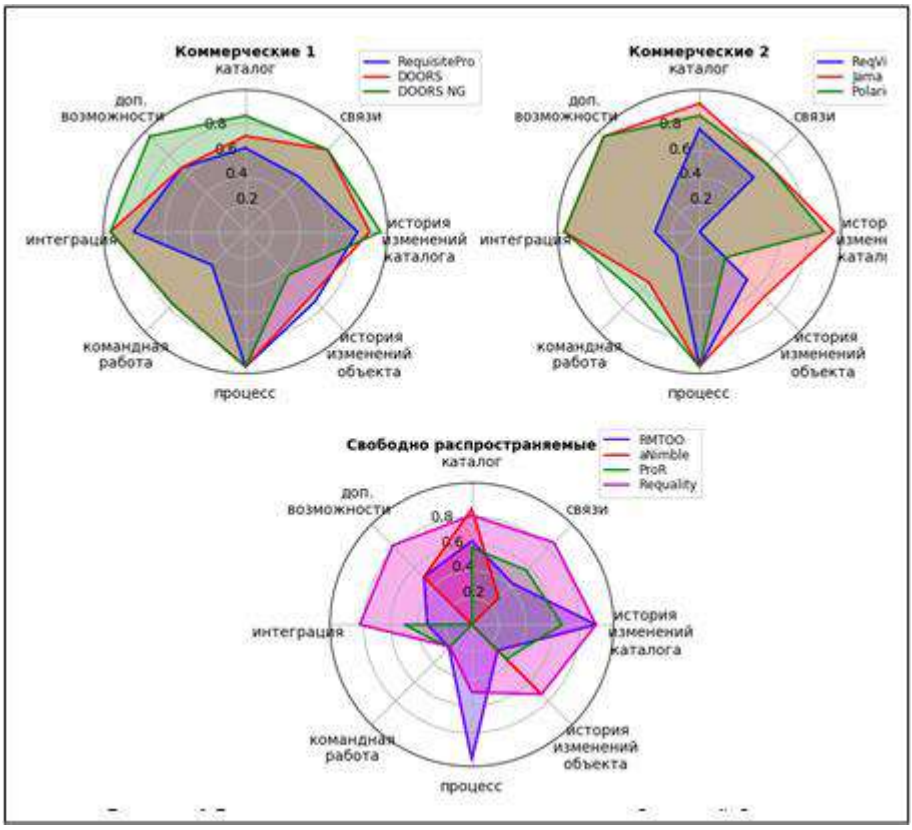


Рис. 1. Результаты оценки инструментов
Fig. 1 Results of tools' evaluation

Суммируя полученные результаты можно сделать следующие выводы. Коммерческие инструменты за исключение ReqView поддерживают все базовые функциональные возможности, необходимые для использования при разработке ответственных систем. Поэтому при сравнении между собой на первый план выходит стоимость владения этими инструментами, удобство использования, возможности по расширению их функциональности и адаптации к особенностям проекта.

Свободно распространяемые инструменты пока не предоставляют всей необходимой функциональности. Исключением является инструмент Requality, в котором присутствуют все наиболее важные функциональные возможности, хотя и наблюдается отставание относительно поддержки командной работы и автоматизации анализа влияния изменений. Это не препятствует использованию Requality в нескольких проектах разработки бортового программного обеспечения в соответствии с КТ-178С, более того его эксплуатации в этих проектах является основой для формирования требований к доработке инструмента. При этом следует отметить, что дополнительная функциональность, адаптирующая Requality для применения при разработке ответственных систем, доступна только в виде коммерческих плагинов.

4. Заключение

В рамках настоящей работы выделен набор функциональных возможностей инструментов управления требованиями, необходимый для их применения при разработке ответственных систем. Для проведения оценки доступности этих возможностей в существующих инструментах предложена методика, основанная на анализе публичной информации о рассматриваемых инструментах и экспертной оценки их возможностей. Данная особенность позволяет применить методику к любому инструменту. С другой стороны, она несет в себе ряд недостатков: субъективность экспертных оценок и возможность получения недостоверных оценок ввиду неполноты документации или недостаточной аккуратности ее изучения. Тем не менее, методика позволяет получить первичную оценку, которая может быть уточнена в дальнейшем по мере получения дополнительной информации.

В рамках настоящей работы методика применена для оценки наиболее распространенных коммерческих программных решений, декларирующих поддержку управления требованиями в контексте разработки ответственных систем (IBM Rational RequisitePro, IBM Rational DOORS, IBM Rational DOORS Next Generation, ReqView, Jama и Polarion), а также ряда свободно распространяемых инструментов (RMTOO, aNimble Platform, Eclipse ProR, Requality). Для каждого из инструментов проведен анализ документации, учебно-методических материалов и исходного кода и сформированы экспертные оценки их возможностей.

В результате анализа выделены значимые недостатки у ряда инструментов, отмечен паритет по реализации базовых функциональных возможностей в коммерческих инструментах и отставание по возможностям свободно распространяемых инструментов.

Список литературы

[1] ISO/IEC/IEEE 29148 Systems and software engineering – Life cycle processes – Requirements engineering.

[2] Dabney, J. B. Return on Investment of Independent Verification and Validation Study Preliminary. Phase 2B Report. NASA, 2003.

[3] GAO-06-391 Assessments of Selected Major Weapon Programs, Report to Congressional Committees, United States Government Accountability Office, 2006.

[4] GAO-09-326SP Assessments of Selected Major Weapon Programs, Report to Congressional Committees, United States Government Accountability Office, 2009.

[5] В.В. Кулямин, Н.В. Пакулин, О.Л. Петренко, А.А. Сортов, А.В. Хорошилов. Формализация требований на практике. Препринт No. 13 ИСП РАН, 2006, 70 стр.

[6] Requirements management: A practice guide, PMI, 2016, 82 p.
[7] Карл И. Вигерс. Разработка требований к программному обеспечению. Русская редакция, 2004, 576 стр.
[8] Руководство Р-4754А по разработке воздушных судов гражданской авиации и систем. М., АР МАК, 2016, 131 стр.
[9] SAE ARP4754A. Guidelines for Development of Civil Aircraft and Systems. 2010.
[10] Квалификационные требования КТ-178С. Требования к программному обеспечению бортовой аппаратуры и систем при сертификации авиационной техники. М., АР МАК, 2016, 131 стр.
[11] Software Considerations in Airborne Systems and Equipment Certification (RTCA DO-178C), 2011.
[12] Квалификационные требования КТ-254. Руководство по гарантии конструирования бортовой электронной аппаратуры. М., АР МАК, 2011, 89 стр.
[13] Design Assurance Guidance for Airborne Electronic Hardware (RTCA DO-254), 2000.
[14] Горелиц Н.К., Гукова А.С., Песков Е.В. Критерии, предъявляемые к программному обеспечению для разработки сложных сертифицируемых систем, критичных по безопасности. Труды ИСП РАН, том 30, вып. 4, 2018 г., стр. 63-78. DOI: 10.15514/ISPRAS-2018-30(4)-4
[15] Joy Beatty, Megan Jackson Stowe et al. Requirements Management Tool Evaluation Report. Seileve, 2016.
[16] Сабуров М.А., Синицын С.В. Роль учета состояния конфигурации программной продукции в управлении проектами. Авиакосмическое приборостроение, 2008, № 6, стр. 2-6.
[17] Requirements Interchange Format, The Object Management Group (OMG), 2016
[18] Open Services for Lifecycle Collaboration Requirements Management Specification Version 2.0. IBM, 2012
[19] Juan M. Carrillo de Gea, Joaquín Nicolás, José L. Fernández Alemán, Ambrosio Toval, Christof Ebert, Aurora Vizcaíno. Requirements engineering tools: Capabilities, survey and assessment. Information and Software Technology, vol. 54, no. 10, 2012, pp. 1142-1157
[20] Rational Unified Process Best Practices for Software Development Teams. Rational Software White Paper, 2001
[21] Rational RequisitePro. Version 2003.06.00, Rational Software Corporation, 2006
[22] IBM Rational RequisitePro. URL: ftp://ftp.software.ibm.com/software/rational/web/datasheets/reqpro.pdf, дата обращения 20.12.2019
[23] Getting started with Rational DOORS Next Generation. URL: https://jazz.net/help-dev/clm/index.jsp?re=1&topic=/com.ibm.rational.rrm.help.doc/topics/c_compose_reqs.html&scope=null, дата обращения 20.12.2019
[24] И.В. Ковернинский, А.В. Кан, В.Б. Волков, Ю.С. Попов, Н.К. Горелиц. Практический опыт реализации подходов программной и системной инженерии для управления требованиями при разработке программного обеспечения в авиационной отрасли. Труды ИСП РАН, том 28, вып. 2, 2016, стр. 173-180. DOI: 10.15514/ISPRAS-2016-28(2)-11
[25] Rational solution for Collaborative Lifecycle Management V6.0.6 documentation. URL: https://www.ibm.com/support/knowledgecenter/SSJJ9R_6.0.6/com.ibm.rational.clm.doc/helpindex_clm.html, дата обращения 20.12.2019
[26] ReqView Documentation Contents. URL: https://www.reqview.com/doc/welcome.html, дата обращения 20.12.2019
[27] Jama Software. URL: https://community.jamasoftware.com, дата обращения 20.12.2019
[28] Polarion ALM Platform Online Help System. URL: https://almdemo.polarion.com/polarion/help/index.jsp, дата обращения 20.12.2019
[29] rmToo – Requirements Management Tool. URL: http://rmtoo.florath.net/, дата обращения 20.12.2019
[30] aNimble Platform. URL: https://sourceforge.net/projects/nimble/http://rmtoo.florath.net/, дата обращения 20.12.2019
[31] ProR ProR Requirements Engineering Platform. URL: http://www.eclipse.org/rmf/pror/, дата обращения 20.12.2019
[32] Requality: руководство пользователя. URL: http://requality.org/ru/doc.ru.html, дата обращения 20.12.2019
[33] Кильдишев Д.С., Хорошилов А.В. Формализация метамодели системы управления требованиями. Труды ИСП РАН, том 30, вып. 5, 2018 г., стр. 163-176. DOI: 10.15514/ISPRAS-2018-30(5)-10.
[34] Горелиц Н. К., Кильдишев Д. С, Хорошилов А. В. Методика анализа инструментов управления требованиями к ответственным системам, Препринт 32, ИСП РАН, 2019 г. (в печати)

Requirements management for safety-critical systems. Overview of solutions

[1]*N.K. Gorelits <nkgorelits@2100.gosniias.ru>*
[2] *D.S. Kildishev <kildishev@ispras.ru>*
[2,3,4,5] *A.V. Khoroshilov <khoroshilov@ispras.ru>*

[1] *State Research Institute of Aviation Systems,*
125319, Russia, Moscow, Viktorenko Str, 7
[2] *Ivannikov Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia
[3] *Moscow Institute of physics and technology (State University),*
141700, Moscow region, Dolgoprudny, Institutskiy per., 9
[4] *Moscow State University named after M. V. Lomonosov,*
Moscow, 119991, GSP-1, Lenin hills, d. 1
[5] *National research University "Higher school of economics",*
101000, Russia, Moscow, Myasnitskaya, d. 20

Abstract. Requirements are the cornerstone of any complex development project. Inaccurate requirements management may lead to significant consequences like cost overruns or even project failure. The situation is even more serious for safety-critical systems where a failure in end product may cause risk to human lives. In such cases the process of development is often regulated by certification authorities. They require to use the best practices in order to achieve the safety of end product. The paper considers the best practices related to requirements management according to regulations in civil aviation and other safety critical domains. The main groups of best practices include project lifecycle definition, requirements representation, requirements identification, source tracking, reviews, traceability, configuration management, change management, change impact analysis, and history tracking. These groups are used to built a methodology for evaluation of requirements management tools applicable for development of safety critical systems. The methodology relies on assessment of 8 feature groups including 4 main ones (requirements catalogue, relations management, configuration management and change impact analysis, team work) and 4 auxiliary ones tools interoperability, reuse, report generation and proactive notifications. This methodology was applied to analyze several commercial and open-source tools such. The commercial products include IBM Rational products (RequisitePro, DOORS и DOORS Next Generation), Polarion, ReqView and Jama. The open-source tools include RMTOO, aNimble Platform, Eclipse ProR and Requality. Commercial tools except for ReqView support all basic functional features required for safety-critical systems development. So, other aspects gets the main priority for decision making, e.g. cost of ownership, usability and extensibility. Open source tools lack one or another necessary features. Requality is the most close to support all basic capabilities, while it have limited collaboration support and change impact analysis automation. Nevertheless, there are ongoing projects, where Requality extended with a commercial plugin is used as a basic requirements management tool for development of DO-178C software.

Keywords: requirements management; safety-critical systems; requirement management tools; change management; traceability

DOI: 10.15514/ISPRAS-2019-31(1)-2

For citation: Gorelits N.K., Kildishev D.S., Khoroshilov A.V. Requirement management for safety-critical systems. Overview of solutions. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 1, 2019. pp. 25-48 (in Russian). DOI: 10.15514/ISPRAS-2019-31(1)-2

References

[1] ISO/IEC/IEEE 29148 Systems and software engineering – Life cycle processes – Requirements engineering.
[2] Dabney, J. B. Return on Investment of Independent Verification and Validation Study Preliminary. Phase 2B Report. NASA, 2003.
[3] GAO-06-391 Assessments of Selected Major Weapon Programs, Report to Congressional Committees, United States Government Accountability Office, 2006.

[4] GAO-09-326SP Assessments of Selected Major Weapon Programs, Report to Congressional Committees, United States Government Accountability Office, 2009.
[5] V.V. Kulyamin, N.V. Pakulin, O.L. Petrenko, A.A. Sortov, A.V. Khoroshilov. Formalization of requirements in practice. Preprint No. 13, ISP RAS, 2006, 70 стр. (in Russian).
[6] Requirements management: A practice guide, PMI, 2016, 82 p.
[7] Karl Wiegers. Software Requirements, 2nd ed. Microsoft Press, 2003, 544 p.
[8] Guideline R-4754A on the development of civil aircraft and systems. M., AR MAK, 2016, 131 p. (in Russian).
[9] SAE ARP4754A. Guidelines for Development of Civil Aircraft and Systems. 2010.
[10] Qualification requirements CT-178C. Software requirements for onboard equipment and systems for certification of aviation equipment. M., AR MAK, 2016, 131 p. (in Russian).
[11] Software Considerations in Airborne Systems and Equipment Certification (RTCA DO-178C), 2011.
[12] Qualification Requirements CT-254. Guidance on the warranty design of onboard electronics. M., AR MAK, 2011, 89 p.
[13] Design Assurance Guidance for Airborne Electronic Hardware (RTCA DO-254), 2000.
[14] Gorelits N.K. et al. Criteria for software to safety-critical complex certifiable systems development. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 4, 2018, pp. 63-78 (in Russian). DOI: 10.15514/ISPRAS-2018-30(4)-4
[15] Joy Beatty et al. Requirements Management Tool Evaluation Report. Seileve, 2016.
[16] M.A. Saburov, S.V. Sinitsin. The Role of Software Configuration Status According in Project Management. Aerospace Instrument-Making, No. 6, 2008, pp. 2-6 (in Russian).
[17] Requirements Interchange Format, The Object Management Group (OMG), 2016
[18] Open Services for Lifecycle Collaboration Requirements Management Specification Version 2.0. IBM, 2012
[19] Juan M. Carrillo de Gea et al. Requirements engineering tools: Capabilities, survey and assessment. Information and Software Technology, vol. 54, no. 10, 2012, pp. 1142-1157
[20] Rational Unified Process Best Practices for Software Development Teams. Rational Software White Paper, 2001
[21] Rational RequisitePro. Version 2003.06.00, Rational Software Corporation, 2006
[22] IBM Rational RequisitePro. URL: ftp://ftp.software.ibm.com/software/rational/web/datasheets/reqpro.pdf, accessed 20.12.2019
[23] Getting started with Rational DOORS Next Generation. URL: https://jazz.net/help-dev/clm/index.jsp?re=1&topic=/com.ibm.rational.rrm.help.doc/topics/c_compose_reqs.html&scope=null, accessed 20.12.2019
[24] Koverninsky I.V. et al. Practical experience of software and system engineering approaches in requirement management for software development in aviation industry. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 2, 2016, pp.173-179 (in Russian). DOI: 10.15514/ISPRAS-2016-28(2)-11
[25] Rational solution for Collaborative Lifecycle Management V6.0.6 documentation. URL: https://www.ibm.com/support/knowledgecenter/SSJJ9R_6.0.6/com.ibm.rational.clm.doc/helpindex_clm.html, accessed 20.12.2019
[26] ReqView Documentation Contents. URL: https://www.reqview.com/doc/welcome.html, accessed 20.12.2019
[27] Jama Software. URL: https://community.jamasoftware.com, accessed 20.12.2019
[28] Polarion ALM Platform Online Help System. URL: https://almdemo.polarion.com/polarion/help/index.jsp, accessed 20.12.2019
[29] rmToo – Requirements Management Tool. URL: http://rmtoo.florath.net/, accessed20.12.2019
[30] aNimble Platform. URL: https://sourceforge.net/projects/nimble/http://rmtoo.florath.net/, accessed20.12.2019
[31] ProR ProR Requirements Engineering Platform. URL: http://www.eclipse.org/rmf/pror/, accessed 20.12.2019
[32] Requality: User Manual (in Russian). URL: http://requality.org/ru/doc.ru.html, accessed 20.12.2019
[33] D.S. Kildishev, A.V. Khoroshilov. Formalizing Metamodel of Requirement Management System. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue. 5, 2018, pp. 163-176. DOI: 10.15514/ISPRAS-2018-30(5)-10
[34] Gorelits N.K., Kildishev D.S., Khoroshilov A.V. Methods to analyze tools for requirements management for critical systems. Preprint 32, ISP RAS, 2019 (in print) (in Russian)

Анализ характера изменений программ и поиск неисправленных фрагментов кода[1]

М.С. Арутюнян <arutunian@ispras.ru>
Г.С. Иванов <gregory@ispras.ru>
В.Г. Варданян <vaag@ispras.ru>
А.К. Асланян <hayk@ispras.ru>
А.И. Аветисян <arut@ispras.ru>
Ш.Ф. Курмангалеев <kursh@ispras.ru>

Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация. Разработчики программного обеспечения часто прибегают к заимствованию кода – как внутри одного проекта, так и из других проектов. Ввиду возможного содержания ошибки в исходном фрагменте это может привести к её дальнейшему распространению по коду ПО. Используемые библиотеки без исходного кода также могут содержать потенциальные ошибки. Целью данной работы является разработка методов анализа характера изменений между версиями компонентов ПО, для которых отсутствует исходный код. А для изменений потенциально относящихся к исправлению дефектов поиск подобных, но не исправленных дефектов при помощи методов поиска клонов кода. Внедрение предлагаемого подхода к анализу используемых компонентов при разработке ПО позволит обеспечить оценку качества предлагаемых программных «заплаток». Поскольку реализованный метод независим от архитектуры операционной системы, а также работает с исполняемым кодом ПО это позволяет применять его как для анализа сторонних компонентов, так и для анализа бинарных сборок собственного программного обеспечения. Средний процент истинных срабатываний на тестовом наборе CoreBench составляет ~73%.

Ключевые слова: статический анализ кода; клоны кода; анализ исполняемого кода; анализ изменений

DOI: 10.15514/ISPRAS-2019-31(1)-3

Для цитирования: Арутюнян М.С., Иванов Г.С., Варданян В.Г., Асланян А.К., Аветисян А.И., Курмангалеев Ш.Ф Анализ характера изменений программ и поиск неисправленных фрагментов кода. Труды ИСП РАН, том 31, вып. 1, 2019 г., стр. 49-58. DOI: 10.15514/ISPRAS-2019-31(1)-3

1. Введение

Во время написания программы программисты часто используют готовые фрагменты кода, которые могут содержать ошибки [3]. Такими фрагментами могут быть как статически линкуемые библиотеки, так и фрагменты исходного кода, скопированные из другого проекта (например, с открытым исходным кодом). Ошибка может содержаться и в статически линкуемой библиотеке, и в заимствованном фрагменте. Затем такая ошибка может дублироваться в другом месте кода (в случае полного или частичного копирования фрагмента с ошибкой). Последующие исправление ошибки, в силу разных причин, не гарантирует исправления во всех клонах фрагмента [25].

Поиск таких ошибок может быть выполнен путём поиска клонов кода и анализа внесённых изменений в скопированный фрагмент кода. Более того, базовая информация об

[1] Работа поддержана грантом РФФИ 18-07-01153А.

исправлениях может использоваться для создания автоматических инструментов исправления ошибок [6] [7] [8].

В случаях полного или частичного отсутствия исходного кода проекта поиск ошибок такого рода усложняется. В то же время, большинство существующих средств анализа изменений работают на основе исходного кода [9] [10].

В статье описан метод сопоставления функций на базе поиска клонов кода [11] и анализа изменений на исполняемом коде, реализованный в рамках инструмента Binside в Институте системного программирования им. В.П. Иванникова Российской академии наук, на базе Binnavi [14]

Дальнейшее изложение построено следующим образом. В разд. 2 рассмотрен алгоритм сопоставления функций на основе анализ клонов кода, а также проблема поддержки анализа программного кода для различных архитектур. Разд. 3 посвящён анализу характера изменений, разд. 4 – поиску неисправленных ошибок. Разд. 5 описывает реализацию метода в инструменте. Разд. 6 обзору аналогичных работ. Разд. 7 завершает статью, описывая результаты работы.

2. Алгоритм сопоставления функций

Рассматриваются два исполняемых файла [11]. Они дизассемблируются и транслируются в промежуточное представление REIL [23], являющееся языком низкого уровня. Оно состоит из 17 инструкций, каждая из которых совершает одно элементарное действие с явно указанными в операндах переменными. Для ассемблерных инструкций, которые, к примеру, записывают во флаги результаты вычислений, при трансляции в REIL создаются явные инструкции работы с флагами. Использование представления REIL позволяет сделать анализ архитектурно независимым.

На основе указанного представления происходит генерация графа зависимостей программы (ГЗП) [22], графа вызовов функций (ГЗФ), графа потока управления и графа потока данных для каждой функции из обоих исполняемых файлов. Узлам ГЗП соответствуют инструкции REIL, а рёбрам – зависимости по данным и по управлению. При построении ГЗП используются графы потока управления и USE-DEF [21] цепочки (используется в качестве дополнительного модуля в Binside).

Процесс сопоставления состоит из двух этапов. Сначала сопоставление происходит на основе эвристик: для каждой функции рассчитывается ряд эвристик. Если результаты конкретной эвристики для некоторой функции из первого исполняемого файла совпадают с результатами эвристической оценки функции из второго исполняемого файла и не совпадают с оценками других функций, то две функции сопоставляются. Ниже приведены применяемые эвристики [12].

1. Сопоставление функций на основе хеширования ассемблерного кода.
2. Сопоставление ребер графа вызовов на основе хеширования ГЗП по MD-индексу [28].
3. Сопоставление ребер графа вызовов на основе хеширования MD-индекса соседних вершин начальной и конечной вершин ребра.
4. Сопоставление функций на основе хеширования вершин ГЗП, считанных на ГЗП на основе MD-индекса.
5. Сопоставление функций на основе хеширования вершин ГЗП. Эвристика считает хеш на основе зависимостей по данным между инструкциями, группирует их и считает конечный хеш.
6. Сопоставление функций на основе хеша ГЗП. Каждому коду операции сопоставляется простое число. После этого вычисляется произведение этих чисел для всех инструкций. Чтобы избежать переполнения, после каждого произведения учитывается только остаток деления на 2^{64}.

На следующем этапе происходит сопоставление тех функции, которые не сопоставились на первом этапе [11].

1. Сопоставляются все пары вершин графов вызовов функций из первого и второго исполняемого файла. Если их ГЗП изоморфны, то вершины графов вызовов функций сопоставляются. Если ни одна пара не сопоставилась, то сопоставляется пара вершин, функции которых похожи больше, чем функции других пар.
2. Рассматриваются предшественники (преемники) для каждой сопоставленной пары вершин: P1 и P2 (S1 и S2). Для всех пар вершин из P1 и P2 (S1 и S2) вычисляется наибольший общий подграф для их ГЗП и строится матрица из сопоставленных частей. Процесс обнаружения общего подграфа распараллеливается для каждой пары ГЗП, чтобы обеспечить масштабируемость.
3. Применяется венгерский алгоритм [24] для нахождения лучшего соответствия ГЗП функций из P1(S1) и ГЗП функций из P2(S2).
4. Шаги 2-4 повторяются до рассмотрения всех сопоставленных функций.

3. Анализ характера изменений в новых версиях исполняемых файлов

Набор сопоставленных функций и инструкций в них позволяет понять, какой фрагмент кода был добавлен или удален. Цель алгоритма – обнаружить важные изменения кода, являющиеся потенциальным исправлением ошибки. Список изменений, рассматриваемых инструментом, представлен ниже.

1. Добавление нового базового блока в исполняемом коде. Если все инструкции базового блока в первом исполняемом файле не сопоставились с инструкциями сопоставленного ему базового блока из второго исполняемого файла версии, считается, что добавился новый блок. Со стороны исполняемого кода это может означать добавление проверки или цикла.
2. Добавление новой инструкции возврата из функции. Если добавленный участок кода добавляет новый путь в конец функции, то считается, что добавилась инструкция возврата из функции. В этом случае выдаётся предупреждение.
3. Изменение аргументов функции. Предупреждение выдаётся, если от добавленных инструкций существует путь по потоку данных до инструкций, которые обрабатывают аргументы функции.
4. Изменение вызова функции. Предупреждение выдаётся, если в сопоставленных инструкциях вызова функций во втором исполняемом файле вызывается другая функция. В ходе рефакторинга имена функций могут меняться, поэтому алгоритм указывает изменение вызова функции с учётом сопоставленных функций.
5. Добавление новой инструкции выхода из цикла.
6. Добавление новой инструкции для перехода в начало цикла.

4. Поиск неисправленных ошибок

В этом разделе рассматривается случай, когда фрагмент кода с ошибкой был скопирован в несколько частей программы, причём в новой версии программы исправление затронуло только один клон кода. Поиск ошибок, оставшихся неисправленными основан на поиске клонов кода [15].

Изменения в новой версии исполняемого файла, описанные в разделе 3, рассматриваются как исправления. После обнаружения исправлений алгоритм находит клоны

неисправленного фрагмента в новой версии исполняемого файла. Такие клоны могут нуждаться в исправлении.

Как показывает практика, эти фрагменты очень часто содержат всего несколько ассемблерных инструкций. Поиск клонов таких фрагментов приводит к многочисленным ложным срабатываниям. Для решения данной проблемы были разработаны три разных алгоритма расширения неисправленных участков кода.

1. Фрагмент неисправленного кода расширяется до уровня функции, в которой он находится. Для неисправленной функции находятся все клоны в новой версии (аналитиком задаётся минимальный процент схожести).
2. Фрагмент неисправленного кода расширяется по базовым блокам по следующему принципу: в множество вставляется базовый блок, в котором найдено исправление, затем рассматриваются соседние базовые блоки по потоку управления. Количество базовых блоков ограничивается аналитиком. В новой версии исполняемого файла ищется полученное множество базовых блоков с учётом кодов операций и потока управления между базовыми блоками.
3. Фрагмент неисправленного кода расширяется по инструкциям по следующему принципу: в множество вставляются инструкции базового блока, в котором найдено исправление, потом рассматриваются соседние базовые блоки по потоку данных. Далее делается поиск полученного множества инструкций с учётом потока данных в новой версии исполняемого файла.

5. Структура инструмента

На рис. 1 представлена общая архитектура инструмента. На вход инструмента подаются два исполняемых файла. Сначала исполняемые файлы дизассемблируются с помощью IDA Pro [13], результаты выгружаются через инструмент Binexport [27] в базу данных PostgresQL [26]. Ассемблерное представление в базе данных транслируется в представление REIL.

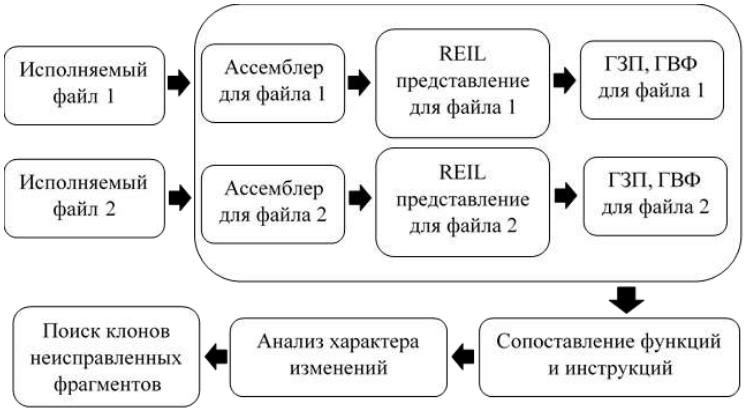


Рис 1. Архитектура инструмента
Fig. 1. Architecture of the tool

После трансляции ассемблерного представления в REIL происходит формирование информации необходимой для дальнейшей работы и возврат управления в основной инструмент, где выполняется сопоставление функций, анализ характера изменений и поиск неисправленных фрагментов

6. Обзор аналогичных работ

SPAIN [16] представляет собой масштабируемую систему анализа изменений, которая автоматически идентифицирует изменения безопасности и суммирует шаблоны изменений, а также соответствующие им шаблоны дефектов. В частности, если даны исходные и исправленные версии исполняемого кода программы, SPAIN находит функции, которые были изменены. Затем система обнаруживает изменённые трассы (то есть, последовательность базовых блоков) для каждой исправленной функции в целях описания изменений на уровне функции. Эти трассы могут содержать безопасные или небезопасные изменения. Посредством семантического анализа определяется, какие изменения безопасны, а какие нет. Затем с помощью анализа помеченных данных на исправленных функциях суммируются образцы небезопасных изменений и соответствующие им дефекты.

PVDF [17] вычисляет семантику исправлений для дефектов. Затем семантика изменений используется для обнаружения новых дефектов в исполняемых файлах. PVDF берёт исполняемый файл с дефектом и соответствующее изменение в качестве входных данных, а затем извлекает семантику. Данный инструмент похож на SPAIN, однако он предполагает наличие изменений и фокусируется только на одном конкретном типе дефекта.

Другие исследования основаны на символьном выполнении. BinHunt [18] пытается найти семантические различия между версиями. Чтобы определить, насколько похожи два базовых блока, инструмент проводит проверку равенства на равенство всех возможных пар уравнений из обоих наборов с использованием SMT-решателей. На основе идентифицированных семантически равных базовых блоков алгоритм обратного отслеживания находит наибольший общий подграф между двумя функциями и получает сходство двух функций. После нахождения изменённых базовых блоков информация передается аналитику. iBinHunt [19] похож на BinHunt, однако использует межпроцедурный анализ и анализ помеченных данных. Эти инструменты рассматривают семантику программы, но они очень трудоёмки.

7. Результаты

В табл. 1 приведены результаты тестов по нахождению клонов неисправленного фрагмента в новых версиях. Такие фрагменты были исправлены в последующих версиях.

Табл. 1. Результаты работы
Table 1. Work results

Проект	Коммиты git		Имя функции с исправленной ошибкой	Имя функции с неисправленной ошибкой
	Старая версия	Новая версия		
Tcpdump	B534e304	d3aae719	juniper_monitor_print	juniper_mlfr_print
Tcpdump	C2ef6938	50a44b6b	ikev1_nonce_print	1.ikev1_hash_print 2.ikev1_sig_print 3.ikev1_ke_print 4.ikev1_vid_print
libosip	79240bdd	a54f15b8	osip_www_ authenticate_init	1.sdp_connection_init 2.osip_authorization_init 3.osip_authentication _info_init
Libosip	80a955e7	03fe3a1c	osip_negotiation_sdp_ build_offer	__osip_negotiation_sdp _build_offer

В табл. 2 приведены результаты анализа изменений на тестовом наборе Corebench [20].

Табл. 2. Результаты на тестах из набора Corebench
Tab. 2. Results on tests from the Corebench suite

Проект	Коммиты git		Количество найденных изменений	Количество правильных срабатываний	Процент правильных срабатываний
	старая	новая			
find	244453b8	f7197f3a	4	2	50%
find	756b47b1	24e2271e	3	2	67%
find	aca12907	b445af98	1	1	100%
grep	02f1daa1	074842d3	2	2	100%
grep	c1cb19fe	8f08d8e2	2	1	50%
grep	c2b9a4fe	6d952bee	6	6	100%
make	87ac68fe	40a49f24	5	3	60%
make	97f106fa	fc644b4c	9	7	77%
make	c3188c6f	3b1432d8	4	3	75%

В среднем процент правильных срабатываний на тестовом наборе Corebench составляет 73.3%.

8. Выводы

В данной статье был предложен метод анализа характера изменений программ и поиска неисправленных фрагментов между версиями компонентов ПО, для которых отсутствует исходный код. Были сформирован список изменений, наиболее часто используемых для исправления ошибок. А также метод поиска фрагментов кода оставшихся неисправленными в новых версиях исполняемого файла при помощи методов поиска клонов кода. Доля корректных срабатываний на тестовом наборе Corebench составляет 73%. Разработанный метод позволит обеспечить оценку качества предлагаемых программных «заплаток».

Список литературы

[1] S. Ducasse, M. Rieger, S. Demeyer. A language independent approach for detecting duplicated code. In Proc. of the 15th International Conference on Software Maintenance, 1999, pp. 109-118.
[2] T. Kamiya, S. Kusumoto, K. Inoue. CCFinder: A multilinguistic tokenbased code clone detection system for large scale source code. IEEE Transactions on Software Engineering, vol. 28, issue 7, 2002, pp. 654-670.
[3] S. C. Misra и V. C. Bhavsar. Relationships between selected software measures and latent bug-density: Guidelines for improving quality. Lecture Notes in Computer Science, vol. 2667, 2003, pp. 724-732.
[4] ГОСТ 19.102-77. Единая система программной документации. Стадии разработки. Москва, Стандартинформ, 2010.
[5] Security Development Lifecycle (SDL). Режим доступа: https://www.microsoft.com/en-us/sdl, дата обращения 10.01.2019.
[6] F. Long, M. Rinard. Automatic patch generation by learning correct code. In Proc. of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'2016), 2016, pp. 298-312.
[7] D. Kim, J. Nam, J. Song, S. Kim. Automatic patch generation learned from human-written patches, In Proc. of the 35th International Conference on Software Engineering (ICSE), 2013, pp. 802-811.

[8] S. Mechtaev, J. Yi, A. Roychoudhury. Directfix: Looking for simple program repairs. In Proc. of the 37th IEEE International Conference on Software Engineering (ICSE), 2015, pp. 448-458.

[9] Y. Tian, J. Lawall, D. Lo. Identifying linux bug fixing patches. In Proc. of the 34th International Conference on Software Engineering (ICSE), 2012, pp. 386-396.

[10] C.S. Corley, N.A. Kraft, L.H. Etzkorn, S.K. Lukins. Recovering traceability links between source code and fixed bugs via patch analysis. In Proc. of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering, 2011, pp. 31-37.

[11] H.K. Aslanyan. Effective and Accurate Binary Clone Detection. Mathematical Problems of Computer Scienc*e,* vol. 48, 2017, pp. 64-73.

[12] H. Aslanyan, A. Avetisyan, M. Arutunian, G. Keropyan, S. Kurmangaleev, V. Vardanyan, Scalable Framework for Accurate Binary Code Comparison. In Proc. of the 2017 Ivannikov ISPRAS Open Conference (ISPRAS), Moscow, 2017. DOI**:** 10.1109/ISPRAS.2017.00013

[13] Ida Pro. Режим доступа: https://www.hex-rays.com/products/ida, дата обращения 10.01.2019.

[14] Binnavi. Режим доступа: https://www.zynamics.com/binnavi.html, дата обращения 10.01.2019.

[15] А.К. Асланян, Ш.Ф. Курмангалеев, В.Г. Варданян, М.С. Арутюнян, С.С. Саргсян. Платформенно-независимый и масштабируемый инструмент поиска клонов кода в бинарных файлах. Труды ИСП РАН, vol. 28, вып 5, 2016 г., стр. 215-226. DOI: 10.15514/ISPRAS-2016-28(5)-13.

[16] Z. Xu, B. Chen, M. Chandramohan, Y. Liu, F. Song. SPAIN: Security Patch Analysis for Binaries Towards Understanding the Pain and Pills. In Proc. of the IEEE/ACM 39th International Conference on Software Engineering, 2017, pp. 462-472.

[17] S. Letian, F. Jianming, C. Jing, P. Guojun. PVDF: An automatic Patch-based Vulnerability Description and Fuzzing method. In Proc. of the 2014 Communications Security Conference (CSC 2014), 2014), pp. 1-8.

[18] D. Gao, M. K. Reiter, D. Song. Binhunt: Automatically finding semantic differences in binary programs. Lecture Notes in Computer Science, vol. 5308, 2008, pp. 238-255.

[19] J. Ming, M. Pan, D. Gao. iBinHunt: Binary Hunting with Inter-procedural Control Flow. In Proc. of the 15th international conference on Information Security and Cryptology, 2012, pp. 92-109

[20] Corebench. Режим доступа: https://www.comp.nus.edu.sg/~release/corebench/, дата обращения 10.01.2019.

[21] M. Jean, M. Lou. Efficient Computation of Interprocedural Definition-Use Chains. ACM Transactions on Programming Languages and Systems, vol. 16, issue 2, 1994, pp. 175-204.

[22] J. Ferranite, J. Karl, D. Joe. The Program Dependence Graph and Its Use in Optimization. ACM Transactions on Programming Languages and Systems, vol. 9, issue 3, 1987, pp. 319-349.

[23] T. Dullien, S. Porst. REIL: A platform-independent intermediate representation of disassembled code for static code analysis. In Proc. of the CanSecWest Conference, 2009, 7 p.

[24] Bruff Derek. The Assignment Problem and the Hungarian Method, 2005. http://www.math.harvard.edu/archive/20_spring_05/handouts/assignment_overheads.pdf дата обращения 10.01.2019.

[25] CVE-2017-12990/Fix printing of ISAKMPv1 Notification payload data. Режим доступа: https://50.3.69.148/hawken/tcpdump/commit/c2ef693866beae071a24b45c49f9674af1df4028, дата обращения 10.01.2019.

[26] PostgreSQL. Режим доступа: https://www.postgresql.org/, дата обращения 10.01.2019.

[27] Binexport. Режим доступа: https://github.com/google/binexport/, дата обращения 10.01.2019.

[28] T. Dullien, E. Carrera, S. Eppler, S. Porst. Automated Attacker Correlation for Malicious Code. Technical report, DTIC Document, 2010, 9 p.

Analysis of program changes nature and searching for unpatched code fragments

M.S. Arutunian <arutunian@ispras.ru>
G.S. Ivanov <gregory@ispras.ru>
V.G. Vardanyan <vaag@ispras.ru>
H.K. Aslanyan <hayk@ispras.ru>
A.I. Avetisyan <arut@ispras.ru>
Sh.F. Kurmangaleev <kursh@ispras.ru>

Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia

Abstract: Software developers often resort to borrowing code both within one project and from another. Due to the possible content of errors in the source code snippet, this can lead to error propagation across program. Libraries used without source code may also contain potential errors. The purpose of this work is developing methods for analyzing the nature of changes between versions of software components for which source code is missing. And for changes potentially related to the correction of defects, search for similar, but not fixed defects using the code clone search methods. The implementation of the proposed approach to the analysis of the components used in software development will ensure the assessment of the quality of the proposed software patches. Since the implemented method is independent of the architecture of the operating system, and also works with executable software code, this allows it to be used for analyzing third-party components and for analyzing binary builds of proprietary software. The average percentage of true positives on the CoreBench test suite is ~ 73%.

Keywords: code static analysis; code clones; binary code analysis; unpatched code fragments

DOI: 10.15514/ISPRAS-2019-31(1)-3

For citation: Arutunian M.S., Ivanov G.S.,Vardanyan V.G., Aslanyan H.K., Avetisyan A.I, Kurmangaleev Sh.F. Analysis of the nature of program changes and the search for uncorrected code fragments. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 1, 2019. pp. 49-58 (in Russian). DOI: 10.15514/ISPRAS-2019-31(1)-3

References

[1] S. Ducasse, M. Rieger, S. Demeyer. A language independent approach for detecting duplicated code. In Proc. of the 15th International Conference on Software Maintenance, 1999, pp. 109-118.

[2] T. Kamiya, S. Kusumoto, K. Inoue. CCFinder: A multilinguistic tokenbased code clone detection system for large scale source code. IEEE Transactions on Software Engineering, vol. 28, issue 7, 2002, pp. 654-670.

[3] S. C. Misra и V. C. Bhavsar. Relationships between selected software measures and latent bug-density: Guidelines for improving quality. Lecture Notes in Computer Science, vol. 2667, 2003, pp. 724-732.

[4] ГОСТ 19.102-77. Единая система программной документации. Стадии разработки. Москва, Стандартинформ, 2010.

[5] Security Development Lifecycle (SDL). Режим доступа: https://www.microsoft.com/en-us/sdl, дата обращения 10.01.2019.

[6] F. Long, M. Rinard. Automatic patch generation by learning correct code. In Proc. of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'2016), 2016, pp. 298-312.

[7] D. Kim, J. Nam, J. Song, S. Kim. Automatic patch generation learned from human-written patches, In Proc. of the 35th International Conference on Software Engineering (ICSE), 2013, pp. 802-811.

[8] S. Mechtaev, J. Yi, A. Roychoudhury. Directfix: Looking for simple program repairs. In Proc. of the 37th IEEE International Conference on Software Engineering (ICSE), 2015, pp. 448-458.

[9] Y. Tian, J. Lawall, D. Lo. Identifying linux bug fixing patches. In Proc. of the 34th International Conference on Software Engineering (ICSE), 2012, pp. 386-396.

[10] C.S. Corley, N.A. Kraft, L.H. Etzkorn, S.K. Lukins. Recovering traceability links between source code and fixed bugs via patch analysis. In Proc. of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering, 2011, pp. 31-37.

[11] H.K. Aslanyan. Effective and Accurate Binary Clone Detection. Mathematical Problems of Computer Science, vol. 48, 2017, pp. 64-73.
[12] H. Aslanyan, A. Avetisyan, M. Arutunian, G. Keropyan, S. Kurmangaleev, V. Vardanyan, Scalable Framework for Accurate Binary Code Comparison. In Proc. of the 2017 Ivannikov ISPRAS Open Conference (ISPRAS), Moscow, 2017. DOI: 10.1109/ISPRAS.2017.00013
[13] Ida Pro. Режим доступа: https://www.hex-rays.com/products/ida, дата обращения 10.01.2019.
[14] Binnavi. Режим доступа: https://www.zynamics.com/binnavi.html, дата обращения 10.01.2019.
[15] H.K. Aslanyan, S.F. Kurmangaleev, V.G. Vardanyan, M.S. Arutunian, S.S. Sargsyan, Platform-independent and scalable tool for binary code clone detection. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 5, 2016, pp. 215-226 (in Russian). DOI: 10.15514/ISPRAS-2016-28(5)-13.
[16] Z. Xu, B. Chen, M. Chandramohan, Y. Liu, F. Song. SPAIN: Security Patch Analysis for Binaries Towards Understanding the Pain and Pills. In Proc. of the IEEE/ACM 39th International Conference on Software Engineering, 2017, pp. 462-472.
[17] S. Letian, F. Jianming, C. Jing, P. Guojun. PVDF: An automatic Patch-based Vulnerability Description and Fuzzing method. In Proc. of the 2014 Communications Security Conference (CSC 2014), 2014), pp. 1-8.
[18] D. Gao, M. K. Reiter, D. Song. Binhunt: Automatically finding semantic differences in binary programs. Lecture Notes in Computer Science, vol. 5308, 2008, pp. 238-255.
[19] J. Ming, M. Pan, D. Gao. iBinHunt: Binary Hunting with Inter-procedural Control Flow. In Proc. of the 15th international conference on Information Security and Cryptology, 2012, pp. 92-109
[20] Corebench. Режим доступа: https://www.comp.nus.edu.sg/~release/corebench/, дата обращения 10.01.2019.
[21] M. Jean, M. Lou. Efficient Computation of Interprocedural Definition-Use Chains. ACM Transactions on Programming Languages and Systems, vol. 16, issue 2, 1994, pp. 175-204.
[22] J. Ferranite, J. Karl, D. Joe. The Program Dependence Graph and Its Use in Optimization. ACM Transactions on Programming Languages and Systems, vol. 9, issue 3, 1987, pp. 319-349.
[23] T. Dullien, S. Porst. REIL: A platform-independent intermediate representation of disassembled code for static code analysis. In Proc. of the CanSecWest Conference, 2009, 7 p.
[24] Bruff Derek. The Assignment Problem and the Hungarian Method, 2005. http://www.math.harvard.edu/archive/20_spring_05/handouts/assignment_overheads.pdf дата обращения 10.01.2019.
[25] CVE-2017-12990/Fix printing of ISAKMPv1 Notification payload data. Режим доступа: https://50.3.69.148/hawken/tcpdump/commit/c2ef693866beae071a24b45c49f9674af1df4028, дата обращения 10.01.2019.
[26] PostgreSQL. Режим доступа: https://www.postgresql.org/, дата обращения 10.01.2019.
[27] Binexport. Режим доступа: https://github.com/google/binexport/, дата обращения 10.01.2019.
[28] T. Dullien, E. Carrera, S. Eppler, S. Porst. Automated Attacker Correlation for Malicious Code. Technical report, DTIC Document, 2010, 9 p.

Определение ограничений облачной платформы на миграцию ресурсов

А.С. Чадин <alexander.chadin@icloud.com>
Г.А. Бизюкин <uashir@mail.ru>
МИРЭА – Российский Технологический Университет,
119454, Россия, г. Москва, проспект Вернадского, д. 78.

Аннотация. В этой статье рассматриваются разные виды миграции виртуальных машин между вычислительными ресурсами, их логические и физические ограничения в системе OpenStack. Для живой миграции рассмотрены такие методы миграции памяти как pre-copy и post-copy, их временные различия. В разд. 2 для метода post-copy описан алгоритм работы на уровне гипервизора. Проведено сравнение временных показателей для методов pre-copy и post-copy. В разд. 3 описаны физические ограничения для виртуальных сред; эти ограничения учитываются при составлении соглашения об уровне предоставления услуг. Логические ограничения на миграцию виртуальных машин описаны в разд. 5; эти ограничения должны учитываться при построении стратегии распределения ресурсов в облаке. В разд. 6 кратко проанализированы родственные работы. В заключении отмечено, что ограничения необходимо учитывать при разработке стратегий распределения ресурсов.

Ключевые слова: облачные системы; живая миграция; распределение ресурсов.

DOI: 10.15514/ISPRAS-2019-31(1)-4

Для цитирования: Чадин А.С., Бизюкин Г.А. Определение ограничений облачной платформы на миграцию ресурсов. Труды ИСП РАН, том 31, вып. 1, 2019 г., стр. 59-68. DOI: 10.15514/ISPRAS-2019-31(1)-4

1. Введение

Облачные вычисления позволяют использовать разные типы ресурсов (CPU, RAM, Storage) наиболее эффективно за счёт предоставления этих ресурсов многим пользователям и проектам в распределенном режиме. При этом одним из самых значимых преимуществ облачного окружения является возможность переносить виртуальные ресурсы между различными узлами. Для автоматического переноса виртуальных ресурсов проектируются системы, учитывающие показатели нагрузки в облачном окружении, аппаратные и логические ограничения. Такие системы работают с аппаратными платформами, которые могут быть построены на различных архитектурах и между ними может отсутствовать совместимость. Данные ограничения необходимо учитывать при проектировании стратегий распределения вычислительных ресурсов в кластере. В данной работе рассмотрены временные, физические и логические ограничения в миграциях виртуальных машин в облачной системе OpenStack [1].

2. Виды миграций виртуальных машин

Миграция виртуальных ресурсов используется для балансирования вычислительных ресурсов в облачном кластере, снижения потребления электроэнергии и переноса ресурсов между географически-распределенными проектами. Миграцией виртуальных машин называют процесс переноса ресурсов виртуальной машины с исходного вычислительного узла на целевой вычислительный узел. Миграция виртуальной машины включает в себя перенос состояния регистров CPU, подключенных устройств [2]. Выделяют три типа миграции.

- *Холодная миграция*. Виртуальная машина находится в выключенном состоянии, её копия создается на целевом узле, к которой подключаются все внешние устройства (блочные устройства, GPU).
- *Горячая миграция*. Виртуальная машина переводится в состояние гибернации (энергосберегающий режим) и страницы оперативной памяти переносятся на целевой узел, где копия виртуальной машины выходит из гибернации.
- *Живая миграция*. Виртуальная машина находится в активном состоянии, её страницы оперативной памяти переносятся гипервизором на целевой узел. После переноса большинства страниц виртуальная машина приостанавливает свою работу и оставшиеся страницы копируются. После копирования всех страниц оперативной памяти подключаются внешние устройства.

Из перечисленных типов миграций наибольший интерес для исследований представляет живая миграция, так как при ее использовании пользователь виртуальной машины может взаимодействовать с рабочим окружением виртуальной машины. В таких условиях время недоступности виртуальной машины должно быть сведено к минимуму.

Различают два метода живой миграции виртуальной машины:

- *Предварительная копия (pre-copy)* – гипервизор копирует страницы памяти виртуальной машины на целевой узел, и только после копирования последней страницы памяти запускает новую виртуальную машину. Данный метод подходит для миграции виртуальных машин с данными, потеря которых критична. В случае потери сетевого соединения между двумя гипервизорами исходная виртуальная машина продолжит свою работу.
- *Копия страниц по запросу (post-copy)* – на целевой узел переносится вся информация, необходимая для запуска виртуальной машины. Страницы оперативной памяти переносятся по запросу. В случае потери соединения восстановить виртуальную машину не представляется возможным.

Рассмотрим время работы метода предварительной копии. При переносе страниц оперативной памяти на целевой узел виртуальная машина продолжает быть доступной для пользователя (с сохранением ввода-вывода, сетевой активности). Для управления и прогнозирования нагрузки узлов в кластере операторы определяют общее время миграции виртуальной машины. Общее время миграции виртуальной машины можно выразить через следующую формулу:

$$T = T_c + T_d$$

Здесь:

T – общее время миграции виртуальной машины. Учитывается от момента поступления команды в гипервизор до включения скопированной виртуальной машины на целевом узле.

T_c – время итеративного перемещения страниц оперативной памяти ВМ. Учитывается от момента поступления команды в гипервизор до момента приостановки ВМ.

T_d – время недоступности ВМ. Учитывается от момента приостановки ВМ на исходном узле для копирования измененных страниц оперативной памяти до включения скопированной виртуальной машины на целевом узле. [4]

Время недоступности ВМ можно также выразить через следующую формулу:

$$t_d = \left(\frac{P + H}{b_d}\right) n_{K+1}$$

Здесь:

P – общий объем страниц памяти, которые мигрируют, в байтах;

H – накладные расходы на протокол миграции (для каждой страницы);

b_d – пропускная способность сети (в байтах в секунду);

n_{K+1} – набор страниц для последней итерации миграции.

Таким образом, общее время миграции виртуальной машины можно выразить следующей формулой:

$$t_{tot} = \left(\frac{P+H}{b}\right)\sum_{k=1}^{K} n_k + t_d$$

При живой миграции важно учитывать тот факт, что в условиях высокой нагрузки, при которых обновление страниц ОЗУ происходит быстрее, чем сетевой канал способен передать, возможна длительная недоступность виртуальной машины. Это объясняется тем, что гипервизор KVM виртуальной машины при переносе страниц оперативной памяти принимает решение о приостановке виртуальной машины и переносе оставшихся страниц памяти при достижении одного из следующих условий [5]:

- достигнуто максимальное количество итераций переноса измененных страниц памяти;
- достигнуто определенное процентное соотношение перенесённой и общей памяти у ВМ;
- объём измененных страниц оперативной памяти незначителен в сравнении с общим количеством оперативной памяти ВМ.

Такие образом, количество страниц памяти, которые передаются при первой итерации миграции, пропорционально общему количеству выделенной виртуальной машине оперативной памяти. Последующие итерации пропорциональны скорости изменения страниц памяти на мигрируемой виртуальной машине [6].

Метод копии страниц по запросу (post copy) запускает виртуальную машину на целевом узле в максимально сжатые сроки. Фактически, на целевой узел копируются состояния регистров CPU и дополнительная информация, необходимая для запуска виртуальной машины. Параллельно с запуском виртуальной машины начинается процесс переноса оперативной памяти с исходного узла на целевой.

В случае, если процесс виртуальной машины запрашивает страницу памяти, которая ещё не была перенесена, генерируется отказ страницы (page fault) и гипервизор обращается к исходной ВМ по сети для получения страницы. Для того, чтобы работа пользователя не была заблокирована на время получения страницы памяти, гипервизор переключает контекст с потока, вызвавшего page fault, на другой поток. После того, как виртуальная машина получит запрошенную страницу памяти, она возвращает контекст на заблокированный поток. Подробно процесс обработки отказа страницы в ходе живой миграции представлен на рис. 1.

В табл. 1 приведено сравнение методов pre copy и post copy по временным критериям.

Табл. 1. Сравнение видов живой миграции по временным критериям
Tab. 1. Comparison of live migration types according to timing criteria

	Предварительная копия (pre copy)	Копирование по запросу (post copy)
Общее время миграции	(Объем RAM / сетевая пропускная способность) + накладные расходы + недетерминированное количество переноса измененных страниц	(Объем RAM / сетевая пропускная способность) + накладные расходы
Время недоступности ВМ (худший случай)	Перенос регистров ВМ + (Объем RAM / сетевая пропускная способность)	Перенос регистров ВМ

Из данного сравнения видно, что метод post copy является более быстрым методом с меньшим временем недоступности ВМ, но обладает повышенным риском потери виртуальной машины из-за обрыва в сети передачи данных.

Рис. 1. Процесс обработки ошибки страницы в гипервизоре
Fig. 1. The processing of page faults in the hypervisor

В некоторых случаях запрашиваемые страницы памяти расположены рядом друг с другом, и отдельный запрос каждой из страниц приведёт к повышенной нагрузке на сеть и увеличению времени задержки для клиента. Адаптивные алгоритмы на основе статистической информации об активности виртуальной машины способны предсказывать последовательность страниц памяти, которая может быть запрошена мигрируемой ВМ в последующем. Эта способность используется для одновременного переноса сразу группы страниц памяти [8].

3. Физические ограничения на миграцию ресурсов

Любые действия, совершаемые в кластере, должны быть ограничены показателями соглашения об уровне предоставляемых услуг (Service Level Agreement). Данное соглашение заключается между клиентом (заказчиком) и облачным провайдером, которое определяет условия предоставления услуг. Рассмотрим некоторые показатели, влияющие на качество предоставления услуг и имеющие значение при распределении ресурсов в облаке:

- **Производительность vCPU.** Определяется по числу обработанных запросов в секунду (MIPS – million instructions per second). По указанной величине можно сравнить предполагаемую производительность ядра виртуальной машины с существующим оборудованием провайдера. Миграция виртуальной машины на узел с менее быстрым процессором или сокращенным набором процессорных инструкций может снизить показатель MIPS. Одновременная миграция множества виртуальных машин может потреблять большое количество процессорного времени, что отразится на существующих виртуальных машинах на исходном и целевом узлах.
- **Производительность СХД.** Измеряется по количеству операций в секунду (IOPS – Input/Output Operations Per Second). Для разных типов дисков выбирается разный блок чтения/записи. В SLA провайдером фиксируется размер блока чтения/записи, количество параллельных процессов и алгоритм нагрузки СХД, при которых были

получены соответствующие показатели IOPS. Также в SLA фиксируется допустимое уменьшение IOPS от эталонного значения – не более 10%.

- **Средняя сетевая задержка.** Как правило, задержка выше 5 мс во внутренней сети провайдера является критичной. Если миграция виртуальных машин производится между вычислительными центрами (соответственно, используется внешняя сеть передачи данных), переназначение внешнего динамического ip-адреса может вызвать кратковременную потерю пакетов.
- **Доступность ВМ.** Пользователь рассчитывает на доступность виртуальной машины большую часть времени, и миграция не должна прервать работу ВМ, либо привести к потере данных. Среди двух методов миграции предпочтительным является метод предварительной копии, так как при нём исходная виртуальная машина продолжает свою работу и не будет потеряна в случае обрыва соединения внутри ЦОД.
- **Согласованность логических и физических ресурсов.** Миграция виртуальной машины проектом OpenStack Nova (при неправильной обработке ошибок от OpenStack Cinder с интегрированным Ceph) может завершиться появлением двух активных виртуальных машин с одним и тем же UUID (при этом в базе данных Nova запись о виртуальной машины с таким UUID будет одна), подключенных к одному блочному устройству, что приведёт к безвозвратной потере данных пользователя. Провайдер должен обеспечивать согласованность ресурсов, предоставляемых пользователю с фактическим наличием ресурсов на уровне гипервизора и системы хранения данных.

Перечень приведенных показателей и их фактические ограничения могут меняться в зависимости от набора предоставляемых физических ресурсов. В ходе миграции ВМ данные ограничения являются основой для логических ограничений, что описано в последующей подглаве.

4. Логические ограничения на миграцию ресурсов

В зависимости от состояния загруженности облачного кластера, администратор может принять решение о необходимости распределения ресурсов для сохранения работы виртуальных ресурсов в заданных показателях. Для облачной системы OpenStack роль администратора в принятии решений может на себя взять автоматизированная система по распределению ресурсов. На сегодняшний день выделяют [9] два основных направления в распределении ресурсов: балансирование вычислительной нагрузки и динамическая консолидация узлов. Чтобы автоматизированная система могла распределять ресурсы, должны быть приняты во внимание следующие факторы:

a) Живая миграция виртуальной машины может влиять на сетевую задержку. Работающая виртуальная машина, обрабатывающая большой массив информации, склонна к частому обновлению страниц оперативной памяти. От пропускной способности сетевого канала и степени загруженности оперативной памяти ВМ зависит время миграции между вычислительными узлами. Для перемещения страниц оперативной памяти в режиме реального времени рекомендуется выделять отдельный сетевой канал для гипервизора.
b) Для построения плана распределения ресурсов необходимо собирать и хранить показатели CPU и RAM (включая количество тактов CPU, затраченных на все выделенные ВМ ядра) как для виртуальных машин, так и для вычислительных узлов. Для сбора метрик может быть задействован сервис OpenStack Ceilometer с активными libvirt-агентами на вычислительных узлах. Для данного сервиса необходимо активировать модуль SNMP [10], который по UDP-протоколу публикует телеметрию узла. Сбор статистики по сетевым показателям (таким как пропускная способность сети и количество переданных пакетов в секунду) позволит оценить приблизительное время миграции виртуальной машины. Собранные метрики Ceilometer отправляет в

базу данных временных рядов (time series database) такую как Gnocchi [11].

c) Процесс живой миграции виртуальной машины требует от исходного вычислительного узла свободных ресурсов пропорционально объему занимаемых ресурсов виртуальной машиной. Коэффициент оверкоммита ресурсов гипервизора должен быть выставлен с учетом возможной необходимости миграции.

d) Живая миграция возможна только между двумя однотипными гипервизорами. Модели процессоров и набор их инструкций могут различаться.

e) Должна использоваться одна и та же подсеть между вычислительными узлами. По окончании миграции виртуальная машина должна послать анонсирующий Ethernet-пакет с локацией нового сетевого контроллера (NIC) чтобы получить адрес во внутренней сети.

f) Если виртуальная машина запущена с временного образа (ephemeral), образ должен находиться на общем хранилище (shared storage), доступном по сети. Это необходимо для исключения физического перемещения образов по сети чтобы сократить время недоступности ВМ. В качестве протокола для предоставления общего хранилища может использоваться NFS.

Приведенные выше ограничения применимы для гипервизора KVM и облачной системы OpenStack. Коэффициент оверкоммита выставляется применительно к ядрам процессора и оперативной памяти. В публичном облаке распространенными значениями являются коэффициенты 16x для CPU и 1.5x для RAM. Таким образом, планировщик виртуальных машин Nova Scheduler определяет на вычислительном узле в 16 раз больше вычислительных ядер, чем физически имеется, что позволит разместить больше виртуальных машин на узле.

Принятие решения о стабилизации вычислительного кластера можно переложить на автоматизированную систему, в которой роли администратора отведено контролирующее значение. Учитывая физические и логические ограничения, система генерирует план действий, который администратор может подтвердить. Проектирование таких систем и стратегий их работы (балансирование вычислительных ресурсов, консолидация для снижения энергопотребления) – цель последующих исследований.

5. Обзор родственных работ

В работе [2] Кристофер Кларк (Christopher Clark) и Кейр Фрейзер (Keir Fraser) рассматривают две проблемы с доступом к ресурсам при миграции виртуальной машины – проблема доступа к сетевым ресурсам и проблема доступа к блочным устройствам. Для решения доступа к сети предложено на целевом узле сохранять тот же ip-адрес для виртуальной машины, что и на исходном (при условии, что оба узла находятся в рамках одной и той же сети). Проблему с доступом к блочным устройствам предлагалось решать с помощью NAS.

Антон Белоглазов (Anton Beloglazov) и Раджкумар Буйя (Rajkumar Buyya) [3] при проектировании сервиса динамической консолидации OpenStack Neat определяют физические требования к облачной среде в наличии распределенного хранилища NFS или GlusterFS для доступа к блочным устройствам. Помимо этого, в качестве логических требований указана необходимость иметь учетную запись пользователя с правами, достаточными для перевода узла в состояние ACPI S3.

Фабио Чеккони (Fabio Checconi), Томмазо Кучинотта (Tommaso Cucinotta) и Мануэль Стейн (Manuel Stein) в своей работе [12] при рассмотрении проблем живой миграции виртуальных машин выделяют необходимость равного распределения процессорного времени между виртуальными машинами. Для виртуальной машины, которая мигрирует на другой узел, это позволит избежать чрезмерного обновления страниц оперативной памяти (что сократит количество итераций по переносу оперативной памяти). Для этого предлагается

использовать планировщик AQuoSA [13], гарантирующий виртуальной машине определенное количество микросекунд до переключения контекста.

В работе по динамической оптимизации нагрузки [14] рассматривается стратегия распределения и консолидации виртуальных машин между узлами для повышения времени отклика узлов и снижения совокупной стоимости владения. Данная стратегия предполагает наличие распределенного хранилища (shared storage) на узлах, сбор и выдача метрик системой OpenStack Ceilometer (на момент написания статьи проект Ceilometer ещё обладал самостоятельным компонентом для хранения и выдачи метрик), а также специальным образом настроенная сетевая карта с технологией Wake-on-LAN.

Дориан Минаролли (Dorian Minarolli), Артаном Мазрекай (Artan Mazrekaj) и Бендт Фрейслебен (Bernd Freisleben) в работе по долгосрочным прогнозам нагрузок [15] в качестве главных ограничений на миграцию вводят логические показатели SLA. Для каждого показателя определяются пороговые значения, которые необходимо учитывать при построении плана миграции виртуальных машин.

Тема выявления и учета ограничений на миграцию виртуальных машин кажется недостаточно проработанной.

6. Заключение

Процесс миграции виртуальных машин в облачной среде затрагивает разные типы ресурсов, требующие синхронизации для сохранения консистентности рабочих процессов ВМ. Данная работа определяет логические и физические ограничения, которые помогут проводить распределение нагрузки с соблюдением пользовательского SLA. Для снижения энергопотребления и уменьшения задержки в ВМ необходимо оперативно реагировать на изменения нагрузки в вычислительном кластере. Разработка сервиса по распределению ресурсов в облачных средах и соответствующих стратегий, нацеленных на снижение времени задержки ответа, должна вестись с учётом логических и физических ограничений на миграцию ресурсов, что наталкивает на необходимость продолжить изучать различные подходы к миграции и их применимость к гетерогенным рабочим нагрузкам.

Рабочее окружение системы OpenStack позволяет с лёгкостью собирать и хранить данные об утилизации виртуальных машин, что может быть использовано стратегиями при построении плана действий. Последующая работа в данном направлении ведёт к проектированию и разработке стратегий динамической консолидации и балансировании виртуальных машин для облачной системы OpenStack, которые могли бы снизить энергопотребление в условиях цикличной и динамической нагрузки.

Список литературы

[1]. Страница проекта OpenStack – https://www.openstack.org/

[2]. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A. Live migration of virtual machines. In Proc. of the 2nd Symposium on Networked Systems Design & Implementation (NSDI'05), USENIX Association, 2005, pp. 273–286.

[3]. Beloglazov A., Buyya R. OpenStack Neat: a framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds. Concurrency and Computation: Practice & Experience, vol. 27, issue 5, 2015, pp. 1310-1333.

[4]. Gustafsson E. Optimizing Total Migration Time in Virtual Machine Live Migration. Master thesis, Teknisk- naturvetenskaplig fakultet, Uppsala Universitet, 2013, 38 p.

[5]. Liguori A., Kivity A., Kamay Y. kvm: the Linux Virtual Machine Monitor. In Proc. of the 2007 Ottawa Linux Symposium, 2007, pp. 225–230.

[6]. Akoush S., Sohan R., Rice A., Hopper A., Predicting the performance of virtual machine migration. In Proceedings of the 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '10), 2010, pp. 37–46.

[7]. Liu P., Yang Z., Song X., Zhou Y. Heterogeneous Live Migration of Virtual Machines. In Proc. of the International Workshop on Virtualization Technology (IWVT'08), 2008.
[8]. Hines M., Deshpande U., Gopalan K. Post-Copy Live Migration of Virtual Machines. ACM SIGOPS Operating Systems Review, vol. 43, issue 3, 2009, pp. 14-26.
[9]. Rybina K., Patni A., Schill A. Analysing the Migration Time of Live Migration of Multiple Virtual Machines. In Proc. of the 4th International Conference on Cloud Computing and Services Science (CLOSER 2014), 2014, pp. 590-597.
[10]. Serhiienko O. Ceilometer: Full-stack Monitoring for OpenStack, OpenStack Superuser, 2014. Режим доступа: https://superuser.openstack.org/articles/ceilometer-full-stack-monitoring-for-openstack/, дата обращения 10.01.2019.
[11]. Danjou J. OpenStack Ceilometer and the Gnocchi experiment, Julien Danjou blog, 2014. Режим доступа: https://julien.danjou.info/openstack-ceilometer-the-gnocchi-experiment/, дата обращения 10.01.2019.
[12]. Checconi F., Cucinotta T., Stein M.: Real-Time Issues in Live Migration of Virtual Machines. Lecture Notes in Computer Science, vol 6043, 2009, pp. 454-466.
[13]. Palopoli, L., Cucinotta, T., Marzario, L., Lipari, G. AQuoSA — adaptive quality of service architecture. Software – Practice and Experience, vol. 39, no. 1, 2009, pp. 1–31.
[14]. Чадин А.С. Динамическая оптимизация нагрузки на вычислительных узлах частных, публичных и гибридных облаков. Труды ИСП РАН, том 27, вып. 6, 2015 г., стр. 307-314. DOI: 10.15514/ISPRAS-2015-27(6)-19.
[15]. Dorian Minarolli, Artan Mazrekaj, Bernd Freisleben. Tackling uncertainty in long-term predictions for host overload and underload detection in cloud computing. Journal of Cloud Computing, vol. 6, № 4, 2017, 18 p.

Determining cloud platform limits on resource migration

A.S. Chadin <alexander.chadin@icloud.com>
G.A. Biziukin <uashir@mail.ru>

MIREA – Russian Technological University,
78, Vernadskogo prospect, Moscow, 119454, Russia

Abstract. This article discusses the different types of virtual machine migration between computing resources, their logical and physical limitations. For live migration, such memory migration methods as pre-copy and post-copy, their temporary differences are considered. Chapter 2 discusses the work in which the requirements for physical and logical constraints. In chapter 2.1 for the post-copy method, the algorithm for working at the hypervisor level is described. For the pre-copy and post-copy methods a comparison of time indicators was made. Chapter 2.2 describes the physical constraints for virtual environments, which should be taken into account when drawing up a service level agreement. Logical restrictions on the migration of virtual machines are described in chapter 2.3, which should be taken into account when building a resource allocation strategy in the cloud. In conclusion, it is noted that constraints must be considered when developing resource allocation strategies.

Keywords: cloud systems; live migration; resource allocation

DOI: 10.15514/ISPRAS-2019-31(1)-4

For citation: Chadin A.S., Biziukin G.A. Determining cloud platform limits on resource migration. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 1, 2019. pp. 59-68 (in Russian). DOI: 10.15514/ISPRAS-2019-31(1)-4

References

[1]. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A. Live migration of virtual machines. In Proc. of the 2nd Symposium on Networked Systems Design & Implementation (NSDI'05), USENIX Association, 2005, pp. 273–286.

[2]. Beloglazov A., Buyya R. OpenStack Neat: a framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds. Concurrency and Computation: Practice & Experience, vol. 27, issue 5, 2015, pp. 1310-1333.
[3]. Gustafsson E. Optimizing Total Migration Time in Virtual Machine Live Migration. Master thesis, Teknisk- naturvetenskaplig fakultet, Uppsala Universitet, 2013, 38 p.
[4]. Liguori A., Kivity A., Kamay Y. kvm: the Linux Virtual Machine Monitor. In Proc. of the 2007 Ottawa Linux Symposium, 2007, pp. 225–230.
[5]. Akoush S., Sohan R., Rice A., Hopper A., Predicting the performance of virtual machine migration. In Proceedings of the 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS ’10), 2010, pp. 37–46.
[6]. Liu P., Yang Z., Song X., Zhou Y. Heterogeneous Live Migration of Virtual Machines. In Proc. of the International Workshop on Virtualization Technology (IWVT'08), 2008.
[7]. Hines M., Deshpande U., Gopalan K. Post-Copy Live Migration of Virtual Machines. ACM SIGOPS Operating Systems Review, vol. 43, issue 3, 2009, pp. 14-26.
[8]. Rybina K., Patni A., Schill A. Analysing the Migration Time of Live Migration of Multiple Virtual Machines. In Proc. of the 4th International Conference on Cloud Computing and Services Science (CLOSER 2014), 2014, pp. 590-597.
[9]. Serhiienko O. Ceilometer: Full-stack Monitoring for OpenStack, OpenStack Superuser, 2014. Available at: https://superuser.openstack.org/articles/ceilometer-full-stack-monitoring-for-openstack/, accessed 10.01.2019.
[10]. Danjou J. OpenStack Ceilometer and the Gnocchi experiment, Julien Danjou blog, 2014. Available at: https://julien.danjou.info/openstack-ceilometer-the-gnocchi-experiment/, accessed 10.01.2019.
[11]. Checconi F., Cucinotta T., Stein M.: Real-Time Issues in Live Migration of Virtual Machines. Lecture Notes in Computer Science, vol 6043, 2009, pp. 454-466.
[12]. Palopoli, L., Cucinotta, T., Marzario, L., Lipari, G. AQuoSA — adaptive quality of service architecture. Software – Practice and Experience, vol. 39, no. 1, 2009, pp. 1–31.
[13]. Chadin A. Dynamic Optimization of Workload on Compute Nodes in Private, Public and Hybrid Clouds. Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 6, 2015, pp. 307- 314 (in Russian). DOI: 10.15514/ISPRAS-2015-27(6)-19
[14]. Dorian Minarolli, Artan Mazrekaj, Bernd Freisleben. Tackling uncertainty in long-term predictions for host overload and underload detection in cloud computing. Journal of Cloud Computing, vol. 6, № 4, 2017, 18 p.

Методы идентификации человека по походке в видео

[1] *А.И. Соколова <ale4kasokolova@gmail.com>*
[2] *А.С. Конушин <anton.konushin@grapics.cs.msu.ru>*
[1] *Национальный исследовательский университет Высшая школа экономики*
101000, Россия, Москва, ул. Мясницкая, 20
[2] *Московский государственный университет им. М.В. Ломоносова*
119991, Россия, Москва, Ленинские горы

Аннотация. Походка – важный биометрический показатель, позволяющий идентифицировать человека на большом расстоянии и без непосредственного контакта. Благодаря этим качествам, отсутствующим у других популярных идентификаторов, таких как отпечатки пальцев и радужная оболочка глаза, распознавание человека по походке в наши дни стало очень распространено и востребовано в различных сферах, где возможно использование систем видеонаблюдения. С развитием методов компьютерного зрения появляется множество подходов к идентификации человека по движениям в видео, использующих как естественные биометрические характеристики (скелет человека, его силуэт, их изменение во время ходьбы), так и абстрактные признаки. Современные методы объединяют в себе классические алгоритмы анализа видео и изображений и новые подходы, показывающие высокие результаты в смежных задачах компьютерного зрения, таких как идентификация человека по лицу или распознавание действий. Однако из-за большого количества условий, влияющих на саму манеру движения человека и ее представление в видео, задача идентификации человека по походке до сих пор не имеет достаточно точного решения. Многие методы заточены исключительно под условия, присутствующие в базах данных, на которых они обучаются, что ограничивает их применимость в реальной жизни. В данной работе проводится обзор современных методов распознавания человека по походке, их анализ и сравнение на нескольких популярных видео коллекциях и для разных формулировок задачи распознавания, а также выявляются проблемы, препятствующие окончательному решению задачи идентификации по походке.

Ключевые слова: походка; биометрия; силуэт; нейронные сети; идентификация

DOI: 10.15514/ISPRAS-2019-31(1)-5

Для цитирования: Соколова А.И., Конушин А.С. Методы идентификации человека по походке в видео. Труды ИСП РАН, том 31, вып. 1, 2019 г., стр. 69-82. DOI: 10.15514/ISPRAS-2019-31(1)-5

1. Введение

Задача идентификации человека по походке особенно актуальна в современном мире. Согласно биометрическим исследованиям манера движения каждого человека индивидуальна, и фальсифицировать ее практически невозможно, что делает походку уникальным идентификатором, таким как отпечатки пальцев или радужная оболочка глаза. Однако в отличие от этих ``классических" характеристик, походку можно наблюдать издалека, не контактируя с человеком напрямую, поэтому с развитием высококачественных систем видеонаблюдения именно походка становится наиболее подходящим показателем для распознавания.

Основной областью применения распознавания человека по походке является сфера безопасности, где часто есть необходимость идентифицировать человека, попадающего в

поле зрения видеокамер, например, для поимки преступников или контроля доступа на закрытые территории.

Задача распознавания человека по походке очень специфична в силу наличия множества факторов, меняющих походку визуально (наличие каблуков или неудобной обуви, переносимые тяжелые предметы, одежда, скрывающая части тела человека) или влияющих на внутреннее представление модели походки (ракурс, освещение, различные параметры камеры). Поэтому, несмотря на успехи современных методов компьютерного зрения, задачу идентификации по походке пока нельзя назвать решенной. В этой статье представлен обзор методов распознавания человека по походке в видео и их сравнение на популярных наборах данных.

На сегодняшний день существует два основных подхода к получению признаков походки и их классификации: построение признаков вручную и обучение признаков. Первый способ более традиционен и, как правило, основывается на вычислении различных свойств бинарных масок силуэта человека или на исследовании взаимного расположения суставов, относительных расстояний и скоростей, а также других кинетических показателей.

Обучение признаков характерно для искусственный нейронных сетей, набравших популярность в последние годы благодаря выдающимся результатам в решении многих задач компьютерного зрения, таким как классификация видео и изображений, сегментация изображений, детекция объектов, визуальный трекинг и другие. Признаки, обучаемые с помощью нейронных сетей, часто обладают более высоким уровнем абстракции, необходимым для качественного распознавания.

Кроме того, высокое качество идентификации достигается методами, комбинирующими два описанных подхода. На начальном этапе вручную вычисляются базовые характеристики походки, а на их основе обучается нейронная сеть, выделяющая более абстрактные признаки. Несмотря на успешность методов глубинного обучения, на данный момент наилучшего результата на некоторых наборах данных достигают неглубокие алгоритмы, поэтому оба глобальных подхода достойны внимания.

2. Базовые признаки походки

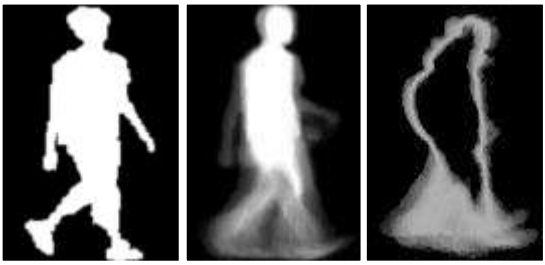
Рассмотрим сначала некоторые классические базовые подходы, в которых признаки походки извлекаются вручную из естественных соображений.

2.1 Бинарные силуэты человека

Наиболее распространенной характеристикой походки является изображения энергии походки (Gait Energy Image, GEI [11]). Такие изображения – усредненные по одному циклу походки бинарные маски силуэта движущегося человека. В предположении, что движения человека во время ходьбы периодически повторяются, вычисляется пространственно-временное описание походки человека. Полученные изображения характеризуют частоты нахождения человека в той или иной позе во время движения. Этот подход получил широкое распространение и лег в основу множества других методов распознавания походки.

Кроме того, многие подходы, не использующие изображения энергии напрямую, предлагают аналогичную агрегацию других базовых признаков. Например, распознавание возможно по изображениям энтропии походки (gait entropy image, GEnI [2]}), где вместо усреднения силуэтов вычисляется энтропия каждого пикселя, или по энергии разницы кадров (frame difference energy image, FDEI [4]), отражающей разности между силуэтами в последовательных кадрах видео.

Визуализация бинарных силуэтов и изображений энергии и энтропии походки представлена на рис. 1.



а) б) в)

Рис. 1. Примеры базовых дескрипторов походки: а) бинарный силуэт, б) изображение энергии походки, в) изображение энтропии походки
Fig. 1. Examples of basic gait descriptors: a) binary silhouette, b) gait energy image, c) gait entropy image

Несмотря на одинаковую простоту и естественность всех этих методов, именно изображения энергии используются и развиваются до сих пор. По таким изображениям, как и по обычным черно-белым, можно вычислять дальнейшие признаки, такие как гистограммы ориентированных градиентов (HOG-дескрипторы) [7, 16] или гистограммы оптического потока (HOF-дескрипторы) [14, 32], или строить более сложные алгоритмы классификации, использующие специфику задачи распознавания походки.

Так, одними из наиболее успешных многоракурсных подходов являются два неглубоких метода, использующих в качестве базовых признаков изображения энергии походки. Первый из них – байесовский подход, предложенный Ли в [15]. Авторы предлагают считать изображения энергии походки случайными матрицами, получающимися из собственно походки и независимого от нее шума, соответствующего различным условиям (таким как угол съемки, различная одежда или наличие переносимых в руках вещей), причем предполагается, что оба слагаемых – нормальные случайные величины. Рассмотрение совместного распределения двух представлений походки в предположении совпадения классов или их различия сводит проблему к задаче оптимизации ковариационных матриц, решаемую с помощью EM-алгоритма. Во втором подходе [19] предлагается обобщение метода линейного дискриминантного анализа [3], а именно – многоракурсный дискриминантный анализ. Для признаков походки, посчитанных для каждого угла обзора, обучается отдельное вложение, так чтобы внутриклассовый разброс был минимален, а межклассовый – максимален.

Идея уменьшения внутриклассовых расстояний и увеличения межклассовых наследуется и в более поздней работе [18], где предлагается единая структура для обучения метрики совместной интенсивности пары изображений и пространственной метрики. Поочередная оптимизация обеих метрик приводит к модели, превосходящей по качеству распознавания базовые модели дискриминантного анализа.

Описанные подходы интуитивно понятны и математически просты, что в дополнение к высоким результатам распознавания дает им преимущество перед многими другими более сложными методами. Общим недостатком методов, использующих GEI для многоракурсного распознавания, является необходимость вычислять изображение энергии для каждого угла обзора, присутствующего в выборке. Поэтому для каждого кадра видео нужно знать, под каким углом он был снят, что не всегда возможно в реальных данных.

Тем не менее, если информация об угле съемки все же известна, ее можно эффективно использовать, применяя модель преобразования ракурса, как это предложено в нескольких подходах [21, 22]. Для признаков походки одного человека, полученных с разных ракурсов, обучается преобразование, переводящее одни в другие. Благодаря таким преобразованиям дескрипторы, соответствующие разным углам съемки, можно вложить в единое пространство, в котором классификация оказывается точнее.

2.2 Поза человека

Другим важнейшим источником информации, помимо силуэтов, является скелет человека. Множество методов распознавания базируется на исследовании позы человека -- положении суставов и основных частей тела и их движении в видео при ходьбе. Подходы, основанные на позе, варьируются от полностью структурных (рассматривающих кинематические характеристики позы) до более сложных, объединяющих в себе кинематические и пространственно-временные характеристики.

К первой группе можно отнести работу [1], в которой распознавание происходит и по походке, и по внешности. В качестве основных характеристик походки при этом берутся абсолютные и относительные расстояния между суставами, а также признаки, основанные на смещениях ключевых точек фигуры между кадрами.

В подходе, предложенном в [30], также исследуется скелет человека, однако авторы вводят более сложную математическую модель, рассматривая семейство гладких Пуассоновских функций расстояния для построения изображения вариации скелета (Skeleton Variance Image, SVI).

Несколько других работ также предлагают модели, основанные на положении частей тела человека, но используют их в совокупности с признаками другой природы. Например, метод 2017 года [8] комбинирует кинематический подход с пространственным, рассматривая в качестве динамических признаков походки как траектории движения суставов, так и изменение формы силуэта во времени.

2.3 Траектории точек фигуры

Еще один неглубокий подход, показывающий высокое качество распознавания, предложен в [6], где рассматриваются траектории движения точек фигуры человека и по ним строятся дескрипторы движения Фишера, которые классифицируются методом опорных векторов.

3. Нейросетевые подходы

Несмотря на обилие структурных неглубоких подходов, сверточные нейронные сети (Convolutional Neural Networks, CNN) занимают прочную позицию во всех задачах компьютерного зрения и в том числе распознавании походки. За последние несколько лет было предложено множество нейросетевых методов идентификации по походке, отличающихся как технически (выбором архитектур сетей, функции потерь, способов обучения), так и идейно -- методом обработки данных и извлечения первичных признаков, подаваемых на вход сети.

В силу того, что фигура и образ человека могут меняться в зависимости от носимой одежды и освещения, важно, чтобы модель обращала внимание не столько на внешние параметры, сколько на само движение фигуры человека. Поэтому большинство методов классифицируют видео не напрямую по кадрам, а вычисляют всевозможные динамические характеристики походки и уже по ним распознают человека.

Одной из таких характеристик, дающих информацию о движении, является оптический поток -- векторное поле видимого движения точек сцены. Его преимущество состоит в том, что обученная на таких данных модель не обращает внимания на цвет, яркость или контрастность кадров видео. Влияние на распознавание оказывает только движение отдельных точек фигуры, а именно это и составляет походку человека.

В нескольких работах [5, 25], появившихся практически одновременно, предлагается рассматривать блоки карт оптического потока аналогично временной составляющей классической двухпоточной модели [24] для распознавания действий. Для нескольких идущих подряд пар соседних кадров вычисляется оптический поток и строится тензор – блок из нескольких карт потока. Для большей точности из этого блока вырезается часть, во

всех кадрах содержащая фигуру человека, и на таких блоках обучается нейронная сеть. На этапе тестирования сеть используется для извлечения признаков, которые потом можно классифицировать любым методом машинного обучения, например, методом опорных векторов (SVM) или методом ближайшего соседа (kNN).

Подход, предложенный в [26], развивает идеи [25], однако отказывается от блоков идущих подряд кадров. Вместо этого более детально исследуется движение точек около важных (как показывают эксперименты) частей тела. В ходе предобработки оценивается поза человека и оптический поток рассматривается в окрестности ступней человека, а также по отдельности в верхней и нижней частях тела (выше и ниже бедер, соответственно). Исследования показывают, что рассмотрение потока в большем масштабе вокруг более «влиятельных» частей дает заметный прирост качества распознавания.

Второй и наиболее популярный источник информации, на основе которого происходит обучение сетей, – бинарные маски силуэтов, о которых уже шла речь при рассмотрении неглубоких методов. В простейшем случае [37] сверточная архитектура обучается по отдельным силуэтам предсказывать человека, которому этот силуэт принадлежит. Как и предыдущих методах, сеть в дальнейшем используется для извлечения признаков, причем переход от дескрипторов отдельных кадров ко всему видео происходит путем выбора максимального отклика по циклу походки. Этот метод наиболее простой из всех глубоких подходов, т.к. при наличии масок силуэтов людей не требует практически никакой дополнительной предобработки. Длину цикла походки определяют, рассматривая автокорреляцию последовательности бинарных изображений. Вследствие того, что два кадра, отличающиеся на полный период походки, должны выглядеть похоже, корреляция таких кадров оказывается больше, чем у любой другой пары, что помогает вычислить длину цикла.

Еще один метод, использующий сами силуэты, предложен в [28]. Двухэтапный алгоритм сначала определяет угол съемки видео, а потом по изначальным данным и модели для найденного угла (своей для каждого ракурса) предсказывает человека. Чтобы учитывать не только пространственные, но и временные характеристики движения, на вход сети подаются не отдельные маски, а блоки из нескольких силуэтов, идущих подряд. Кроме того, изменчивость между кадрами учитывается благодаря архитектурам сетей в обеих подзадачах: авторы используют трехмерные сети, в которых свертки производятся не только в пространстве, но и во времени.

Отдельно стоит выделить множество методов, совмещающих «ручное» выделение признаков, и глубинное обучение. В качестве входных данных для сетей часто используют изображения энергии походки, упоминавшиеся выше. Основанные на этой идее модели варьируются от простейших [23], где неглубокая сеть предсказывает человека по получаемым изображениям энергии, вычисленным для различных ракурсов, до более сложных, как, например, [31], где определяется степень похожести пары GEI изображений и исследуются различные методы сравнения нейросетевых признаков походки, полученных из изображений энергии. Еще в нескольких работах [27, 36] также с помощью двух- и трехпоточных сиамских архитектур определяется, какие изображения походки близки и принадлежат одному человеку, а какие -- разным.

Стоит отметить, что большинство популярных в последние годы архитектур и подходов успешно применяются для распознавания по походке. Например, автокодировщики, используемые для генерации изображений и обучения представлений, также оказываются применимы для идентификации. Авторы [34] предлагают решать проблемы вариативности ракурсов и переносимых предметов с помощью множества автокодировщиков, каждый из которых, подобно модели [21], производит свое преобразование. Таким образом, проходя через последовательность «кодирующих» слоев, изображение приводится к боковому ракурсу, более простому для распознавания. Близкий подход предлагается и в [33], однако для преобразования изображения энергии походки к «базовому» виду (снятому сбоку, без

сумки и пальто) применяются генеративные состязующиеся модели (Generative Adversarial Models, GAN).

Метод [10] также базируется на противоборствующих моделях, однако решает задачу верификации, оценивая и трансформируя углы съемки для вычисления признаков, особенных для каждого ракурса. Кроме того, авторы строят изображение энергии периода походки (period energy image, PEI), усредняя силуэты по коротким временным промежуткам внутри цикла походки. Такой подход дает заметный прирост качества по сравнению с изображениями энергии походки, используемыми в большинстве методов.

Наряду с классическими сверточными сетями прямого распространения для распознавания по походке, как и для других задач анализа видео, используются рекуррентные нейронные сети. Такие архитектуры позволяют даже по простейшим данным (например, силуэты в отдельных кадрах [29] вычислять информативные динамические признаки походки. В более эффективном подходе [9] рекуррентные слои применяются к скелету человека, а именно к тепловым картам для суставов, полученным на предыдущих сверточных слоях из отдельных кадров. Такая модель делает предсказание на основании изменения позы человека, не опираясь напрямую на фигуру и силуэт человека, что делает ее более общей.

Многие из обсуждаемых подходов оцениваются на одних и тех же наборах данных при одних и тех же условиях, в этом обзоре мы приводим сравнение некоторых описанных методов и выделяем наиболее успешные решения.

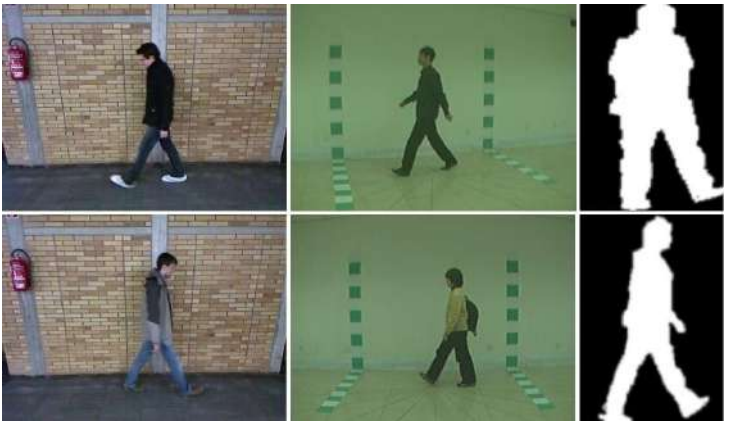


Рис. 2. Примеры кадров из баз данных походок:
TUM-GAID (слева), CASIA-B (посередине) и OULP (справа)
Fig. 2. Examples of frames from the gait database:
TUM-GAID (left), CASIA-B (middle), and OULP (right)

4. Базы данных для распознавания человека по походке

Наиболее широко используемыми в наше время сложными наборами данных для распознавания человека по походке являются базы TUM-GAID [12], OU-ISIR Large Population Dataset (OULP) [13] и CASIA Gait Dataset B [35]. Примеры кадров видео из этих баз можно найти на рис. 2.

Первая база данных используется для распознавания сбоку, все видео в ней сняты под углом 90º, она не очень большая (по 10 видео для 305 человек), однако состоит из полноценных цветных видео, что делает ее применимой для большого количества подходов. Кроме того, в этой базе присутствуют данные, снятые с разницей в полгода, что дает возможность проверить устойчивость алгоритмов к изменениям походки со временем.

Два других набора предназначены для многоракурсного распознавания. В то время, как CASIA – сравнительно небольшая по количеству человек база, но с очень большой

вариативностью ракурсов (11 различных углов съемки от 0 до 180 градусов для 124 человек), набор OULP состоит из видеопоследовательностей для более, чем 4000 человек, снятых двумя камерами, причем ракурс съемки плавно меняется от 55º до 85º. Данные из этой коллекции распространяются в виде масок силуэтов, поэтому не все описанные методы применимы к этой базе данных.

Многие из рассмотренных методов оценены именно на этих наборах данных, поэтому и мы приведем сравнение подходов на них.

5. Результаты работы методов

На описанных базах данных оценка качества алгоритмов происходит следующим образом. Сначала модель настраивается на части данных (как правило, это все видео для некоторого подмножества людей), а затем тестируется на другом наборе данных, состоящем из видео для остальных людей. Для баз TUM и OU-ISIR разделение на обучающую и тестовую выборки предоставлено авторами коллекций. В экспериментах с CASIA модель обучается на первых 24 людях и тестируется на всех оставшихся ста. В качестве метрики качества обычно рассматривается точность распознавания -- доля правильно классифицируемых видео.

Табл. 1. Сравнение результатов распознавания на базе TUM-GAID
Table 1. Comparison of recognition results with the base TUM-GAID

Метод	Точность
Sokolova, блоки ОП [25]	97.5%
Sokolova, части тела [26]	99.8%
Castro, SNN+ SVM [6]	98.0%
Marín-Jiménez [20]	98.9%
Castro, дескрипторы Фишера [6]	99.2%
Zhang [37]	97.7%

В табл. 1 приведено сравнение результатов распознавания на базе TUM-GAID. Наилучшего качества достигает нейросетевой подход [26], однако до его появления глубинные методы долгое время не могли превзойти по качеству метод [6], не использующий нейронные сети. Как будет видно ниже, и в задаче многоракурсного распознавания все еще продолжается борьба глубоких и неглубоких методов.

Табл. 2. Сравнение результатов распознавания видео, снятых в разные дни, на базе TUM-GAID
Table 2. Comparison of video recognition results taken on different days with the base TUM-GAID

Метод	Точность
Castro, CNN + SVM [5]	59.4%
Marín-Jiménez [20]	63.6%
Castro, дескрипторы Фишера [6]	60.4%

Интересно также рассмотреть устойчивость алгоритмов ко времени съемки видео. Оказывается, что качество идентификации сильно портится, если между временем первого попадания человека в поле зрения камер и моментов тестовой съемки и распознавания проходит много времени. В табл. 2 показаны результаты распознавания на базе TUM-GAID, когда между «обучающими» и «тестовыми» видео проходит полгода. Точность каждого из представленных алгоритмов падает примерно на 40%, что говорит о том, что выученные этими алгоритмами признаки плохо переносимы во времени.

Для многоракурсных баз сравнение, как правило, производится для всевозможных пар ракурсов: данные, снятые под некоторым «тестовым» углом классифицируются алгоритмом, при настройке которого используется другой, «обучающий» угол съемки.

Для сравнения различных алгоритмов на базе OU-ISIR используют два популярных протокола тестирования. Один из них, как уже было сказано, предоставлен авторами: из

коллекции выбрано 1912 человек, которые пятью способами разделяются пополам на обучающую и тестовую выборки, после чего качество моделей, обученных на этих разбиениях, усредняется. Второй протокол реализует кросс-валидацию, причем модели строятся на данных для 3844 людей, для которых в базе присутствуют данные с обеих камер. Для удобства результаты сравнения обычно агрегируют, рассматривая разности между «обучающим» и «тестовым» углами. В табл. 3 показана средняя точность методов для каждого из 4 возможных значений разности углов.

Табл. 3. Сравнение результатов распознавания на базе OU-ISIR
Table 3. Comparison of recognition results with the base OU-ISIR

Метод	0º	10º	20º	30º
Zhang [37]	94,1%	71,6%	21.8%	2,9%
Shiraga [23]	94,9%	93,9%	90,5%	80,65%
Li [15]	98,3%	98,2%	97,3%	94,6%
Mansur [19]	96,8%	96,3%	94,2%	90,3%

Самый простой метод [37] оказывается несостоятелен, когда углы съемки сильно отличаются, однако остальные подходы показывают достаточно высокое качество распознавания. Наилучших результатов при таких экспериментах также достигает метод, не использующий нейронные сети.

Табл. 4. Сравнение результатов кросс-валидации на базе OU-ISIR
Table 4. Tab. 4. Comparison of cross-validation results with the base OU-ISIR

Метод	0º	10º	20º	30º
Sokolova [26]	98,4%	98,2%	97,1%	94,1%
Shiraga [23]	96,5%	95,8%	92,5%	84,9%
Wu [31]	98,9%	95,5%	92,4%	85,3%
Takemura [27]	99,3%	99,2%	98,6%	96,9%
He [10]	-	96,7%	93,2%	94,2%

Однако при наличии большего количества данных для обучения и тестирования нейросетевые методы оказываются очень успешны. В табл. 4 приведены результаты сравнения алгоритмов при использовании данных для почти 4 тысяч людей.

Благодаря большему размеру обучающей выборки методы, использующие такой протокол тестирования, достигают более высокой точности. Отсутствие открытых реализаций и общего протокола тестирования не дает возможности сравнить все методы и найти оптимальный, однако даже имеющиеся результаты показывают, что на сегодняшний день глубокие и неглубокие подходы продолжают развиваться и показывают практически равное качество.

Табл. 5. Сравнение средних результатов для трех ракурсов базы CASIA
Table 5. Comparison of average results for three views of the CASIA base

Метод	54º	90º	126º
Sokolova [26]	77.8%	68.8%	74.7%
Wu [31]	77.8%	64.9%	76.1%
Feng [9]	52.2%	60.0%	61.9%
Yu, SPAE [34]	63.3%	62.1%	66.3%
Yu, GaitGAN [33]	64.5%	58.2%	65.7%

Для базы CASIA качество распознавания для различных ракурсов тоже, как правило, агрегируется. Следуя распространенному подходу, приведем в табл. 5 средние значения точности распознавания для тестовых ракурсов 54º, 90º и 126º (в качестве обучающих в каждом эксперименте используются оставшиеся 10 углов).

Кроме классической задачи классификации, в которой для человека в видео необходимо определить, кем из базы он является, задачу распознавания по походке часто формулируют в форме задачи верификации. Для пары видеопоследовательностей с двигающимся человеком требуется определить, разные ли люди в кадре или один и тот же.

Задача верификации интересна не только сама по себе, но и как дополнение к идентификации в случае, если человек впервые попадает в поле зрения камер и его еще нет в индексе. Даже отсутствующий в базе человек будет каким-то образом классифицирован идентификатором, и проверка уверенности модели в своём решении – отдельная сложная задача. Один из возможных подходов к решению – дополнительная верификация пары, состоящей из тестового видео и видео с кандидатом в ответы.

Для задачи верификации могут использоваться все описанные методы выделения признаков походки, однако вместо классификации или нахождения ближайшего объекта на последнем этапе оценивается близость пары дескрипторов и сравнивается с некоторым порогом. Достаточно близкие дескрипторы считаются принадлежащими одному человеку. Для оценки качества в такой задаче, как правило, строится ROC-кривая и считается площадь под ее графиком.

Интересно отметить, что несмотря на то, что по сути решается одна и та же задача распознавания по походке и вычисляются одни и те же дескрипторы, подходы, показывающие самые высокие результаты в задаче идентификации, могут оказаться не самыми успешными при оценке качества верификации. На рис. 3 изображены графики ROC-кривых для нескольких методов многоракурсного распознавания на базе OU-ISIR, предоставленные их авторами.

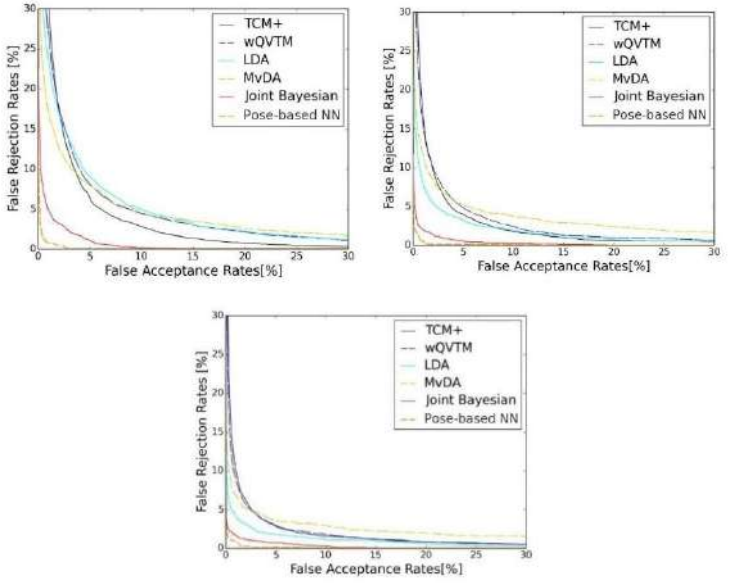


Рис. 3. Сравнение ROC-кривых различных методов для многоракурсной задачи верификации на базе OULP для тестового угла 85° и обучающих углов, равных 55°, 65°, and 75°, соответственно
Рис. 3. Comparison of ROC curves for various methods for a multi-view OULP-based verification task for a test angle of 85° and training angles of 55°, 65°, and 75° respectively

Кривая, которая соответствует подходу [26], отстающему в качестве идентификации от байесовского метода [15], оказывается ниже всех на всех графиках, что показывает, что этот алгоритм более точно определяет, совпадают ли люди в видео. Это в очередной раз

подтверждает, что одного идеального метода до сих пор не существует и разные подходы оказываются лучше при разных условиях и предположениях.

Результаты сравнения методов многоракурсного распознавания показывают, что различные дескрипторы не уступают друг другу в информативности. Методы, основанные на выделении и исследовании силуэтов, решают задачу практически с той же точностью, как и подходы, рассматривающие позу и движение точек в кадре, а иногда и лучше.

6. Заключение

Несмотря на все множество используемых признаков и разнообразие предлагаемых моделей и методов обучения, задача распознавания походки все еще не теряет актуальности: существующие решения пока не достигли идеальной точности идентификации. На представление движения влияет большое количество различных условий, а наборы данных, пригодные для этой задачи, ограничены по сравнению с другими проблемами компьютерного зрения, для которых собраны миллионы изображений лиц или десятки тысяч фигур для реидентификации. Базы данных, собранные на данный момент, пока не способны учесть все возможные вариации походки, что препятствует созданию совершенной модели.

Список литературы

[1] Arseev S., Konushin A., Liutov V. Human Recognition by Appearance and Gait. Programming and Computer Software, vol. 44, issue 4, 2018, pp. 258–265

[2] Khalid Bashir, Tao Xiang, Shaogang Gong. Gait recognition using gait entropy image. In Proc. of the 3rd international conference on crime detection and prevention, 2009, pp. 1–6.

[3] Belhumeur P.N., Hespanha J.P., Kriegman D.J. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 7, 1997, pp. 711–720.

[4] Chen C., Liang J., Zhao H., Hu H., Tian J. Frame difference energy image for gait recognition with incomplete silhouettes. Pattern Recognition Letters, vol. 30, no. 11, 2009, pp 977–984.

[5] Francisco Manuel Castro, Manuel J. Marín-Jiménez, Nicolás Guil, Nicolás Pérez de la Blanca. Automatic learning of gait signatures for people identification. Lecture Notes in Computer Science, vol. 10306, 2017. pp. 257–270.

[6] Francisco M. Castro, Manuel J. Marín-Jiménez, Rafael Medina Carnicer. Pyramidal Fisher Motion for multiview gait recognition. In Proc. of the 22nd International Conference on Pattern Recognition, 2014, pp. 1692–1697.

[7] Dalal N., Triggs B. Histograms of oriented gradients for human detection. In Proc. of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 01. 2005. pp. 886–893.

[8] Deng M., Wang C., Cheng F., Zeng W. Fusion of spatial-temporal and kinematic features for gait recognition with deterministic learning. Pattern Recognition, 2017, 67, pp. 186 – 200

[9] Feng Y., Li Y., Luo J. Learning effective gait features using LSTM. In Proc. of the 23rd International Conference on Pattern Recognition (ICPR), 2016, pp. 325–330.

[10] He Y., Zhang J., Shan H., Wang L. Multitask gans for view-specific feature learning in gait recognition. IEEE Transactions on Information Forensics and Security, vol. 14, no. 1, 2019, pp. 102–113.

[11] Han J., Bhanu B. Individual recognition using gait energy image. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, issue 2, 2006, pp. 316–322.

[12] Hofmann M., Geiger J., Bachmann S., Schuller B., Rigoll G. The TUM Gait from Audio, Image and Depth (GAID) database: Multimodal recognition of subjects and traits. Journal of Visual Communication and Image Representation, vol. 25, no. 1, 2014, pp.195 – 206.

[13] Iwama H., Okumura M., Makihara Y., Yagi Y. The OU-ISIR Gait Database Comprising the Large Population Dataset and Performance Evaluation of Gait Recognition. IEEE Transactions on Information Forensics and Security, vol. 7, issue 5, 2012, pp.1511–1521.

[14] Laptev I., Marszalek M., Schmid C., Rozenfeld B. Learning Realistic Human Actions from Movies. In Proc. of the CVPR 2008 – IEEE Conference on Computer Vision & Pattern Recognition, 2008, pp. 1–8

[15] Li C., Sun S., Chen X., Min X. Cross-view gait recognition using joint Bayesian. In Proc. of the Ninth International Conference on Digital Image Processing (ICDIP 2017), 2017.
[16] Liu Y., Zhang J., Wang C., Wang L. Multiple HOG templates for gait recognition. In Proc. of the 21st International Conference on Pattern Recognition (ICPR2012). 2012. pp. 2930–2933
[17] Makihara Y., Sagawa R., Mukaigawa Y., Echigo T., Yagi Y. Gait recognition using a view transformation model in the frequency domain. Lecture Notes in Computer Science, vol. 3953, 2006, pp. 151–163.
[18] Makihara Y., Suzuki A., Muramatsu D., Li X., Yagi Y. Joint intensity and spatial metric learning for robust gait recognition. In Proc. of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6786–6796
[19] Mansur A., Makihara Y., Muramatsu D., Yagi Y. Cross-view gait recognition using view-dependent discriminative analysis. In Proc. of the 2014 IEEE/IAPR International Joint Conference on Biometrics, 2014.
[20] M.J. Marín-Jiménez, F.M. Castro, N. Guil, F. de la Torre, R. Medina-Carnicer. Deep multitask learning for gait-based biometrics. In Proc. of the IEEE International Conference on Image Processing (ICIP), 2017.
[21] Muramatsu D., Makihara Y., Yagi Y. View transformation model incorporating quality measures for cross-view gait recognition. IEEE transactions on cybernetics, 2016, vol. 46, issue 7, pp. 1602–1615.
[22] Muramatsu D., Makihara Y., Yagi Y. Crossview gait recognition by fusion of multiple transformation consistency measures. IET Biometrics, vol. 4, issue 2, 2015, pp. 62–73.
[23] Shiraga K., Makihara Y., Muramatsu D., Echigo T., Yagi Y. GEINet: View-invariant gait recognition using a convolutional neural network. In Proc. of the 2016 International Conference on Biometrics (ICB), 2016, pp. 1–8.
[24] Simonyan K., Zisserman A. Two-stream convolutional networks for action recognition in videos. In Proc. of the of the 27th International Conference on Neural Information Processing Systems, vol. 1 of NIPS'14, 2014, pp. 568–576.
[25] Sokolova A., Konushin A. Gait recognition based on convolutional neural networks. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. vol. XLII-2/W4, 2017. pp. 207–212.
[26] Sokolova A. and Konushin A. Pose-based Deep Gait Recognition. IET Biometrics, 2018
[27] Takemura N., Makihara Y., Muramatsu D. On input/output architectures for convolutional neural network-based crossview gait recognition. IEEE Transactions on Circuits and Systems for Video Technology, 2017.
[28] Thapar D., Nigam A., Aggarwa, D., Agarwal P. VGR-net: A view invariant gait recognition network. In Proc. of the EEE 4th International Conference on Identity, Security, and Behavior Analysis (ISBA), 2018, pp. 1-8.
[29] Tong S., Fu Y., Ling H., Zhang E. Gait identification by joint spatial-temporal feature. Lecture Notes in Computer Science, vol. 10568, 2017, pp. 457–465.
[30] Whytock T., Belyaev A., Robertson N.M. Dynamic distance-based shape features for gait recognition. Journal of Mathematical Imaging and Vision, vol. 50, no. 3, 2014, pp. 314–326.
[31] Wu Z., Huang Y., Wang L., Wang X., Tan T. A comprehensive study on cross-view gait based human identification with deep cnns. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 2, 2016, pp. 209–226.
[32] Yang Y., Tu D., Li G. Gait recognition using flow histogram energy image. In Proc. of the 22nd International Conference on Pattern Recognition, 2014, pp. 444–449
[33] Yu S., Chen H., Reyes E. B. G., Poh, N. GaitGAN: Invariant Gait Feature Extraction Using Generative Adversarial Network. In Proc. of the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017, pp 532–539
[34] Yu S., Chen H., Wang Q., Shen L., Huang Y. Invariant feature extraction for gait recognition using only one uniform model. Neurocomputing, vol. 239, 2017, pp. 81 – 93
[35] Yu S., Tan D., Tan T. A Framework for Evaluating the Effect of View Angle, Clothing and Carrying Condition on Gait Recognition. In Proc. of the 18th International Conference on Pattern Recognition (ICPR), vol. 4, 2006, pp. 441–444
[36] Zhang C. Liu W., Ma H., Fu H. Siamese neural network based gait recognition for human identification. In Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2016, pp. 2832-2836.
[37] Zhang X., Sun S., Li C., Zhao X., Hu Y. Deepgait: A learning deep convolutional representation for gait recognition. Lecture Notes in Computer Science, vol. 10568, 2017, pp. 447–456.

Methods of gait recognition in video

[1] *A.I. Sokolova <nkgorelits@2100.gosniias.ru>*
[2] *A.S. Konushin <anton.konushin@grapics.cs.msu.ru>*
[1] *National research University Higher School of Economics,*
20, Myasnitskaya st., Moscow, 101000, Russian
[2] *Lomonosov Moscow State University*
Leninskie Gory, Moscow, 119991, Russia

Abstract. Human gait is an important biometric index that allows to identify a person at a great distance without direct contact. Due to these qualities, which other popular identifiers such as fingerprints or iris do not have, the recognition of a person by the manner of walking has become very common in various areas where video surveillance systems can be used. With the development of computer vision techniques, a variety of approaches for human identification by movements in a video appear. These approaches are based both on natural biometric characteristics (human skeleton, silhouette, and their change during walking) and abstract features trained automatically which do not have physical justification. Modern methods combine classical algorithms of video and image analysis and new approaches that show excellent results in related tasks of computer vision, such as human identification by face and appearance or action and gesture recognition. However, due to the large number of conditions that can affect the walking manner of a person itself and its representation in video, the problem of identifying a person by gait still does not have a sufficiently accurate solution. Many methods are overfitted by the conditions presented in the databases on which they are trained, which limits their applicability in real life. In this paper, we provide a survey of state-of-the-art methods of gait recognition, their analysis and comparison on several popular video collections and for different formulations of the problem of recognition. We additionally reveal the problems that prevent the final solution of gait identification challenge.

Keywords: gait; biometrics; silhouette; neural networks; identification

DOI: 10.15514/ISPRAS-2019-31(1)-5

For citation: Sokolova A.I., Konushin A.S. Methods of gait recognition in video. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 1, 2019. pp. 69-82 (in Russian). DOI: 10.15514/ISPRAS-2019-31(1)-5

References

[1] Arseev S., Konushin A., Liutov V. Human Recognition by Appearance and Gait. Programming and Computer Software, vol. 44, issue 4, 2018, pp. 258–265
[2] Khalid Bashir, Tao Xiang, Shaogang Gong. Gait recognition using gait entropy image. In Proc. of the 3rd international conference on crime detection and prevention, 2009, pp. 1–6.
[3] Belhumeur P.N., Hespanha J.P., Kriegman D.J. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 7, 1997, pp. 711–720.
[4] Chen C., Liang J., Zhao H., Hu H., Tian J. Frame difference energy image for gait recognition with incomplete silhouettes. Pattern Recognition Letters, vol. 30, no. 11, 2009, pp 977–984.
[5] Francisco Manuel Castro, Manuel J. Marín-Jiménez, Nicolás Guil, Nicolás Pérez de la Blanca. Automatic learning of gait signatures for people identification. Lecture Notes in Computer Science, vol. 10306, 2017. pp. 257–270.
[6] Francisco M. Castro, Manuel J. Marín-Jiménez, Rafael Medina Carnicer. Pyramidal Fisher Motion for multiview gait recognition. In Proc. of the 22nd International Conference on Pattern Recognition, 2014, pp. 1692–1697.
[7] Dalal N., Triggs B. Histograms of oriented gradients for human detection. In Proc. of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 01. 2005. pp. 886–893.
[8] Deng M., Wang C., Cheng F., Zeng W. Fusion of spatial-temporal and kinematic features for gait recognition with deterministic learning. Pattern Recognition, 2017, 67, pp. 186 – 200

[9] Feng Y., Li Y., Luo J. Learning effective gait features using LSTM. In Proc. of the 23rd International Conference on Pattern Recognition (ICPR), 2016, pp. 325–330.
[10] He Y., Zhang J., Shan H., Wang L. Multitask gans for view-specific feature learning in gait recognition. IEEE Transactions on Information Forensics and Security, vol. 14, no. 1, 2019, pp. 102–113.
[11] Han J., Bhanu B. Individual recognition using gait energy image. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, issue 2, 2006, pp. 316–322.
[12] Hofmann M., Geiger J., Bachmann S., Schuller B., Rigoll G. The TUM Gait from Audio, Image and Depth (GAID) database: Multimodal recognition of subjects and traits. Journal of Visual Communication and Image Representation, vol. 25, no. 1, 2014, pp.195 – 206.
[13] Iwama H., Okumura M., Makihara Y., Yagi Y. The OU-ISIR Gait Database Comprising the Large Population Dataset and Performance Evaluation of Gait Recognition. IEEE Transactions on Information Forensics and Security, vol. 7, issue 5, 2012, pp.1511–1521.
[14] Laptev I., Marszalek M., Schmid C., Rozenfeld B. Learning Realistic Human Actions from Movies. In Proc. of the CVPR 2008 – IEEE Conference on Computer Vision & Pattern Recognition, 2008, pp. 1–8
[15] Li C., Sun S., Chen X., Min X. Cross-view gait recognition using joint Bayesian. In Proc. of the Ninth International Conference on Digital Image Processing (ICDIP 2017), 2017.
[16] Liu Y., Zhang J., Wang C., Wang L. Multiple HOG templates for gait recognition. In Proc. of the 21st International Conference on Pattern Recognition (ICPR2012). 2012. pp. 2930–2933
[17] Makihara Y., Sagawa R., Mukaigawa Y., Echigo T., Yagi Y. Gait recognition using a view transformation model in the frequency domain. Lecture Notes in Computer Science, vol. 3953, 2006, pp. 151–163.
[18] Makihara Y., Suzuki A., Muramatsu D., Li X., Yagi Y. Joint intensity and spatial metric learning for robust gait recognition. In Proc. of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6786–6796
[19] Mansur A., Makihara Y., Muramatsu D., Yagi Y. Cross-view gait recognition using view-dependent discriminative analysis. In Proc. of the 2014 IEEE/IAPR International Joint Conference on Biometrics, 2014.
[20] M.J. Marín-Jiménez, F.M. Castro, N. Guil, F. de la Torre, R. Medina-Carnicer. Deep multitask learning for gait-based biometrics. In Proc. of the IEEE International Conference on Image Processing (ICIP), 2017.
[21] Muramatsu D., Makihara Y., Yagi Y. View transformation model incorporating quality measures for cross-view gait recognition. IEEE transactions on cybernetics, 2016, vol. 46, issue 7, pp. 1602–1615.
[22] Muramatsu D., Makihara Y., Yagi Y. Crossview gait recognition by fusion of multiple transformation consistency measures. IET Biometrics, vol. 4, issue 2, 2015, pp. 62–73.
[23] Shiraga K., Makihara Y., Muramatsu D., Echigo T., Yagi Y. GEINet: View-invariant gait recognition using a convolutional neural network. In Proc. of the 2016 International Conference on Biometrics (ICB), 2016, pp. 1–8.
[24] Simonyan K., Zisserman A. Two-stream convolutional networks for action recognition in videos. In Proc. of the of the 27th International Conference on Neural Information Processing Systems, vol. 1 of NIPS'14, 2014, pp. 568–576.
[25] Sokolova A., Konushin A. Gait recognition based on convolutional neural networks. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. vol. XLII-2/W4, 2017. pp. 207–212.
[26] Sokolova A. and Konushin A. Pose-based Deep Gait Recognition. IET Biometrics, 2018
[27] Takemura N., Makihara Y., Muramatsu D. On input/output architectures for convolutional neural network-based crossview gait recognition. IEEE Transactions on Circuits and Systems for Video Technology, 2017.
[28] Thapar D., Nigam A., Aggarwa, D., Agarwal P. VGR-net: A view invariant gait recognition network. In Proc. of the EEE 4th International Conference on Identity, Security, and Behavior Analysis (ISBA), 2018, pp. 1-8.
[29] Tong S., Fu Y., Ling H., Zhang E. Gait identification by joint spatial-temporal feature. Lecture Notes in Computer Science, vol. 10568, 2017, pp. 457–465.
[30] Whytock T., Belyaev A., Robertson N.M. Dynamic distance-based shape features for gait recognition. Journal of Mathematical Imaging and Vision, vol. 50, no. 3, 2014, pp. 314–326.
[31] Wu Z., Huang Y., Wang L., Wang X., Tan T. A comprehensive study on cross-view gait based human identification with deep cnns. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 2, 2016, pp. 209–226.

[32] Yang Y., Tu D., Li G. Gait recognition using flow histogram energy image. In Proc. of the 22nd International Conference on Pattern Recognition, 2014, pp. 444–449
[33] Yu S., Chen H., Reyes E. B. G., Poh, N. GaitGAN: Invariant Gait Feature Extraction Using Generative Adversarial Network. In Proc. of the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017, pp 532–539
[34] Yu S., Chen H., Wang Q., Shen L., Huang Y. Invariant feature extraction for gait recognition using only one uniform model. Neurocomputing, vol. 239, 2017, pp. 81 – 93
[35] Yu S., Tan D., Tan T. A Framework for Evaluating the Effect of View Angle, Clothing and Carrying Condition on Gait Recognition. In Proc. of the 18th International Conference on Pattern Recognition (ICPR), vol. 4, 2006, pp. 441–444
[36] Zhang C. Liu W., Ma H., Fu H. Siamese neural network based gait recognition for human identification. In Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2016, pp. 2832-2836.
[37] Zhang X., Sun S., Li C., Zhao X., Hu Y. Deepgait: A learning deep convolutional representation for gait recognition. Lecture Notes in Computer Science, vol. 10568, 2017, pp. 447–456.

К разработке открытого программного обеспечения для реконструкции CAD-моделей

С.Е. Сляднев <sergey.slyadnev@gmail.com>
В.Е. Турлапов <vadim.turlapov@itmm.unn.com>
Нижегородский государственный университет им. Н.И. Лобачевского,
603950, Россия, г. Нижний Новгород, пр.Гагарина, 23

Аннотация. Описано открытое программное обеспечение, специализированное для решения задач реконструкции CAD-моделей из дискретного представления. Кратко изложены принципы построения системы, ее архитектура и указаны пути дальнейшего развития. Продемонстрировано использование ПО для автоматической реконструкции лопатки турбины из структурированного облака точек. Продемонстрирован интерактивный метод реконструкции лопатки из триангуляции. В обоих примерах использован метод скиннинга NURBS-поверхности с привлечением оператора сглаживания промежуточных кривых. На примере показано, что модифицированный оператор скиннинга не минимизирует общую энергию деформации модели, но позволяет получить гладкую поверхность, компенсируя погрешность в исходных данных. Результат реконструкции есть параметрическая модель лопатки, переменными проектирования в которой являются координаты контрольных точек профильных кривых. Представленная архитектура открытого ПО может быть использована для произвольного способа как частичной, так и полной параметризации реконструированных CAD-моделей с целью их дальнейшей оптимизации.

Ключевые слова: САПР; реинжиниринг; реконструкция CAD-моделей; скиннинг лопатки турбины; сглаживание

DOI: 10.15514/ISPRAS-2019-31(1)-6

Для цитирования: Сляднев С.Е., Турлапов В.Е. К разработке открытого программного обеспечения для реконструкции CAD-моделей. Труды ИСП РАН, том 31, вып. 1, 2019 г., стр. 83-104. DOI: 10.15514/ISPRAS-2019-31(1)-6

1. Введение

Актуальной инженерной проблемой остается воссоздание точной цифровой модели объекта из неполного описания его геометрической формы [1]. Такие данные получают, как правило, оцифровкой реального изделия, например, при помощи 3D-сканирования. Результат оцифровки подлежит реконструкции для дальнейшей работы в некотором инженерном программном обеспечении. Задачами реконструкции занимается дисциплина, известная как обратный инжиниринг (реинжиниринг). Ключевым этапом промышленного реинжиниринга является моделирование геометрической формы объекта в виде его граничного представления (boundary representation, B-Rep) [2], которое, на сегодняшний день, является стандартом де факто для систем автоматизированного проектирования (САПР). Как отмечают авторы обзора [1], CAD-модель (в граничном представлении) является своеобразным «языком», на котором общаются различные средства автоматизации проектирования и производства.

В инженерной практике задачи реконструкции имеют различные истоки. Например, если изделие создавалось без использования САПР, то его цифровое представление отсутствует

изначально. Иногда геометрическая модель оказывается недоступна в силу иных причин, таких как устаревание или исчезновение оригинальной CAD-системы. В таких случаях целью реинжиниринга является получение ранее не существовавшей или утраченной цифровой модели. Восстановленная геометрия может использоваться для дальнейшего редактирования CAD-модели, автоматизированного изготовления детали на станке с ЧПУ, а также для решения расчетных задач (исследование прочности, термический анализ, аэродинамический анализ и т.д.) или параметрической оптимизации формы. Если параметрическая оптимизация осуществляется на полигональной модели, то проблема реинжиниринга может возникать при обратном переходе от деформированной сетки [3] к CAD-представлению.

Подходы обратного инжиниринга находят применение и в качестве основного средства моделирования еще не существующих объектов [4]. Так, в автомобильной промышленности часто используется техника оцифровки физического макета, выполненного скульптором для демонстрации внешних обводов кузова. Иногда геометрическое моделирование выполняется изначально в полигональном виде, после чего реинжиниринг обеспечивает переход к точному граничному представлению [5]. В последнем случае, техники реконструкции играют в сущности ту же роль, что и традиционные техники моделирования «снизу-вверх» с той разницей, что вместо работы «с чистого листа» используется вспомогательная полигональная «подложка». Так или иначе, в задачах реконструкции инженер имеет дело с некоторым опорным геометрическим представлением в виде неупорядоченного или структурированного облака точек, либо поверхностной триангуляции. Опорная геометрия направляет процесс моделирования и служит для сопоставления целевой и воссозданной эталонной формы объекта.

В зависимости от целей обратного инжиниринга, можно рассмотреть различные критерии успешности процесса реконструкции. Например, при создании цифровой копии физического макета, требуется обеспечить минимальную геометрическую невязку между CAD-моделью и ее полигональным представлением. В автомобильной промышленности также предъявляются строгие требования к эстетическим качествам изделия, поэтому в процессе моделирования должны использоваться т.н. поверхности «класса А». С другой стороны, при воссоздании цифровой модели поврежденной детали, либо детали с большим износом, высокая точность реконструкции не только не требуется, но является нежелательной в тех пространственных зонах, где реальная форма существенно расходится с эталонной. Минимизация исключительно геометрической невязки приводит к тому, что результирующая модель оказывается лишенной таких ключевых качеств, как, например, осевая симметрия, параллельность или ортогональность граней и т.п. Таким образом, обратный инжиниринг есть многокритериальный процесс, в котором набор и строгость привлекаемых критериев регулируется в зависимости от конкретного индустриального приложения.

Для решения задач обратного проектирования используют специализированное программное обеспечение или расширения традиционных САПР. Сложности, возникающие в процессе реконструкции, привели к необходимости всестороннего научного исследования для выработки точных, надежных и эффективных алгоритмов реинжиниринга. В то же время, как показывает инженерная практика, для полноценного решения хотя бы узкого класса задач реконструкции, одного алгоритма недостаточно. Возникает потребность в согласованном применении множества геометрических операторов и реализации соответствующей объектной модели для представления предметной области и организации вычислений. К сожалению, в современной литературе уделяется мало внимания архитектурам программных пакетов, предназначенных для реконструкции. С точки зрения авторов, недостаток информации связан с тем, что практически все продукты автоматизированного реинжиниринга в области САПР являются коммерческими, и производители данных систем не заинтересованы в раскрытии деталей их реализации.

Основным вкладом настоящей работы является создание открытой архитектуры программного обеспечения для решения задач реинжиниринга CAD-моделей из сеточного представления. Цель данной статьи состоит в том, чтобы дать общие сведения о принципах функционирования разработанной системы и направлениях ее расширения. Вслед за авторами [6], мы считаем, что публикация систем с открытыми исходными кодами способствует снятию барьеров, препятствующих независимому воспроизведению научных результатов. Кроме того, реализованные средства прототипирования позволяют использовать предлагаемую систему для обмена идеями и переноса технологий от академических исследователей к индустрии, как это указано в [7].

Статья организована следующим образом. В разд. 2 описаны существующие методы реконструкции CAD-моделей и контурно даны некоторые признаки для их классификации. Рабочие гипотезы, установленные для предлагаемой системы, вводятся в разд. 3. Там же даются основные этапы процедуры интерактивной реконструкции. Разд. 4 содержит описание принципов, лежащих в основе разработанного ПО. Использование системы реконструкции для воссоздания параметрической модели лопатки турбины описано в разд. 5. В заключении (разд. 6) приведены некоторые направления, выбранные нами для дальнейшего развития представленного открытого ПО.

2. Состояние проблемы

Авторы обзора [8] отмечают, что исследования в области обратного инжиниринга часто обходят стороной вопрос о программном обеспечении, реализующем тот или иной подход к реконструкции. В то же время, именно наличие и востребованность программного обеспечения (в том числе коммерческого) позволяет судить о практической применимости опубликованных подходов и алгоритмов, их точности, надежности и эффективности. Обзор [8] содержит сравнение существующего программного обеспечения в смысле полноты реализуемых операторов обратного инжиниринга. В то же время, все рассматриваемые системы являются коммерческими, что оказывается существенным препятствием для их использования в научной среде. Так, возникают следующие проблемы. Во-первых, в подавляющем большинстве случаев коммерческие системы не раскрывают библиографической базы, указывающей на конкретные реализованные подходы к реконструкции. Во-вторых, оригинальные работы, выполненные в рамках коммерческих компаний, редко доступны через публичные средства научной коммуникации, поскольку предпочтение отдается производству внутренних технических отчетов. Солидарно с автором [9], мы считаем, что действительно научный подход к проблемам реинжиниринга должен базироваться на принципах «открытой науки» (open science), предполагающих публикацию не только научных статей, но также исходных кодов и входных данных для алгоритмов.

Автор обзора [10] описывает подходы к реконструкции CAD-моделей, реализованные в широко используемом коммерческом программном обеспечении. В частности, указано, что интерактивные методы, задействующие пользователя для ручной сегментации сеток, остаются востребованными в индустрии, несмотря на их трудоемкость. Кроме того, согласно [10], интерактивная реконструкция с пользовательской сегментацией позволяет добиться наилучшего качества результирующих поверхностей.

Особое место занимают подходы, позволяющие реконструировать не единственный экземпляр геометрической модели, но параметрическое семейство $S(\mathbf{a})$, где $\mathbf{a} \in \mathbb{R}^N$ – вектор переменных проектирования, а N – размерность пространства параметров. Параметрическая модель является не просто описанием формы, но своеобразным генератором моделей, сохраняющим в их структуре т.н. «замысел проектировщика» (design intent) для различных значений вектора $\mathbf{a}$. Подходы к реконструкции, восстанавливающие не только «геометрический артефакт», но инженерную концепцию (замысел проектировщика), доминируют, например, в области турбостроения [11,12]. Развиваются методы реинжиниринга параметрических моделей машиностроительных деталей. Например, авторы современной работы [13] предлагают

эффективный метод синтеза CSG-дерева в виде программы, выполняющей булевы операции на предопределенном множестве примитивов, распознанных из полигональных сеток. Метод декомпозиции граничных моделей на элементы объемов изъятия был предложен нами в [14]. В работе [15] представлена серия подходов к реконструкции параметрических моделей на базе коммерческой системы SolidWorks. С точки зрения программной реализации, в системах реинжиниринга параметрических моделей интерес представляют не только геометрические операторы, но и модель организации связанных данных.

Большинство методов, публикуемых по поводу реконструкции, ограничивается воссозданием «немой» формы изделия (dumb model), которая лишена конструктивных элементов и потому не допускает полноценного редактирования. В частности, для моделей без истории построения решение задач параметрической оптимизации оказывается затруднено. В обзоре [16] указаны два способа параметризации функциональных (сообразующихся с аэродинамическим потоком) моделей: частичная и полная. Отсутствие декомпозиции модели на конструктивные элементы оставляет возможность лишь частичной параметризации.

В зависимости от конкретного программного обеспечения, могут привлекаться разнообразные схемы геометрического представления. Желательно, чтобы модель, полученная в результате реконструкции, могла быть передана в распространенные САПР для дальнейшей работы, например, последующего редактирования или технологической подготовки производства. На сегодняшний день, lingua franca систем автоматизированного проектирования является формат STEP (ISO 10303), регламентирующий в качестве основного граничное представление моделей [2]. Универсальной формой, позволяющей моделировать как каноническую геометрию, так и «скульптурные» поверхности, является NURBS. В то же время, в процессе реконструкции могут использоваться иные схемы представления, например, поверхности подразделения [17,18] или T-сплайны [19], с той оговоркой, что в конце моделирования они будут преобразованы в стандартный вид. Так, популярными средствами моделирования с использованием поверхностей подразделения, совместимыми с технологией NURBS, являются дополнения Clayoo и Xirus для CAD-системы Rhinoceros, а также, например, плагин Power Surfacing для ПО SolidWorks.

3. Процедура реконструкции

В данном разделе описана интерактивная процедура реконструкции, предлагаемая нами в качестве основной для реализуемой системы. Оговоримся, однако, что представленное ПО может использоваться и для полностью автоматического восстановления параметрических моделей, что мы демонстрируем в разд. 5.

3.1. Основные этапы

Процесс реконструкции начинается с получения исходных данных в виде пространственной триангуляционной сетки. Распространенным форматом для хранения и передачи сеточных данных в области САПР является STL, реализующий представление полигональной геометрии в простейшем виде, без топологии. Сеточные данные нередко требуют дополнительной обработки, такой как заполнение отверстий, избавление от самопересечений, упрощение путем децимации, сглаживание и т.п. Хотя подобная обработка не имеет прямого отношения к реконструкции, соответствующие геометрические операторы, как правило, являются компонентами систем реинжиниринга, поскольку без них процесс реконструкции оказывается затруднен. Широкие возможности для работы с полигональными моделями предоставляются библиотекой научной визуализации VTK [28], которая задействована в нашей системе.

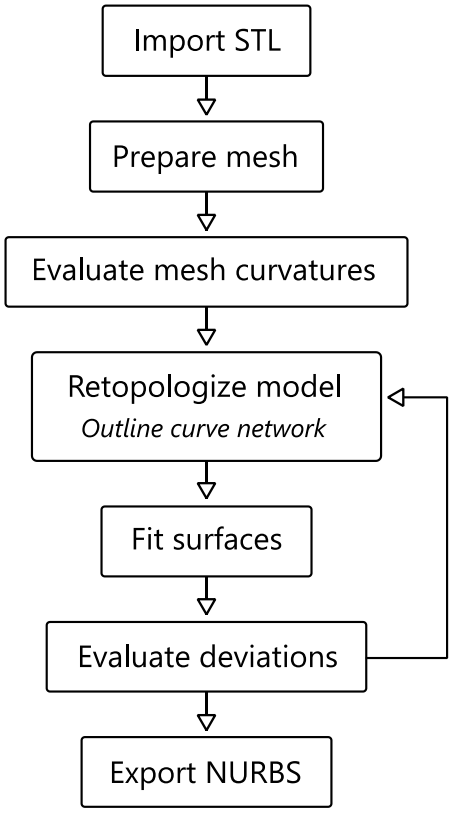


Рис. 1. Основные этапы процесса реконструкции объекта свободной формы в разрабатываемой системе

Fig. 1. The main stages of the process of reconstruction of a free-form object in the developed system

При сегментации модели необходимо учитывать расположение предполагаемых конструктивных линий. С этой целью, система реинжиниринга позволяет оценивать кривизну исходной сетки [29]. Данные о кривизнах играют вспомогательную роль при интерактивном задании топологии пользователем. После того, как топология модели определена, система осуществляет построение геометрических примитивов по выбору пользователя. Здесь привлекаются различные операторы, такие как построение поверхностей Кунса, Гордона, интерполяция упорядоченного облака точек, скиннинг и т.п. На заключительном этапе пользователь оценивает качество реконструкции и, в случае удовлетворительного результата, завершает процесс, либо возвращается на предыдущие этапы для редактирования топологической модели и ассоциированных параметров. Основные этапы реконструкции объекта свободной формы показаны на рис. 1. Заметим, что реинжиниринг в целом не является последовательным процессом, поскольку для достижения удовлетворительного результата, пользователю приходится возвращаться к предыдущим этапам.

В случае, если реконструируется типовая машиностроительная деталь, подход к реинжинирингу, как правило, отличается от изложенного выше. Помимо использования других методов сегментации и реконструкции, важной особенностью процедуры восстановления машиностроительной детали является наличие этапа распознавания конструктивных элементов [30]. Последнее необходимо для получения редактируемой модели, отвечающей предполагаемому инженерному замыслу. В нашей системе реализованы базовые средства распознавания конструктивных элементов, использующие классический принцип анализа атрибутированного графа смежности граней [31].

3.2. Топологическая модель

Одним из ключевых этапов обратного инжиниринга является сегментация полигональной поверхности. В качестве основной рабочей гипотезы мы принимаем, что сегментация осуществляется пользователем интерактивно, в результате чего на сеточной модели задается топология (рис. 2). Данный подход был описан в обзоре [32] применительно к сегментации моделей свободной формы.

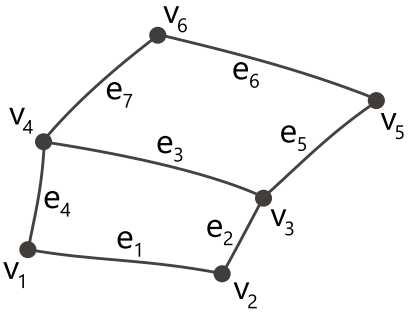


Рис. 2. Сеть кривых, формирующая топологию полигональной модели (отмечены вершины и ребра)
Fig. 2. A network of curves that forms the topology of a polygonal model (vertices and edges are marked)

Известны примеры успешной реализации метода интерактивной реконструкции по сети кривых в коммерческом программном обеспечении [21]. Пользователь осуществляет разметку модели на порции, руководствуясь собственными практическими соображениями и, возможно, картой распределения дискретных кривизн. В простейшем случае, система может ограничивать пользователя созданием четырехугольных криволинейных ячеек, поскольку они допускают затягивание порциями NURBS в натуральных границах [33]. Последнее обстоятельство позволяет обойтись без использования обрезанных поверхностей (trimmed surfaces), и, как следствие, упростить задействованные структуры данных и алгоритмы (например, гладкого сопряжения).

Сеть кривых не только обеспечивает сегментацию полигональных данных, но служит носителем атрибутов, управляющих процессом реконструкции. В частности, ребра модели могут хранить информацию о желаемом порядке гладкости стыка, маркеры силуэтных линий и т.п. Кроме того, наличие сети кривых определяет сам способ редактирования модели, состоящий в модификации опорных ребер и перестроении затронутых поверхностей.

Каждое ребро каркасной модели имеет натуральную (геометрическую) ориентацию, определяемую параметризацией соответствующей кривой. Для обеспечения единообразного порядка обхода контура, порция каркасной модели (patch) содержит не сами ребра, но их вхождения со знаком ориентации (Рис. 3). Таким образом, каркасная модель представляет собой направленный иерархический граф (рис. 4).

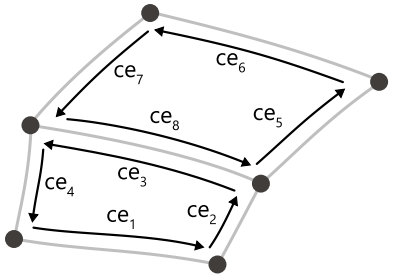


Рис. 3. Ориентированные полуребра каркасной модели
Fig. 3. Oriented half-edges of the frame model

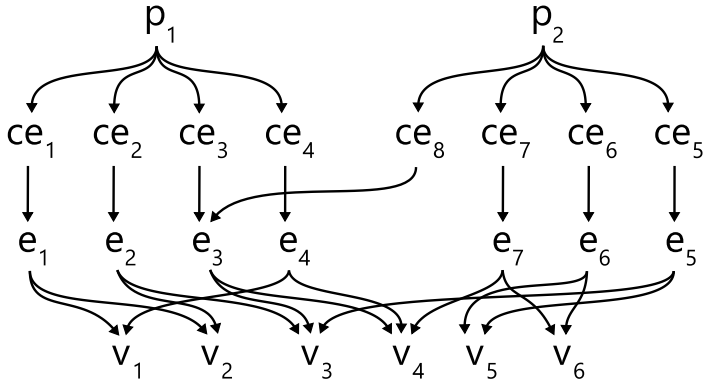


Рис. 4. Топология каркасной модели задается иерархическим ориентированным графом. Узлы p_1 и p_2 соответствуют двум реконструируемым лоскутам, изображенным на рис. 2
Fig. 4. The topology of the framework model is defined by a hierarchical oriented graph. Nodes p_1 and p_2 correspond to two reconstructed flaps shown in fig. 2

Структура каркасной модели определяет схему представления эквивалентную B-Rep в смысле разделения «геометрии» и «топологии», как описано в монографии [34]. Для полноценного моделирования с использованием сети кривых, система должна предоставлять базовый набор операторов редактирования топологии, включающий функции разделения кривых и лоскутов, слияния границ, завершения контура и т.п. В работе [35], посвященной интерактивной системе реинжиниринга, содержатся сведения о достаточном наборе топологических операторов для редактирования поверхностей подразделения. Те же подходы применимы и в схеме представления, основанной на NURBS.

3.3. Операторы реконструкции

На текущий момент в нашей системе реализованы следующие геометрические операторы, не считая алгоритмов, изначально доступных в библиотеках OpenCascade и VTK:

1. интерполяция кривых и поверхностей (алгоритмы A9.1, A9.4 из [36]);
2. скиннинг поверхности (согласно описанию в главе 10.3 монографии [36]);
3. сглаживание кривых и поверхностей (согласно идеям, изложенным в [37]);
4. построение лоскутов Кунса в виде поверхностей NURBS [33];
5. локальное редактирование NURBS-кривых [38];
6. проецирование отрезка прямой на триангуляцию для моделирования сети кривых;
7. построение огибающих облака точек на плоскости [39].

3.4. Создание граничной модели

После моделирования лоскутов, заполняющих криволинейные ячейки сети кривых, в структуре целевого объекта оказывается полностью определена геометрическая и топологическая информация. Построение итоговой граничной модели сводится к преобразованию структур данных в схему представления геометрического ядра OpenCascade для последующей трансляции в нейтральные форматы (STEP, IGES).

4. Архитектура системы

Ниже мы описываем базовые принципы реализации открытой платформы реконструкции.

4.1 Объектная модель

Центральным объектом системы является Деталь (Part), подлежащая реконструкции. Деталь содержит результирующую CAD-модель в граничном представлении библиотеки OpenCascade. Входом системы является поверхностная сетка, загружаемая в объект Триангуляция (Triangulation) из файла формата STL (рис. 5).

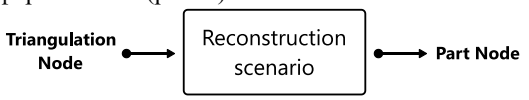


Рис. 5. Входные и выходные данные системы
Fig. 5. System Input and Output

Объекты модели данных и взаимосвязи между ними организованы при помощи расширения модуля OCAF библиотеки OpenCascade [40]. Этот модуль реализует многоцелевое иерархическое хранилище данных, обеспечивающее такие фундаментальные сервисы, как транзакционность, двунаправленный откат изменений (undo/redo), сохранение в файл и чтение из файла, операции CRUD (Create, Read, Update, Delete). Доступ к данным осуществляется с использованием шаблона проектирования DAO (Data Access Object).

Каждый объект модели данных связан с единственным экземпляром класса Презентация (Presentation), содержащим один или несколько Конвейеров (Pipeline) визуализации VTK (Рис. 6).

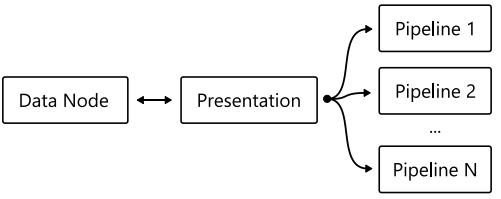


Рис. 6. Объект данных и класс Презентация, содержащий набор Конвейеров
Fig. 6. The data object and class Presentation containing a set of Pipelines.

Конвейер в нашей системе является адаптацией принципа конвейера (pipeline) из библиотеки VTK с дополнительными оговорками:

a) конвейер в нашей системе имеет источником данных объект OCAF, сохраняющий время последней модификации для эффективной перерисовки;
b) предполагается, что один конвейер отвечает единственному объекту сцены.

4.2 Визуализация

Для работы с CAD-данными была реализована подсистема визуализации, переводящая криволинейное представление CAD-модели в полигональную форму, а также соответствующие интерактивные сервисы на базе библиотеки VTK. В распространенных геометрических ядрах, таких как OpenCascade или Parasolid, граничная модель содержит ошибки, связанные с ограниченным машинным представлением вещественных чисел, а также погрешности аппроксимации и неточности геометрических построений. На практике эти погрешности хранятся в структуре CAD-модели в виде геометрических допусков, локально ассоциированных с ее граничными элементами [41].

Хотя существующие САПР, как правило, скрывают дефекты модели визуально, само наличие ошибок может привести к неверной работе операторов моделирования или проблемам при передаче данных из одной инженерной системы в другую. В этой связи, подсистема визуализации нашей платформы была реализована таким образом, чтобы обнажать геометрические и топологические дефекты, делая их очевидными для пользователя. Для B-Rep и сеточных моделей автоматически распознаются и маркируются цветом такие дефекты, как открытые и немногообразные (non-manifold) ребра, разомкнутые грани, «висячие» (dangling) вершины и проч. Акцентирование аномалий позволяет избежать их возникновения еще на этапе

моделирования, тем самым предотвращая необходимость использования операторов «лечения» модели [42] перед ее дальнейшим использованием.

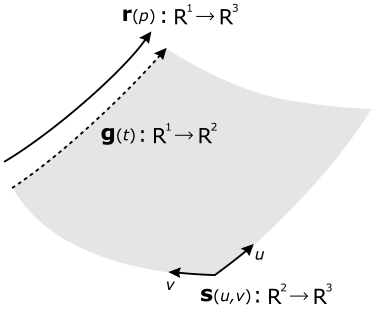


Рис. 7. Представление ребра модели при помощи двух кривых (в пространстве моделирования и в области определения несущей поверхности)
Fig. 7. Representation of the edge of the model using two curves (in the modeling space and in the domain of the bearing surface)

Большие значения геометрических допусков возникают, как правило, из-за дефектов на стыке между гранями модели. Последние нередко связаны с наличием несогласованных ребер на смежных контурах, причем рассогласование возможно между параметрической кривой $\mathbf{r}$ в пространстве моделирования и ее собственным прообразом $\mathbf{g}$ в области определения несущей поверхности $\mathbf{s}(u, v)$ (рис. 7). Возникает требование синхронной параметризации, которое формализуется следующим образом:

$$\left\|\mathbf{s}\left(\mathbf{g}(t)\right) - \mathbf{r}(t)\right\| < \varepsilon, \qquad (1)$$

где ε – геометрический допуск, параметр t принимает значения на отрезке $[t_0, t_1]$, а в качестве нормы $\|\cdot\|$ выбрано евклидово расстояние. Геометрическое моделирование с гарантированным соблюдением правила (1) упрощается, если структура данных модели организована на сети трехмерных кривых, задающих топологию модели (см. Раздел 3.2). В этом случае, поверхностное моделирование обходится без использования параметрических кривых $\mathbf{g}(t)$ вплоть до момента, когда модель преобразуется в структуры данных ядра OpenCascade.

Кроме того, в системе доступны средства визуализации графов, а также инструменты анализа дифференциальных свойств кривых и поверхностей. Подробнее данный инструментарий был описан в нашей предыдущей работе [43].

4.3 Геометрические ядра

Основным ядром геометрического моделирования, используемым в нашей системе, является библиотека OpenCascade [44]. Структуры данных этой библиотеки служат для представления CAD-моделей в виде, пригодном для визуализации, базового моделирования и обмена посредством открытых трансляторов (STEP, IGES). Разработка новых операторов моделирования осуществляется в рамках независимой NURBS-библиотеки, реализуемой в соответствии с широко известной монографией [36].

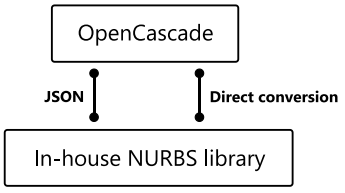


Рис. 8. Взаимодействие между двумя задействованными геометрическими ядрами
Fig. 8. Interaction between two involved geometric cores

Передача данных между двумя библиотеками возможна следующими путями: при помощи формата JSON (JavaScript Object Notation) и с использованием специализированного программного модуля для соответствующего преобразования кривых и поверхностей (Рис. 8). Собственная библиотека геометрического моделирования адаптирована для решения задач обратного инжиниринга, тогда как компоненты OpenCascade играют роль геометрического ядра общего назначения. Похожие архитектурные решения прослеживаются, например, в работе [45].

```
{
   entity: surface,
   type: b-surface,
   continuity: C0,
   domain: {
      U_min: 0,
      U_max: 1,
      V_min: 0,
      V_max: 1
   },
   ...
   properties: {
      U_degree: 1,
      V_degree: 1,
      U_knots: [0, 0, 0.5, 1, 1],
      V_knots: [0, 0, 0.5, 1, 1],
      num_poles_in_U_axis: 3,
      num_poles_in_V_axis: 3,
      poles: {
         u0: [[-10, -10, 0], [-10, 0, 0], [-10, 10, 0]],
         u1: [[0, -10, 0], [0, 0, 10], [0, 10, 0]],
         u2: [[10, -10, 0], [10, 0, 0], [10, 10, 0]]
      }
   }
}
```

Рис. 9. Фрагмент определения B-поверхности первой степени в формате JSON
Fig. 9. Fragment of determining the B-surface of the first degree in JSON format

Формат JSON (Рис. 9) является нейтральным по отношению к обеим библиотекам. С его помощью реализованы процедуры модульного тестирования.

4.4 Командная строка

На данный момент система предполагает использование командной консоли в качестве главного инструмента взаимодействия с пользователем. В текущей версии доступно более ста команд, представляющих различные операции для работы с моделью. Разработку эргономичных элементов графического интерфейса мы оставляем на будущее.

5. Приложения

5.1 Автоматическая реконструкция лопатки

Одной из задач, решенных с использованием представленной системы, является воссоздание параметрической модели рабочего колеса турбины из дискретно заданных линий тока [12]. На рис. 10 показана геометрическая модель лопатки, которая на следующем этапе реплицируется в осевом направлении ротора и объединяется с моделью рабочего колеса турбины при помощи твердотельной булевой операции.

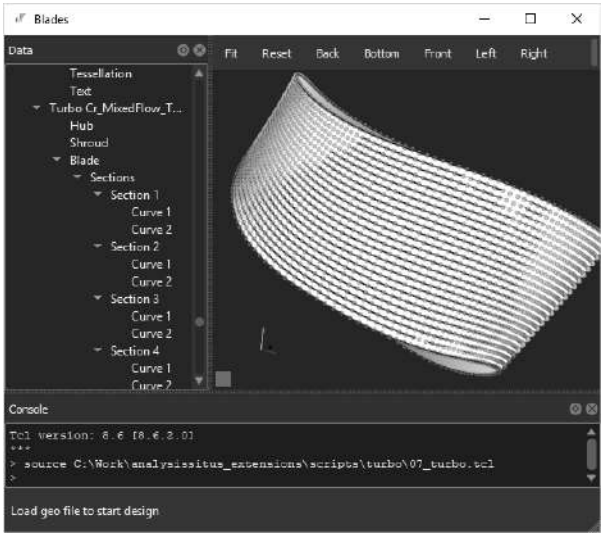


Рис. 10. Твердотельная модель лопатки турбины
Fig. 10. Solid-state model of a turbine blade

Лопатка турбины является примером функциональной поверхности, для моделирования которой используются лоскуты NURBS. В то же время, для решения задач оптимизации формы, в геометрической модели лопатки должна присутствовать параметрическая структура, набранная, как правило, из ее радиальных сечений. Каждое сечение параметризуется определенным образом, предоставляя, тем самым, некоторый набор переменных проектирования, формирующий пространство поиска оптимальной геометрии. В работе [12] параметрами проектирования являлись координаты контрольных точек сечений, однако та же архитектура организации вычислений применима для иных способов параметризации.

Ключевым отличием процедуры реинжиниринга от прямого моделирования «снизу-вверх» является необходимость устранения шума, присутствующего в исходных данных. С этой целью, мы реализовали технику сглаживания [37] и применили ее к набору промежуточных интерполяционных кривых, используемых для получения итоговых контрольных точек поверхности скиннинга.

5.2 Интерактивная реконструкция лопатки

Процесс реинжиниринга, описанный выше, осуществляется полностью автоматически благодаря тому, что исходное облако точек является структурированным и упорядоченным. Так, сечения лопатки изначально выделены в структуре облака отдельными срезами, принадлежащими некоторым поверхностям вращения, образующие которых также реконструируются без участия пользователя. Если же опорная геометрия лопатки представлена триангуляцией, то автоматическая реконструкция оказывается затруднена. Ниже мы описываем интерактивную процедуру восстановления геометрии пера лопатки для таких случаев.

Процедура начинается с выбора секущих плоскостей для извлечения точек, формирующих профильные кривые лопатки. Пересечение триангуляции с плоскостью выполняется при помощи библиотеки VTK, после чего результирующий набор точек сортируется в порядке следования алгоритмом построения огибающей [39], реализованным в нашей системе (рис. 11).

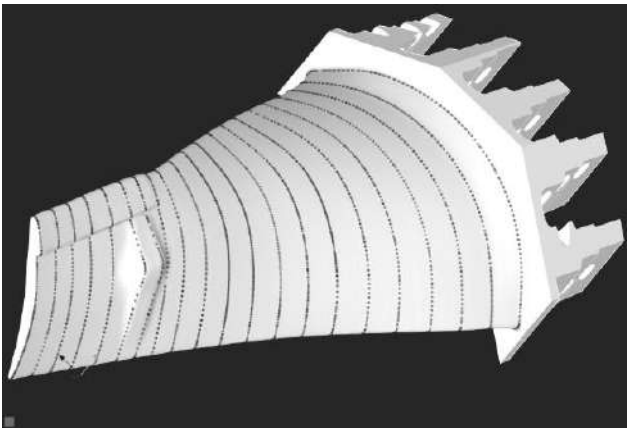


Рис. 11. Сечения лопатки, построенные путем пересечения триангуляции с интерактивно позиционируемой плоскостью

Fig. 11. Blade cross-sections constructed by intersecting triangulation with an interactively positioned plane

Упорядоченные точки приближаются методом наименьших квадратов для получения B-кривых с заданной точностью. Для дальнейшего скиннинга кривые сечений должны быть совместимы [46], однако предыдущие шаги реконструкции этого не гарантируют. Более того, приведение сечений в совместимый вид путем унификации узлов, является практически нецелесообразным, так как итоговая поверхность оказывается чрезмерно сложной (рис 12).

Как замечают авторы [47], сокращение объема данных, представляющих геометрию лопатки, позволяет не только ускорить процедуру реконструкции, но также получить более гладкую поверхность и уменьшить расход памяти. Добавим также, что неоправданно высокое количество и концентрация контрольных точек сплайна затрудняет процесс параметрической оптимизации. Более того, избыточная сложность геометрического представления негативно сказывается на надежности последующих операторов моделирования, например, пересечения поверхностей.

Простой способ понизить сложность итоговой поверхности состоит в том, чтобы избежать процедуры унификации профильных кривых, задавая их на едином узловом векторе изначально. Для этого исходные кривые дискретизируются с заданным количеством точек, после чего интерполируются с использованием центростремительной параметризации [48]. Результирующая поверхность имеет значительно меньшую сложность, однако остаются дефекты формы (рис. 13).

Для устранения эффекта «скручивания» лопатки, каждое сечение предварительно сегментируется. Мы используем интерактивный подход к сегментации, однако этот процесс допускает автоматизацию [49].

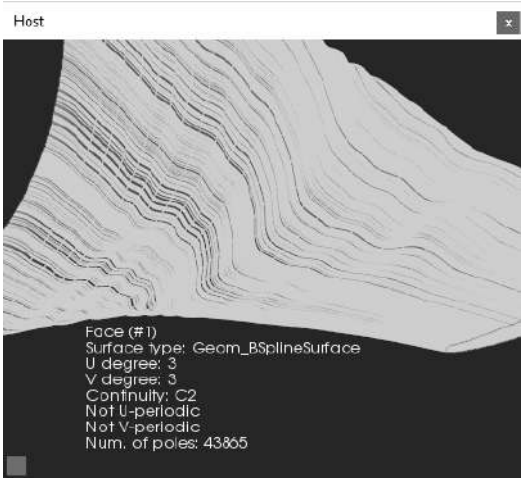


Рис. 12. Поверхность лопатки, построенная скиннингом после унификации профильных кривых. Изолинии отвечают узловым значениям базисных сплайнов. Видны характерные сгущения. Поверхность содержит более 43 тыс. контрольных точек
Fig. 12. The surface of the blade built by skinning after the unification of the profile curves. The isolines correspond to the node values of the basis splines. Visible characteristic thickening. The surface contains more than 43 thousands of control points

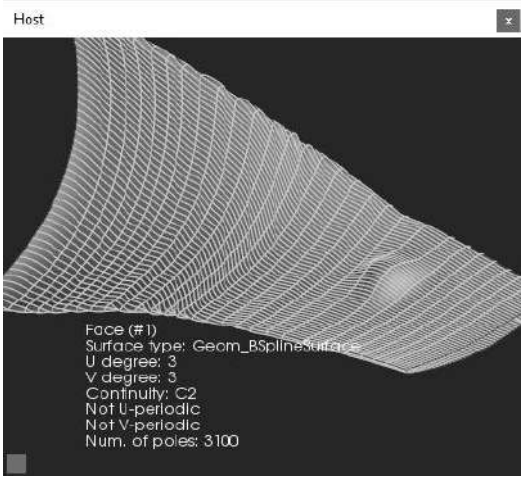


Рис. 13. Поверхность, построенная скиннингом на изначально совместных кривых (содержит 3100 контрольных точек). Виден эффект «бокового скручивания» из-за асинхронной параметризации профильных кривых
Fig. 13. A surface constructed by skinning on initially joint curves (contains 3,100 control points). The effect of «side twisting» is visible due to asynchronous parametrization of profile curves

Исходные данные реконструкции, как правило, содержат погрешность, связанную с несовершенством метода оцифровки. Кроме того, конструктивные линии, определенные пользователем для сегментации сторон лопатки, могут оказаться непреднамеренно извилистыми из-за ошибок ввода или разреженности облака точек. Для устранения искажений формы мы используем оператор скиннинга со сглаживанием [12]. Идея оператора состоит в том, чтобы изменить пространственное расположение контрольных точек промежуточных кривых скиннинга для минимизации энергии деформации [50]. Регулировка коэффициента сглаживания

$\lambda > 0$ позволяет получить более гладкую поверхность ценой увеличения геометрической невязки (рис. 14).

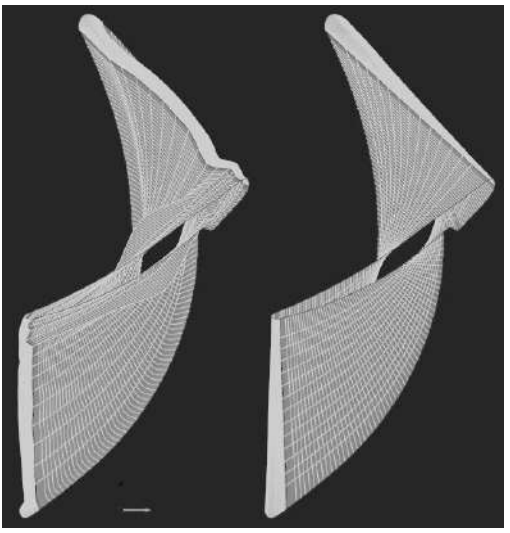


Рис. 14. Скиннинг без сглаживания (слева) и со сглаживанием при значении коэффициента $\lambda = 1$
Fig. 14. Skinning without smoothing (left) and with smoothing at a value of the coefficient $\lambda = 1$

Итак, основными шагами процесса интерактивной реконструкции являются:

1) сечение триангуляции плоскостью и сортировка точек путем построения их огибающей по K ближайшим соседям [39];
2) сегментация каждого сечения с целью разделения сторон лопатки;
3) аппроксимация точек сечения с заданным допуском;
4) вторичная интерполяция регулярного облака, построенного дискретизацией профильных кривых [51], с целью их задания на едином узловом векторе;
5) объединение кривых в замкнутые контуры;
6) скиннинг со сглаживанием [12].

Для оценки 9воздействия коэффициента сглаживания на форму лопатки мы использовали приближенное значение энергии деформации E, рассчитанное по формуле (2).

$$E = \iint \left(\frac{\partial^2 \mathbf{s}}{\partial u^2}\right)^2 + 2\left(\frac{\partial^2 \mathbf{s}}{\partial u \partial v}\right)^2 + \left(\frac{\partial^2 \mathbf{s}}{\partial v^2}\right)^2 dudv \quad (2)$$

Здесь оператор $(\cdot)^2$ есть скалярный квадрат, а интегрирование осуществляется по всей области определения поверхности $\mathbf{s}(u, v)$.

Экспериментально полученная зависимость энергии деформации от значения коэффициента сглаживания показана на рис. 15.

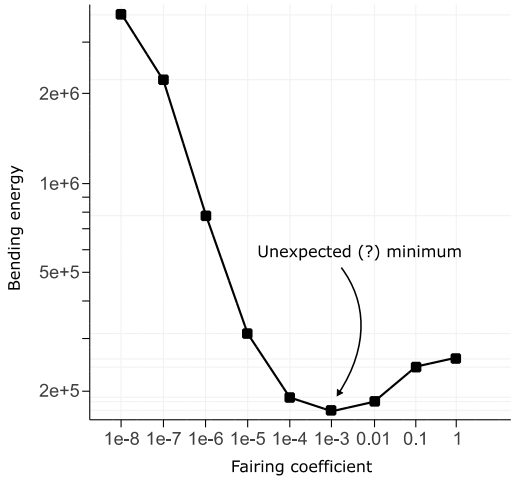


Рис. 15. Зависимость приближенной энергии деформации от коэффициента сглаживания λ
Fig. 15. Dependence of the approximate strain energy on the smoothing coefficient λ

Из рис. 15 видно, что зависимость между энергией деформации поверхности и коэффициентом сглаживания не есть монотонно убывающая функция. Действительно, функционал энергии (2) не является целевым для процедуры сглаживания направляющих скиннинга. Согласно подходу [12], сглаживание кривой $\mathbf{c}(u)$ гарантирует минимизацию лишь функционала (3), агрегирующего геометрическую невязку $\mathbf{c} - \mathbf{c_0}$ с энергией изгиба.

$$E_c = \int \{\lambda[\mathbf{c}'']^2 + [\mathbf{c} - \mathbf{c_0}]^2\}\, du \qquad (3)$$

Убедимся, что в результате скиннинга со сглаживанием, кривизна поверхности лопатки может возрасти. Табл. 1 показывает рост абсолютного значения наименьшей средней кривизны с увеличением значения коэффициента сглаживания λ.

Добавим, что для получения более гладкой формы уместно применение оператора сглаживания к исходным профильным кривым, а не только к направляющим скиннинга.

Табл. 1. Значения максимальной и минимальной средних кривизн в окрестности локального минимума энергии деформации (рис. 15)
Tab. 1. The values of the maximum and minimum mean curvature in the vicinity of the local minimum of the deformation energy (fig. 15)

λ	Mean curvature (min)	Mean curvature (max)
1e-4	**-0.492**	**1.25**
1e-3	**-0.492**	**1.21**
0.01	**-0.493**	**1.21**
0.1	**-0.539**	**1.22**
1	**-0.557**	**1.22**

В заключение проанализируем динамику роста геометрической невязки с увеличением коэффициента сглаживания. График зависимости максимальной геометрической невязки (в единицах измерения модели) показан на рис. 16. Как и следовало ожидать, при достижении некоторого значения λ расстояние между исходной триангуляцией и сглаженной поверхностью перестает изменяться (направляющие скиннинга приобретают минимально возможную энергию изгиба).

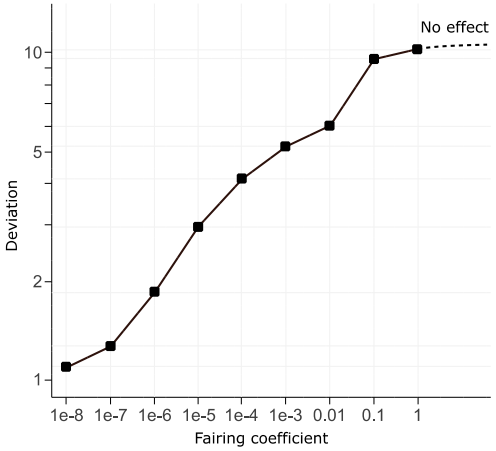


Рис. 16. Зависимость максимальной геометрической невязки от коэффициента сглаживания λ
Fig. 16. Dependence of the maximum geometric residual on the smoothing coefficient λ

Измерения производились в точках, снятых с результирующей поверхности $\mathbf{s}(u, v)$ с шагом 0.01 в параметрическом направлении u и 0.005 в параметрическом направлении v (рис. 17).

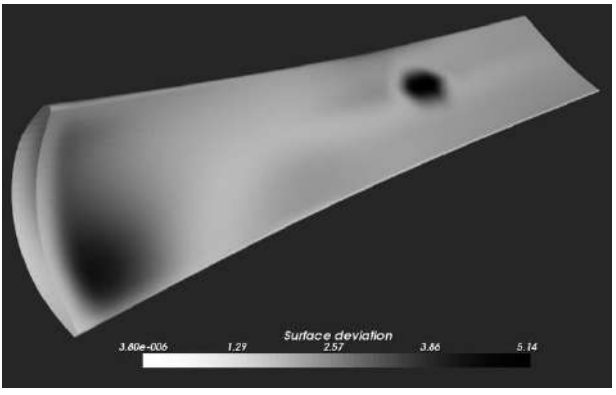


Рис. 17. Карта искажений сглаженной лопатки относительно исходной триангуляции для $\lambda = 0.001$
Fig. 17. The distortion map of the smoothed blade relative to the original triangulation for $\lambda = 0.001$

6. Заключение

Итогом развития представленной системы мы видим создание полнофункциональной открытой CAD-платформы для обратного инжиниринга. С ее помощью, на момент публикации данной статьи, был успешно реализован ряд алгоритмов, интегрированных в коммерческое программное обеспечение.

Для оценки геометрической точности в режиме реального времени интерес представляют подходы, привлекающие вычисления на GPU [52]. Еще одним пробелом в текущей системе является недостаточность набора операторов аппроксимации поверхностей. С практической точки зрения перспективной выглядит стратегия, изложенная в работе [53], а также дополняющий ее подход [54].

Исходные коды системы и расширенная документация доступны в глобальной сети по адресу www.analysissitus.org.

Список литературы

[1] Geng Z. and Bidanda B. Review of reverse engineering systems – current state of the art. Virtual and Physical Prototyping, vol. 12, no. 2, 2017, pp, 161–172.

[2] Requicha A.G. Representations for Rigid Solids: Theory, Methods, and Systems. ACM Computing Surveys vol. 12, no. 4, 1980, pp. 437–464.

[3] Sederberg T.W. and Parry S.R. Free-form deformation of solid geometric models. In Proc. of the 13th annual conference on Computer graphics and interactive techniques (SIGGRAPH'86), 1986, pp. 151–160.

[4] Bradley C. and Currie B. Advances in the Field of Reverse Engineering. Computer-Aided Design and Applications, vol. 2, no. 5, 2005m pp. 697–706.

[5] Gulanová J., Gulan L., Forrai M., and Hirz M. Generative engineering design methodology used for the development of surface-based components. Computer-Aided Design and Applications vol. 14, no. 5, 2017, pp. 642–649.

[6] Hafer L. and Kirkpatrick A.E. Assessing open source software as a scholarly contribution. Communications of the ACM, vol. 52, no. 12, 2009, pp. 126-129.

[7] Brown C.M. PADL-2: A Technical Summary. IEEE Computer Graphics and Applications, vol. 2, issue 2, 1982, pp. 69-84.

[8] Francesco Buonamici, Monica Carfagni, Rocco Furferi, Lapo Governi, Alessandro Lapini, Yary Volpe. . Reverse engineering modeling methods and tools : a survey. Computer-Aided Design and Applications, vol. 15, issue 3, 2018, pp. 443-464.

[9] Ibanez L, Schroeder W, Hanwell MD. Practicing Open Science. In Implementing Reproducible Computational Research, Chapman and Hall/CRC, 2014, pp. 241–280.

[10] Varady T. Automatic Procedures to Create CAD Models from Measured Data. Computer-Aided Design and Applications, vol. 5, no. 5, 2008, pp. 577–588.

[11] Mohaghegh K., Sadeghi M.H., and Abdullah A. Reverse engineering of turbine blades based on design intent. The International Journal of Advanced Manufacturing Technology, vol. 32, issue 9–10, 2007, pp. 1009–1020.

[12] Сляднев С.Е., Турлапов В.Е. Реконструкция параметрической модели лопатки турбины из набора аэродинамических сечений с использованием техники сглаживания. Труды 28-й Международной конференции по компьютерной графике и машинному зрению (ГрафиКон-2018**)**, 2018 г., стр. 495–499.

[13] Du T., Inala J.P., Pu Y. et al. InverseCSG: automatic conversion of 3D models to CSG trees. ACM Transactions on Graphics, vol. 27, no. 4, 2018, pp. 1–16.

[14] Сляднев С., Турлапов В. Метод декомпозиции машиностроительных твердотельных моделей на элементы объема изъятия. Труды 26-й Международной конференции по компьютерной графике, обработке изображений и машинному зрению, системам визуализации и виртуального окружения (ГрафиКон-2016**)**, 2016 г., стр. 58–63.

[15] Ye X., Liu H., Chen L., Chen Z., Pan X., and Zhang S. Reverse innovative design - an integrated product design methodology. CAD Computer Aided Design, vol. 40, no. 7, 2008, pp. 812–827.

[16] Harries S., Abt C., and Brenner M. Upfront CAD – Parametric Modeling Techniques for Shape Optimization. Computational Methods in Applied Sciences, vol. 48, 2019, pp. 191–211.

[17] Catmull E. and Clark J. Recursively generated B-spline surfaces on arbitrary topological meshes. Computer-Aided Design, vol. 10, no. 6, 1978, pp. 350–355.

[18] Antonelli M., Beccari C.V., Casciola G., Ciarloni R., and Morigi S. Subdivision surfaces integrated in a CAD system. Computer Aided Design, vol. 45, no. 11, 2013, pp. 1294–1305.

[19] Sederberg T.W., Zheng J., Bakenov A., and Nasri A. T-splines and T-NURCCs. ACM Transactions on Graphics, vol. 22 issue 3, 2003, pp. 477-484.

[20] Eck M. and Hoppe H. 1996. Automatic reconstruction of B-spline surfaces of arbitrary topological type. In Proc. of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH '96), 1996, pp. 325–334.

[21] Albat F., Müller R. Free-Form Surface Construction in a Commercial CAD/CAM System. Lecture Notes in Computer Science, vol. 3604. 2005, pp. 1-13.

[22] Andrews J., Jin H., and Séquin C. Interactive Inverse 3D Modeling. Computer-Aided Design and Applications, vol. 9, no. 6, 2012, pp. 881–900.

[23] Brooks F.P. The computer scientist as toolsmith II. Communications of the ACM, vol. 39, no. 3, 1996, pp. 61–68.

[24] Mobius J. and Kobbelt L. OpenFlipper: An open source geometry processing and rendering framework. Lecture Notes in Computer Science, vol. 6920, 2012, pp. 488–500.

[25] Cignoni P., Callieri M., Corsini M., Dellepiane M., Ganovelli F., and Ranzuglia G. MeshLab: an Open-Source Mesh Processing Tool. In Proc. of the Sixth Eurographics Italian Chapter Conference, 2008, pp. 129–136.

[26] Wang J., Gu D., Gao Z., Yu Z., Tan C., and Zhou L. Feature-Based Solid Model Reconstruction. Journal of Computing and Information Science in Engineering, vol. 13, no. 1, 2013.

[27] Hirz M., Rossbacher P., and Gulanová J. Future Trends in CAD – from the Perspective of Automotive Industry. In Proc. of the CAD'16, 2016, pp. 734–741.

[28] Schroeder W., Martin K., Lorensen B. Visualization Toolkit: An Object-Oriented Approach to 3D Graphics. Kitware, 2006, 528 p.

[29] Meyer M., Desbrun M., Schröder P., and Barr A.H. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In Visualization and Mathematics III, Mathematics and Visualization, Springer, 2003, pp. 35–57.

[30] Anwer N. and Mathieu L. From reverse engineering to shape engineering in mechanical design. CIRP Annals, vol. 65, issue 1, 2016, pp. 165–168.

[31] Venkataraman S., Sohoni, M., and Kulkarni, V. A graph-based framework for feature recognition. In Proc. of the sixth ACM symposium on Solid modeling and applications (SMA '01), 2001, 194–205.

[32] Várady, T., Martin R.R., and Cox J. Reverse engineering of geometric models—an introduction. Computer-Aided Design, vol. 29, no. 4, 1997, pp. 255–268.

[33] Lin F. and Hewitt W.T. Expressing Coons-Gordon surfaces as nurbs. Computer-Aided Design, vol. 26, no. 2, 1994, pp. 145–155.

[34] Corney J.R. and Lim T. 3D Modelling with ACIS, 2nd edition. Saxe-Coburg Publications, 2001. 388 p.

[35] Beccari C.V., Farella E., Liverani A., Morigi S., and Rucci M. A fast interactive reverse-engineering system. Computer-Aided Design, vol. 42, no. 10, 2010, pp. 860–873.

[36] Piegl L., Tille, W. The NURBS Book. Springer, 1996, 646 p.

[37] Kallay M. Constrained optimization in surface design, In Modeling in Computer Graphics, Springer, 1993, pp. 85-93.

[38] Bartels R.H., and Beatty J.C. A technique for the direct manipulation of spline curves. In Proc. of the Graphics Interface Conference'89, 1989, pp. 33–39.

[39] Moreira, A. and Santos, M.Y. Concave hull : a k-nearest neighbours approach for the computation of the region occupied by a set of points. In Proc. of the Second International Conference on Computer Graphics Theory and Applications, 2007, pp. 61–68.

[40] Slyadnev S. Active Data: SDK for organizing data of CAD/CAM/CAE software (powered by Open CASCADE Technology). Technical report. Режим доступа: https://www.researchgate.net/publication/282149692_Active_Data_SDK_for_organizing_data_of_CAD CAMCAE_software_powered_by_Open_CASCADE_Technology, дата обращения 28.01.2019.

[41] Jackson D.J. Boundary representation modelling with local tolerances. In. Proc. of the Third Symposium on Solid Modeling and Applications (SMA '95), 1995, pp. 247–254.

[42] Frischmann F. Topological and Geometric Healing on Solid Models. Master thesis, Faculty of Civil Engineering and Geodesy, Technische Universitat Munchen. 2011, 137 p.

[43] Slyadnev S., Malyshev A., and Turlapov V. CAD model inspection utility and prototyping framework based on OpenCascade. In Proc. of the 27th International Conference on Computer graphics, image processing and computer vision, visualization systems and virtual environment (GraphiCon 2017), pp. 323–327.

[44] Сляднев С. Open CASCADE Technology Overview. Режим доступа: http://isicad.net/articles.php?article_num=17368, дата обращения 28.01.2019.

[45] Colombo G., Facoetti G., Rizzi C., and Vitali A. Simplynurbs: A software library to model nurbs for medical applications. Computer-Aided Design and Applications, vol. 12, no. 6, 2015, pp. 794–802.

[46] Piegl L.A. and Tiller W. Surface skinning revisited. The Visual Computer, vol. 18, no. 4, 2002, pp. 273–283.

[47] Pérez-Arribas F. and Pérez-Fernández R. A B-spline design model for propeller blades. Advances in Engineering Software, vol. 118, 2017, pp. 35–44.

[48] Lee E.T.Y. Choosing nodes in parametric curve interpolation. Computer Aided Design vol. 21, no. 6, 1989, pp. 363–370.

[49] Ke Y., Fan S., Zhu W., Li A., Liu F., and Shi X. Feature-based reverse modeling strategies. Computer Aided Design, vol. 38, no. 5, 2006, pp.485–506.

[50] Juhász, I. and Róth, Á. Adjusting the energies of curves defined by control points. Computer-Aided Design, vol. 107, 2019, pp. 77-88.

[51] Pagani L. and Scott P.J. Curvature based sampling of curves and surfaces. Computer Aided Geometric Design, vol. 59, 2018, pp. 32–48.

[52] Kurella V., Stone B., and Spence A. GPU accelerated CAD to inspection data deviation colormap generation. Computer-Aided Design and Applications, vol. 14, no. 2, 2017, pp. 234–241.

[53] Weiss V., Andor L., Renner G., and Várady T. Advanced surface fitting techniques. Computer Aided Geometric Design, vol. 19, 2002, pp. 19–42.

[54] Vaitkus M. and Várady T. Parameterizing and extending trimmed regions for tensor-product surface fitting. Computer Aided Design, vol. 104, 2018, pp. 125-140.

Toward the development of open source software for the reconstruction of CAD-models

S.E. Slyadnev <sergey.slyadnev@gmail.com>
V.E. Turlapov <vadim.turlapov@itmm.unn.com>

Lobachevsky State University of Nizhni Novgorod,
23, Gagarin Avenue), Nizhnij Novgorod, 603950, Russia

Abstract. We describe an open source software package aimed at solving reverse engineering problems for CAD models defined in the polygonal form. We briefly discuss the main principles behind the new software, its architecture, and directions for further development. Examples of a turbine blade demonstrate the use of the software. In the first example, a turbine blade is reconstructed automatically from a structured point cloud. Another example is the interactive reconstruction of a turbine blade from an unstructured surface triangulation. In both cases, we use surface skinning strategy enhanced by a curve fairing operator. We illustrate by an example that the modified skinning operator does not minimize the total bending energy of the surface, but allows constructing a smooth patch where input inaccuracies are compensated. The reconstruction result is a parametric model of a turbine blade where the design variables are the coordinates of the poles of each profile curve. The presented software architecture can be used for partial or complete parameterization of the reconstructed CAD models aimed at their subsequent optimization.

Keywords: CAD; reverse engineering; CAD-model reconstruction; turbine blade skinning; fairing

DOI: 10.15514/ISPRAS-2019-31(1)-6

For citation: Slyadnev S.E., Turlapov V.E. Toward the development of open source software for the reconstruction of CAD-models. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 1, 2019. pp. 83-104 (in Russian). DOI: 10.15514/ISPRAS-2019-31(1)-6

References

[1] Geng Z. and Bidanda B. Review of reverse engineering systems – current state of the art. Virtual and Physical Prototyping, vol. 12, no. 2, 2017, pp, 161–172.

[2] Requicha A.G. Representations for Rigid Solids: Theory, Methods, and Systems. ACM Computing Surveys vol. 12, no. 4, 1980, pp. 437–464.

[3] Sederberg T.W. and Parry S.R. Free-form deformation of solid geometric models. In Proc. of the 13th annual conference on Computer graphics and interactive techniques (SIGGRAPH'86), 1986, pp. 151–160.

[4] Bradley C. and Currie B. Advances in the Field of Reverse Engineering. Computer-Aided Design and Applications, vol. 2, no. 5, 2005m pp. 697–706.

[5] Gulanová J., Gulan L., Forrai M., and Hirz M. Generative engineering design methodology used for the development of surface-based components. Computer-Aided Design and Applications vol. 14, no. 5, 2017, pp. 642–649.

[6] Hafer L. and Kirkpatrick A.E. Assessing open source software as a scholarly contribution. Communications of the ACM, vol. 52, no. 12, 2009, pp. 126-129.

[7] Brown C.M. PADL-2: A Technical Summary. IEEE Computer Graphics and Applications, vol. 2, issue 2, 1982, pp. 69-84.

[8] Francesco Buonamici, Monica Carfagni, Rocco Furferi, Lapo Governi, Alessandro Lapini, Yary Volpe. . Reverse engineering modeling methods and tools : a survey. Computer-Aided Design and Applications, vol. 15, issue 3, 2018, pp. 443-464.

[9] Ibanez L, Schroeder W, Hanwell MD. Practicing Open Science. In Implementing Reproducible Computational Research, Chapman and Hall/CRC, 2014, pp. 241–280.

[10] Varady T. Automatic Procedures to Create CAD Models from Measured Data. Computer-Aided Design and Applications, vol. 5, no. 5, 2008, pp. 577–588.

[11] Mohaghegh K., Sadeghi M.H., and Abdullah A. Reverse engineering of turbine blades based on design intent. The International Journal of Advanced Manufacturing Technology, vol. 32, issue 9–10, 2007, pp. 1009–1020.

[12] Slyadnev S., Turlapov V. In Proc. of the 27th International Conference on Computer graphics, image processing and computer vision, visualization systems and virtual environment (GraphiCon 2018), 2018, pp. 495–499 (in Russian).

[13] Du T., Inala J.P., Pu Y. et al. InverseCSG: automatic conversion of 3D models to CSG trees. ACM Transactions on Graphics, vol. 27, no. 4, 2018, pp. 1–16.

[14] Slyadnev S., Turlapov V. The method of decomposition of engineering solid-state models into elements of the volume of withdrawal. In Proc. of the 26th International Conference on Computer graphics, image processing and computer vision, visualization systems and virtual environment (GraphiCon 2016), 2016, pp. 58–63 (in Russian).

[15] Ye X., Liu H., Chen L., Chen Z., Pan X., and Zhang S. Reverse innovative design - an integrated product design methodology. CAD Computer Aided Design, vol. 40, no. 7, 2008, pp. 812–827.

[16] Harries S., Abt C., and Brenner M. Upfront CAD – Parametric Modeling Techniques for Shape Optimization. Computational Methods in Applied Sciences, vol. 48, 2019, pp. 191–211.

[17] Catmull E. and Clark J. Recursively generated B-spline surfaces on arbitrary topological meshes. Computer-Aided Design, vol. 10, no. 6, 1978, pp. 350–355.

[18] Antonelli M., Beccari C.V., Casciola G., Ciarloni R., and Morigi S. Subdivision surfaces integrated in a CAD system. Computer Aided Design, vol. 45, no. 11, 2013, pp. 1294–1305.

[19] Sederberg T.W., Zheng J., Bakenov A., and Nasri A. T-splines and T-NURCCs. ACM Transactions on Graphics, vol. 22 issue 3, 2003, pp. 477-484.

[20] Eck M. and Hoppe H. Automatic reconstruction of B-spline surfaces of arbitrary topological type. In Proc. of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH '96), 1996, pp. 325–334.

[21] Albat F., Müller R. Free-Form Surface Construction in a Commercial CAD/CAM System. Lecture Notes in Computer Science, vol. 3604. 2005, pp. 1-13.

[22] Andrews J., Jin H., and Séquin C. Interactive Inverse 3D Modeling. Computer-Aided Design and Applications, vol. 9, no. 6, 2012, pp. 881–900.

[23] Brooks F.P. The computer scientist as toolsmith II. Communications of the ACM, vol. 39, no. 3, 1996, pp. 61–68.

[24] Mobius J. and Kobbelt L. OpenFlipper: An open source geometry processing and rendering framework. Lecture Notes in Computer Science, vol. 6920, 2012, pp. 488–500.

[25] Cignoni P., Callieri M., Corsini M., Dellepiane M., Ganovelli F., and Ranzuglia G. MeshLab: an Open-Source Mesh Processing Tool. In Proc. of the Sixth Eurographics Italian Chapter Conference, 2008, pp. 129–136.

[26] Wang J., Gu D., Gao Z., Yu Z., Tan C., and Zhou L. Feature-Based Solid Model Reconstruction. Journal of Computing and Information Science in Engineering, vol. 13, no. 1, 2013.

[27] Hirz M., Rossbacher P., and Gulanová J. Future Trends in CAD – from the Perspective of Automotive Industry. In Proc. of the CAD'16, 2016, pp. 734–741.

[28] Schroeder W., Martin K., Lorensen B. Visualization Toolkit: An Object-Oriented Approach to 3D Graphics. Kitware, 2006, 528 p.

[29] Meyer M., Desbrun M., Schröder P., and Barr A.H. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In Visualization and Mathematics III, Mathematics and Visualization, Springer, 2003, pp. 35–57.

[30] Anwer N. and Mathieu L. From reverse engineering to shape engineering in mechanical design. CIRP Annals, vol. 65, issue 1, 2016, pp. 165–168.

[31] Venkataraman S., Sohoni, M., and Kulkarni, V. A graph-based framework for feature recognition. In Proc. of the sixth ACM symposium on Solid modeling and applications (SMA '01), 2001, 194–205.

[32] Várady, T., Martin R.R., and Cox J. Reverse engineering of geometric models—an introduction. Computer-Aided Design, vol. 29, no. 4, 1997, pp. 255–268.

[33] Lin F. and Hewitt W.T. Expressing Coons-Gordon surfaces as nurbs. Computer-Aided Design, vol. 26, no. 2, 1994, pp. 145–155.

[34] Corney J.R. and Lim T. 3D Modelling with ACIS, 2nd edition. Saxe-Coburg Publications, 2001. 388 p.

[35] Beccari C.V., Farella E., Liverani A., Morigi S., and Rucci M. A fast interactive reverse-engineering system. Computer-Aided Design, vol. 42, no. 10, 2010, pp. 860–873.

[36] Piegl L., Tille, W. The NURBS Book. Springer, 1996, 646 p.

[37] Kallay M. Constrained optimization in surface design, In Modeling in Computer Graphics, Springer, 1993, pp. 85-93.

[38] Bartels R.H., and Beatty J.C. A technique for the direct manipulation of spline curves. In Proc. of the Graphics Interface Conference'89, 1989, pp. 33–39.

[39] Moreira, A. and Santos, M.Y. Concave hull : a k-nearest neighbours approach for the computation of the region occupied by a set of points. In Proc. of the Second International Conference on Computer Graphics Theory and Applications, 2007, pp. 61–68.

[40] Slyadnev S. Active Data: SDK for organizing data of CAD/CAM/CAE software (powered by Open CASCADE Technology). Technical report. Available at: https://www.researchgate.net/publication/282149692_Active_Data_SDK_for_organizing_data_of_CAD CAMCAE_software_powered_by_Open_CASCADE_Technology, accessed 28.01.2019.

[41] Jackson D.J. Boundary representation modelling with local tolerances. In. Proc. of the Third Symposium on Solid Modeling and Applications (SMA '95), 1995, pp. 247–254.

[42] Frischmann F. Topological and Geometric Healing on Solid Models. Master thesis, Faculty of Civil Engineering and Geodesy, Technische Universitat Munchen. 2011, 137 p.

[43] Slyadnev S., Malyshev A., and Turlapov V. 2017. CAD model inspection utility and prototyping framework based on OpenCascade. In Proc. of the 27th International Conference on Computer graphics, image processing and computer vision, visualization systems and virtual environment (GraphiCon 2017), pp. 323–327.

[44] Slyadnev S. Open CASCADE Technology Overview. Available at: http://isicad.net/articles.php?article_num=17368, дата обращения 28.01.2019.

[45] Colombo G., Facoetti G., Rizzi C., and Vitali A. Simplynurbs: A software library to model nurbs for medical applications. Computer-Aided Design and Applications, vol. 12, no. 6, 2015, pp. 794–802.

[46] Piegl L.A. and Tiller W. Surface skinning revisited. The Visual Computer, vol. 18, no. 4, 2002, pp. 273–283.

[47] Pérez-Arribas F. and Pérez-Fernández R. A B-spline design model for propeller blades. Advances in Engineering Software, vol. 118, 2017, pp. 35–44.

[48] Lee E.T.Y. Choosing nodes in parametric curve interpolation. Computer Aided Design vol. 21, no. 6, 1989, pp. 363–370.

[49] Ke Y., Fan S., Zhu W., Li A., Liu F., and Shi X. Feature-based reverse modeling strategies. Computer Aided Design, vol. 38, no. 5, 2006, pp.485–506.

[50] Juhász, I. and Róth, Á. Adjusting the energies of curves defined by control points. Computer-Aided Design, vol. 107, 2019, pp. 77-88.

[51] Pagani L. and Scott P.J. Curvature based sampling of curves and surfaces. Computer Aided Geometric Design, vol. 59, 2018, pp. 32–48.

[52] Kurella V., Stone B., and Spence A. GPU accelerated CAD to inspection data deviation colormap generation. Computer-Aided Design and Applications, vol. 14, no. 2, 2017, pp. 234–241.

[53] Weiss V., Andor L., Renner G., and Várady T. Advanced surface fitting techniques. Computer Aided Geometric Design, vol. 19, 2002, pp. 19–42.

[54] Vaitkus M. and Várady T. Parameterizing and extending trimmed regions for tensor-product surface fitting. Computer Aided Design, vol. 104, 2018, pp. 125-140.

Математическая модель, описывающая динамику воздушных потоков в турбинном спирометре

А.В. Максимов <maksimov_alexey@inbox.ru>
Е.А. Киселев <evg-kisel2006@yandex.ru>
С.Д. Кургалин <kurgalin@bk.ru>
С.А. Зуев <zsa_zuev@mail.ru>

Воронежский государственный университет,
394018, Россия, г. Воронеж, Университетская пл., д. 1

Аннотация. Заболевания органов дыхания в настоящее время достаточно распространены, поэтому разработка новых эффективных способов их диагностики является актуальной. В данной работе мы описываем разработанную нами математическую модель взаимодействия потоков воздуха с подвижными частями прибора для недавно созданного турбинного спирометра нового типа. Он обладает рядом технических особенностей, которые должны быть учтены при моделировании. Среди прочих это достаточно существенная инерция турбины и слабое трение. Модель основана на уравнении моментов и содержит несколько эмпирических параметров. Поскольку трение в системе мало, то основные соотношения рассматриваются в линейном приближении. Экспериментальная проверка модели проведена в двух режимах работы спирометра. Во-первых, исследовано движение турбины по инерции после выключения внешнего источника воздуха. Во-вторых, проанализирована зависимость угловой скорости вращения турбины от скорости внешнего постоянного потока воздуха. Расчеты показали, что в указанных двух режимах разработанная математическая модель достаточно хорошо описывает результаты экспериментов. Также в настоящей работе указан простой способ определения эмпирических параметров на этапе калибровки прибора. Он основан на применении метода наименьших квадратов и не требует привлечения больших вычислительных мощностей. Это является важным обстоятельством, поскольку исследуемый спирометр предназначен для использования не только в специализированных медицинских учреждениях, но также и в бытовых условиях. На базе соотношений разработанной математической модели, предложен численный метод нахождения скорости входного потока воздуха. Это позволяет, опираясь на показания прибора, получать клинически значимую информацию о состоянии органов дыхания.

Ключевые слова: медицинская диагностика; спирометр; математическая модель; разработка медицинских приборов; турбинный спирометр; органы дыхания; портативный медицинский прибор; динамика воздушных потоков; датчик

DOI: 10.15514/ISPRAS-2019-31(1)-7

Для цитирования: Максимов А.В., Киселев Е.А., Кургалин С.Д., Зуев С.А. Математическая модель, описывающая динамику воздушных потоков в турбинном спирометре. Труды ИСП РАН, том 31, вып. 1, 2019 г., стр. 105-114. DOI: 10.15514/ISPRAS-2019-31(1)-7

1. Введение

Поскольку в настоящее время заболевания органов дыхания имеют достаточно высокую распространенность, разработка новых эффективных способов их медицинской диагностики является актуальной. Помимо выявления наличия определенных заболеваний, точная их диагностика помогает подобрать оптимальную дозу лечебных препаратов в зависимости от текущего состояния пациента. Эта проблема становится особенно острой,

когда пациент не находится в условиях стационара. Поэтому важным направлением медицинского приборостроения является разработка компактных спирометров, которые можно использовать не только в специализированных медицинских учреждениях, но и в бытовых условиях.

В последние десятилетия стремительное развитие получили микротехнологии, а также появилось множество способов удаленной передачи данных. Это обеспечило необходимую технологическую базу для создания портативных приборов, приспособленных для использования в домашних условиях и использующих удаленный обмен информацией с лечащим врачом. Основной технической трудностью при создании компактных спирометров является разработка малогабаритных датчиков, преобразующих движение воздушных потоков при дыхании в электрические сигналы. Одно из популярных решений данной проблемы – применение в приборах для измерения объемной скорости дыхания турбинных датчиков, имеющих сравнительно небольшой размер, но вполне сопоставимых по точности измерения скорости входного потока воздуха, например, с трубкой Флейша [1, 2].

При использовании турбинных датчиков, применяемых в настоящее время в спирометрии, возникает следующая проблема. С одной стороны, датчик должен быть компактным и обладать высокой чувствительностью при измерении параметров дыхания. По этой причине детали его конструкции должны иметь минимально возможные размеры и вес. С другой стороны, спирометр периодически необходимо подвергать санитарной обработке в целях соблюдения гигиены. Поэтому он должен легко разбираться, а его детали обладать достаточной прочностью [3]. В связи с этим существует необходимость создания спирометра нового типа с простой технологией санитарной обработки, но обеспечивающего при этом достаточную для диагностики точность измерения скорости потоков воздуха. Этот прибор и представлен в настоящей работе. Разработка таких приборов связана с изучением динамики потоков воздуха, формирующихся в их внутреннем пространстве. В связи с созданием нового вида спирометра, имеющего конструкцию, отличную от существующих, возникла необходимость построения и новой математической модели, которая учитывала бы его специфику и с достаточной степенью точности описывала динамику воздушных потоков внутри прибора.

Для математического описания движения потоков воздуха обычно используется модель сплошной среды. В этом подходе объектами изучения являются поле скоростей, распределение давления, плотности и температуры. Движение воздуха обычно описывается уравнением Навье – Стокса, которое для получения однозначного решения, дополняют уравнением неразрывности, уравнением состояния и уравнением теплопроводности [4].

Точное решение указанной выше задачи возможно лишь в нескольких частных случаях, а численное ее решение имеет большую вычислительную сложность [5]. Поэтому весьма актуальным является использование различных упрощенных моделей [6, 7]: уравнения Рейнольдса, модель Спаларта–Аллмараса, модели $k-\varepsilon$ и $k-\omega$ и др. Решение уравнений, составляющих основу этих моделей, также является достаточно сложным, поэтому часто в аэродинамике ограничиваются какими-либо эмпирическими формулами [8]. Математические модели такого рода, как правило, просты и поэтому для расчетов не требуется использовать значительные компьютерные ресурсы. Поскольку разработанный нами прибор предназначен для применения также и в домашних условиях, за основу модели будет взят именно такой подход.

Таким образом, целью настоящей работы является создание математической модели взаимодействия воздушных потоков с подвижными частями прибора для турбинного спирометра нового типа, не требующей для проведения расчетов больших вычислительных мощностей.

2. Материалы и методы исследования

2.1 Описание турбинного спирометра

Схема исследуемого турбинного спирометра и основные элементы его конструкции представлены на рис. 1. Цифрами 1 и 2 на схеме обозначены проницаемые перегородки, которые закручивают поток воздуха, цифрой 3 – подвижная турбина, удерживаемая двумя неодимовыми магнитами 4 и 5. Количество оборотов турбины регистрируется оптическим датчиком 6. $\vec{u}$ – скорость входного потока воздуха, $\vec{\omega}$ – угловая скорость вращения турбины, $\vec{L}$ – момент импульса турбины, I – ее момент инерции относительно оси z.

В отличие от аналогичных турбинных датчиков Gold Standart и flowMir, используемых в спирометрах компаний Micro Medical и Mir, данный прибор содержит турбину, которая массивнее в несколько раз. Благодаря этому она является более прочной. Следовательно, вероятность того, что она выйдет из строя при разборке или сборке прибора для санитарной обработки будет меньшей, чем у спирометров Micro Medical, Mir или подобных им.

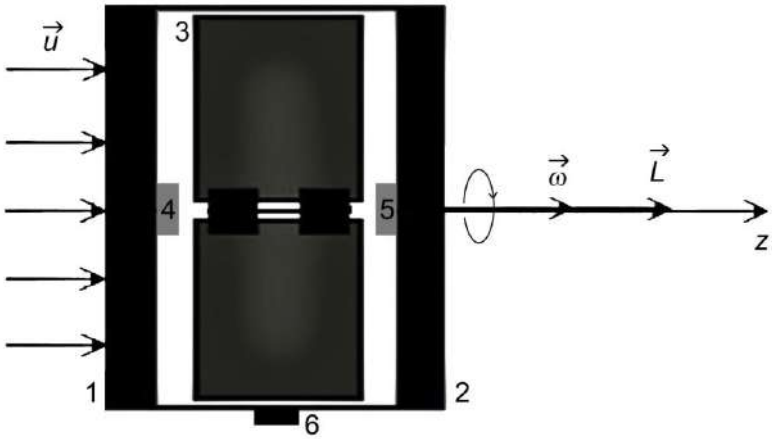


Рис. 1. Схема турбинного спирометра, элементы его конструкции (1–6) и основные обозначения
Fig. 1. Scheme of the turbine spirometer, elements of its structure (1–6) and basic notations

Другой особенностью конструкции является фиксация оси турбины в корпусе прибора с помощью неодимовых магнитов, а не традиционным механическим способом, как это обычно происходит в других устройствах такого типа. Это позволяет легко разбирать и собирать прибор, а также существенно снижает трение в системе, повышая тем самым его чувствительность к малым изменениям скорости входного потока воздуха. Кроме того, в отличие от известных аналогов в приборе используется только один оптический датчик, регистрирующий инфракрасный сигнал, отраженный от лопастей турбины, покрытых поочередно черным и белым пластиком PLA. Данный тип покрытия достаточно долговечен и при этом нетоксичен.

Датчик 6 (рис. 1) срабатывает в тот момент, когда мимо него проходит поверхность лопасти белого цвета. За один оборот турбины он регистрирует появление сигнала трижды, поскольку у турбины имеется шесть лопастей. Таким образом, если произойдет случайное несрабатывание датчика, то погрешность при подсчете количества оборотов за некоторый промежуток времени составит не один полный оборот, а лишь его часть. Некоторые из указанных выше инженерных решений уже были использованы нами ранее в иной конструкции турбинного спирометра [9].

2.2 Формулировка математической модели

В качестве основного соотношения математической модели мы используем уравнение моментов, которое в проекциях на ось z имеет вид

$$\frac{dL_z}{dt} = M_1(u_z) + M_2(\omega_z), \tag{1}$$

где $M_1(u_z)$ – момент сил, действующий на турбину, который обусловлен движением потоков воздуха в приборе (мы предполагаем, что он зависит только от проекции скорости входного потока воздуха u_z), $M_2(\omega_z)$ – момент сил, вызванный трением, $L_z = I\omega_z$. Будем считать, что ось z направлена в сторону движения потока воздуха при выдохе. Общий вид функций $M_1(u_z)$ и $M_2(\omega_z)$ мы устанавливаем эмпирически.

Сначала рассмотрим вращение турбины по инерции после выключения источника, подававшего в течение длительного времени воздух через проницаемую перегородку 1 (см. рис. 1) с постоянной скоростью. В этой ситуации довольно быстро остаточное продольное движение воздуха прекращается: $u_z = 0$. В то же время турбина в течение нескольких секунд продолжает вращаться по инерции.

Формула (1) при этих условиях дает

$$I\frac{d\omega_z}{dt} = M_2(\omega_z). \tag{2}$$

Рассмотрим простейший линейный вид функции $M_2(\omega_z)$

$$M_2(\omega_z) = -Q - P\omega_z, \tag{3}$$

где P и Q – некоторые положительные коэффициенты. Подставив (3) в (2), получим следующее дифференциальное уравнение относительно функции $\omega_z(t)$

$$\frac{d\omega_z}{dt} = -q - p\omega_z, \tag{4}$$

где $p = P/I$, $q = Q/I$. Решение уравнения (4) имеет вид

$$\omega_z(t) = \omega_z(0)e^{-pt} - \frac{q}{p}(1 - e^{-pt}). \tag{5}$$

Чтобы выяснить насколько соотношение (5) применимо на практике, мы провели 10 экспериментов по исследованию движения турбины по инерции после выключения внешнего потока воздуха и сравнили фактически полученные зависимости $\omega_z(t)$ с моделью (5). Параметры p и qрассчитывались при этом методом наименьших квадратов.

Для оценки степени соответствия разрабатываемой модели экспериментальным данным вычислялся коэффициент корреляции R между теоретическим и экспериментальным рядом значений

$$R = \frac{\overline{(\omega_T\omega_E)} - \overline{(\omega_T)}\,\overline{(\omega_E)}}{\sqrt{D_T}\sqrt{D_E}}, \tag{6}$$

где ω_E – экспериментально измеренное значение угловой скорости, ω_I определяется формулой (5), D_E и D_I – соответствующие дисперсии, черта сверху обозначает среднее значение.

Мы рассчитывали также отклонение δ экспериментальной зависимости от теоретической по среднеквадратичной норме

$$\delta = \sqrt{\frac{\sum_{i=1}^{n} (\omega_i^T - \omega_i^E)^2}{n}}, \quad (7)$$

где n – количество измерений.

Для исследования зависимости $M_1(u_z)$ было проведено 12 измерений величины угловой скорости вращения турбины ω_z при разных значениях скорости входного потока воздуха u_z в стационарном режиме работы спирометра: прибор подвергался длительному воздействию внешнего источника воздуха, обеспечивавшего постоянное значение величины u_z на входе. Скорость u_z измерялась с помощью анемометра, который имеет точность ~ 0,1 м/с. По характеру зависимости $\omega_z(u_z)$ можно судить и о поведении функции $M_1(u_z)$.

3. Результаты исследований и их обсуждение

3.1 Движение турбины по инерции

Выполненные расчеты показали, что представленная теоретическая модель (5) достаточно хорошо согласуется с экспериментальными данными. Коэффициент корреляции во всех случаях в среднем имеет величину $\sim 0{,}97$, отклонение по среднеквадратичной норме $\delta \sim 3\,\text{с}^{-1}$.

На рис. 2 и 3 показаны графики зависимости $\omega_z(t)$ при разных значениях $\omega_z(0)$.

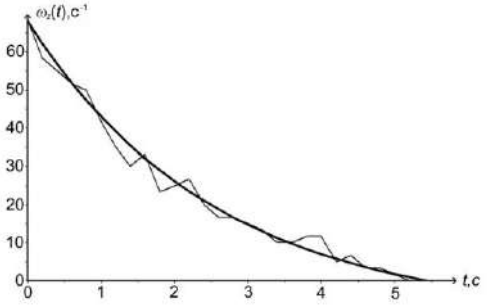


Рис. 2. Экспериментальная (тонкая линия) и теоретическая (жирная линия) зависимости угловой скорости от времени (начальная скорость 68 c^{-1})

Fig. 2. Experimental (thin line) and theoretical (bold line) dependences of the angular velocity on time (the initial velocity is 68 c-1)

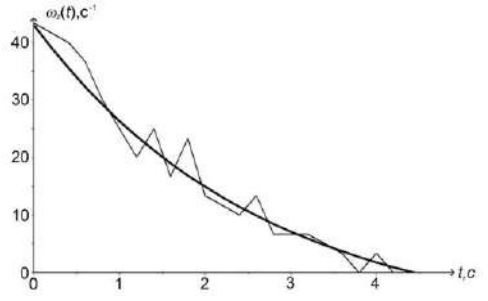


Рис. 3. Экспериментальная (тонкая линия) и теоретическая (жирная линия) зависимости угловой скорости от времени (начальная скорость 43 c^{-1})

Fig. 3. Experimental (thin line) and theoretical (bold line) dependences of the angular velocity on time (the initial velocity is 43 c^{-1})

Как следует из рис. 2, 3, отклонение экспериментальной кривой от теоретической носит знакопеременный характер. Кроме того, оно достаточно мало $\delta \sim 3\text{c}^{-1}$ по сравнению со средней скоростью ω_z, поэтому, с учетом высокого значения коэффициента корреляции $R \sim 0{,}97$, оно вполне может объясняться случайными факторами (небольшие колебания оси турбины, случайные несрабатывания оптического датчика и т. д.).

Во всех проведенных экспериментах значения параметров p и q оказались близкими. Например, в случае, соответствующем рис. 2, $p \approx 0{,}39\text{c}^{-1}$ и $q \approx 3{,}6\text{c}^{-2}$. В эксперименте, соответствующем рис. 3, получено, что $p \approx 0{,}38\text{c}^{-1}$ и $q \approx 3{,}7\text{c}^{-2}$. Это также свидетельствует в пользу того, что достаточно простая математическая модель (3) в данной ситуации вполне применима, поскольку, как мы видим из рис. 2 и 3, в среднем она отражает основные экспериментальные закономерности поведения функции $\omega_z(t)$.

3.2 Движение турбины в стационарном режиме

Рассмотрим стационарный режим работы спирометра, т.е. случай когда $u_z = const$.

На рис. 4 изображен график зависимости ω_z от u_z. Представленная на графике теоретическая зависимость является линейной функцией

$$\omega_z = k(u_z - u_{\min}), \tag{8}$$

которая была построена методом наименьших квадратов. Коэффициент корреляции (6) здесь достаточно велик и составляет величину $R \sim 0{,}98$. Отклонение по среднеквадратичной норме $\delta \sim 2\,\text{c}^{-1}$ на исследуемом участке мало по сравнению со средним значением ω_z.

Таким образом, проведенные эксперименты показывают, что зависимость ω_z от u_z в рассматриваемой ситуации близка к линейной, а имеющиеся отклонения экспериментальной кривой от теоретической (8) можно объяснить случайными факторами. Подобное обстоятельство другими авторами отмечалось ранее и для иных типов конструкций турбинного спирометра [3, 10–12].

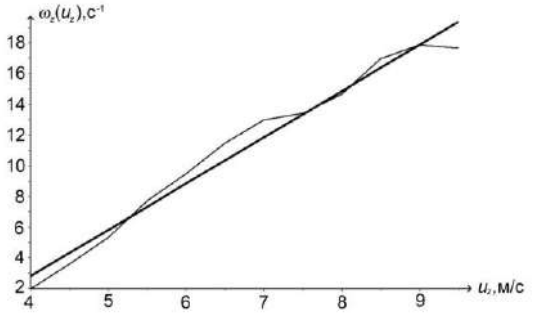


Рис. 4. Экспериментальная (тонкая линия) и теоретическая (жирная линия) зависимости угловой скорости от скорости внешнего потока воздуха при стационарном режиме работы спирометра
Fig. 4. Experimental (thin line) and theoretical (bold line) dependences of the angular velocity on velocity of the external air flow in stationary mode of the spirometer

В силу формулы (8) логично будет предположить, что функция $M_1(u_z)$ также линейно зависит от скорости u_z

$$M_1(u_z) = Au_z, \tag{9}$$

где A – некоторая положительная постоянная.

Покажем, что модель (9) при $u_z = const$ приводит к соотношению (8). С этой целью подставим формулы (9) и (3) в уравнение моментов (1)

$$\frac{dL_z}{dt} = I\frac{d\omega_z}{dt} = Au_z - Q - P\omega_z = 0. \quad (10)$$

Поделим (10) на момент инерции I и обозначим $a = A/I$, после чего получим:

$$au_z - q - p\omega_z = 0. \quad (11)$$

Отсюда непосредственно следует выражение (8), где коэффициенты k, $u_{\min}$ и a связаны следующими простыми соотношениями

$$k = \frac{a}{p}, u_{\min} = \frac{q}{a}. \quad (12)$$

3.3 Измерение объемной скорости дыхания и калибровка прибора

С учетом (3) и (9) соотношение (1) будет выглядеть следующим образом

$$\frac{d\omega_z}{dt} = au_z - p\omega_z - q. \quad (13)$$

Эмпирические постоянные a, p и q следует определять на этапе калибровки прибора.

Предположим, что известны все эмпирические константы, тогда мы предлагаем следующий численный метод для нахождения скорости входного потока воздуха u_z. Сначала с помощью оптического датчика количества оборотов методом конечных разностей определяем ω_z. Затем вычисляем производную $d\omega_z/dt$ также методом конечных разностей. Далее из формулы (13) выражаем интересующую нас скорость u_z через угловую скорость вращения турбины ω_z и ее первую производную

$$u_z = \frac{1}{a}\frac{d\omega_z}{dt} + \frac{p}{a}\omega_z + \frac{q}{a}. \quad (14)$$

С учетом (12) соотношение (14) можно записать и более кратко

$$u_z = \frac{1}{a}\frac{d\omega_z}{dt} + \frac{1}{k}\omega_z + u_{\min}. \quad (15)$$

Формула (15) является основой математической модели взаимодействия потоков воздуха в данном спирометре с его подвижными частями (турбиной). Она позволяет, используя описанный выше численный метод, находить скорость u_z в любой момент времени, а с ее помощью, в свою очередь, можно определять клинически значимые параметры дыхания, такие как объем форсированного выдоха (объем воздуха в литрах при совершении резкого выдоха), среднюю и мгновенную объемную скорость дыхания и др., которые имеют важное значение в медицинской диагностике [13, 14]. Для этого необходимо полученную скорость u_z умножить на площадь входного отверстия спирометра и результат выразить в литрах в секунду. Затем, с учетом температуры и влажности внутри прибора, требуется привести найденное значение объема выдыхаемого воздуха к альвеолярным условиям (температура 37^0 С, влажность 100 %, давление 760 мм рт. ст.). [13]

Итак, для практического использования формулы (15), остается определить три параметра: k, $u_{\min}$ и a, что можно сделать на этапе калибровки. Для этого, выполнив сначала серию экспериментов в стационарном режиме работы спирометра, необходимо найти величины k и $u_{\min}$. Для обеспечения надежности результатов, полученных методом наименьших

квадратов обычно бывает достаточно провести 10–15 измерений. Затем, проведя серию измерений в ситуации, когда турбина вращается по инерции, можно определить величины p и q также методом наименьших квадратов. После этого нужно найти параметр a с помощью (12), т. е. по формуле $a = kp = q/u_{\min}$. При этом калибровочные измерения следует проводить при различных значениях температуры и влажности с некоторой дискретизацией, сохранив полученные данные в памяти прибора. Это необходимо, поскольку от данных параметров зависит плотность воздуха и его вязкость. В специализированных таблицах температура обычно изменяется с шагом $1–2^0$ C [13]. Выбор оптимальной частоты дискретизации для данного прибора требует проведения дополнительного исследования.

Таким образом, на основе математической модели, которую дает формула (15), нами получен способ калибровки исследуемого спирометра, а также, опираясь на результаты измерений угловой скорости ω_z, сформулирован достаточно простой численный метод для нахождения скорости входного потока воздуха u_z.

4. Заключение

В данной работе предложена математическая модель для разработанного нами турбинного спирометра новой конструкции, описывающая взаимодействие воздушных потоков, характеризующихся скоростью $u_z(t)$, с турбиной прибора, движение которой описывается угловой скоростью $\omega_z(t)$. Главное практическое значение для определения параметров выдоха имеет формула (15), так как она позволяет, зная зависимость $\omega_z(t)$, вычислять скорость входного потока воздуха $u_z(t)$. По этим данным можно судить о клинически значимых параметрах дыхания.

Расчеты в рамках представленной новой математической модели не требуют привлечения больших вычислительных мощностей, что является важным, поскольку разработанный спирометр, предназначен для широкого применения в различных условиях, в том числе и домашних. Таким образом, поставленную цель можно считать достигнутой.

Достоверность результатов данной работы и возможность практического применения представленной здесь математической модели экспериментально подтверждается. Полученные соотношения для момента силы сопротивления (3) и момента, вызванного движением внешних потоков воздуха (9), описывают основные характерные особенности, экспериментальных кривых. Высокое значение коэффициента корреляции $R \sim 0{,}97$ позволяет сделать вывод, что имеющиеся отклонения экспериментальных кривых от теоретических обусловлены в основном случайными факторами, например, периодическим несрабатыванием оптического датчика, колебаниями оси турбины и т. п.

Основываясь на результатах проведенных экспериментов, предложен способ калибровки прибора. Для его реализации необходимо выполнить измерения угловой скорости вращения турбины ω_z в двух ситуациях: вращение турбины по инерции после выключения внешнего источника воздуха и в стационарном режиме работы спирометра, т. е. под действием потока воздуха постоянной скорости u_z. Это позволяет, применяя метод наименьших квадратов, достаточно легко найти все необходимые эмпирические параметры k, $u_{\min}$ и a, а если известны эти параметры, то формула (15) дает простой численный метод нахождения скорости входного потока воздуха u_z.

Определенные трудности при калибровке связаны с необходимостью вычисления указанных выше параметров при разных значениях температуры и влажности внутри и снаружи прибора. В дальнейшем предполагается усовершенствовать математическую

модель, путем включения в нее результатов исследования зависимости параметров k, $u_{\min}$ и a от влажности и температуры. Это позволит уменьшить объем данных, хранимых в приборе (величины k, $u_{\min}$ и a при разных значениях температуры и влажности), и существенно упростить процедуру его калибровки.

Список литературы

[1]. Godschalk I., Brackel H.J., Peters J.C., Bogaard J.M. Assessment of accuracy of applicability of a portable electronic diary card spirometer for asthma treatment. Respiratory Medicine, vol. 90, no. 10, 1996, pp. 619-622.

[2]. Yeh M.P., Adams T.D., Gardner R.M., Yanowitz F.G. Turbine flowmeter vs. Fleisch pneumotachometer: a comparative study for exercise testing. Journal of Applied Physiology, vol. 63, no. 3, 1987, pp. 1289-1295.

[3]. Сокол Е.И., Кипенский А.В., Томашевский Р.С., Король Е.И., Гура Ю.Н. Спирометрия. Ее техническое обеспечение. Проблемы и перспективы. Технічна електродинаміка: тем. вип. Проблеми сучасної електротехніки, ч. 3, 2008, стр. 119-124.

[4]. Лойцянский Л.Г. Механика жидкости и газа. Дрофа, Москва, 2003 г., 840 стр.

[5]. Ахметов В.К., Шкадов В.Я. Численное моделирование вязких вихревых течений для технических приложений. АСВ, Москва, 2009 г., 176 стр.

[6]. Гарбарук А.В., Стрелец М.Х., Шур М.Л. Моделирование турбулентности в расчетах сложных течений. Изд-во Политехнического университета, Санкт-Петербург, 2012 г., 88 стр.

[7]. Димитриенко Ю.И. Нелинейная механика сплошной среды. Физматлит, Москва, 2009 г., 624 стр.

[8]. Голубев А.Г., Калугин В.Т., Луценко А.Ю., Москаленко В.О., Столярова Е.Г., Хлупнов А.И., Чернуха П.А. Аэродинамика. Изд-во МГТУ им. Н.Э. Баумана, Москва, 2010 г., 687 стр.

[9]. Дыхательный тренажер, патент № 171354, Российская федерация, заявка № 2016145837 22.11.2016, опубликован 29.05.2017, бюл. № 16, 2 стр.

[10]. Chowienczyk P.J., Lawson C.P. Pocket-sized device for measuring forced expiratory volume in one second and forced vital capacity. British Medical Journal, vol. 285, 1982, pp. 15-17.

[11]. Goreke U., Habibi S., Azgin K., Beyaz M. A MEMS turbine prototype for respiration harvesting. Journal of Physics: Conference Series, vol. 660, no. 1, 2015.

[12]. Goreke U., Habibi S., Azgin K., Dogrusoz Y.S. The Development and Performance Characterization of Turbine Prototypes for a MEMS Spirometer. IEEE Sensors, vol. 16, no. 3, 2016, pp. 628-633.

[13]. Старшов А.М., Смирнов И.В. Спирография для профессионалов. Изд-во Познавательная книга Пресс, Москва, 2003 г., 77 стр.

[14]. Чучалин А.Г., Черняк А.В., Чикина С.Ю., Авдеев С.Н., Науменко Ж.К., Неклюдова Г.В., Айсанов З.Р., Калманова Е.Н. Функциональная диагностика в пульмонологии: Практическое руководство. Издательский холдинг Атмосфера, Москва, 2009 г., 192 стр.

Mathematical model describing air flow dynamics in a turbine spirometer

A.V. Maksimov <maksimov_alexey@inbox.ru>
E.A. Kiselev <evg-kisel2006@yandex.ru>
S.D. Kurgalin <kurgalin@bk.ru>
S.A. Zuev <zsa_zuev@mail.ru>

Voronezh State University,
1, Universitetskaya sq., Voronezh, 394018, Russia

Abstract. Diseases of the respiratory system are currently quite common, so the development of new effective ways to diagnose them is relevant. In this paper, we describe a mathematical model developed by us for the interaction of air flows with moving parts of a device for a recently created turbine spirometer of a new type. It has a number of technical features that must be taken into account when modeling. Among others, this is quite substantial inertia of the turbine and low friction. The model is based on the moment equation and contains several empirical parameters. Since the friction in the system is small, the main

relations are considered in the linear approximation. Experimental verification of the model was carried out in two modes of operation of the spirometer. Firstly, the inertial motion of the turbine after turning off the external air source was investigated. Secondly, the dependence of the angular velocity of rotation of the turbine on the speed of an external constant air flow was analyzed. The calculations showed that in this two modes, the developed mathematical model describes the experimental results quite well. Also in this paper a simple method is given for determining empirical parameters at the device calibration stage. It is based on the use of the least squares method and does not require the involvement of large computational powers. This is an important circumstance, since the spirometer under investigation is intended for use not only in specialized medical institutions, but also in home conditions. On the base of the relations of the developed mathematical model, a numerical method is proposed for finding the velocity of the incoming air flow. This allows, basing on the readings of the device, to obtain clinically relevant information about the state of the respiratory system.

Keywords: medical diagnostics; spirometer; mathematical model; development of medical devices; turbine spirometer; respiratory organs; portable medical device; dynamics of air flows; sensor.

DOI: 10.15514/ISPRAS-2019-31(1)-7

For citation: Maksimov A.V., Kiselev E.A., Kurgalin S.D., Zuev S.A. Mathematical model describing air flow dynamics in a turbine spirometer. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 1, 2019. pp. 105-114 (in Russian). DOI: 10.15514/ISPRAS-2019-31(1)-7

References

[1]. Godschalk I., Brackel H.J., Peters J.C., Bogaard J.M. Assessment of accuracy of applicability of a portable electronic diary card spirometer for asthma treatment. Respiratory Medicine, vol. 90, no. 10, 1996, pp. 619-622.

[2]. Yeh M.P., Adams T.D., Gardner R.M., Yanowitz F.G. Turbine flowmeter vs. Fleisch pneumotachometer: a comparative study for exercise testing. Journal of Applied Physiology, vol. 63, no. 3, 1987, pp. 1289-1295.

[3]. Sokol E.I., Kipenskij A.V., Tomashevskij R.S., Korol' E.I., Gura Yu.N. Spirometry. Its technical support. Problems and perspectives. Tekhnichna elektrodinamika: tem. vip. Problemi suchasnoi elektrotekhniki [Technical electrodynamics: thematic issue Problems of modern electrical engineering], vol. 3, pp. 119-124, 2008 (in Russian)

[4]. Lojcyanskij L.G. Mechanics of fluid and gas. Izd. Drofa [Drofa publ.], Moscow, 2003, 840 p. (in Russian)

[5]. Ahmetov V.K., Shkadov V.Ya. Numerical modeling of viscous vortex flows for technical applications. Izd. ASV [ASV publ.], Moscow, 2009, 176 p. (in Russian)

[6]. Garbaruk A.V., Strelec M.H., Shur M.L. Modeling of turbulence in the calculations of complex flows. Izd. Politekhnicheskogo universiteta [Publ. of Polytechnical University], Saint-Petersburg, 2012, 88 p. (in Russian)

[7]. Dimitrienko Yu.I. Nonlinear continuum mechanics. Izd. Fizmatlit [Fizmatlit publ.], Moscow, 2009, 624 p. (in Russian)

[8]. Golubev A.G., Kalugin V.T., Lucenko A.Yu., Moskalenko V.O., Stolyarova E.G., Hlupnov A.I., Chernuha P.A. Aerodynamics. Izd. MGTU im. N.E. Baumana [Publ. of Bauman Moscow State Technical University], Moscow, 2010, 687 p. (in Russian)

[9]. Respiratory training apparatus, patent № 171354, Russian Federation, request № 2016145837 22.11.2016, publication 29.05.2017, bulletin № 16, 2 p. (in Russian)

[10]. Chowienczyk P.J., Lawson C.P. Pocket-sized device for measuring forced expiratory volume in one second and forced vital capacity. British Medical Journal, vol. 285, 1982, pp. 15-17.

[11]. Goreke U., Habibi S., Azgin K., Beyaz M. A MEMS turbine prototype for respiration harvesting. Journal of Physics: Conference Series, vol. 660, no. 1, 2015.

[12]. Goreke U., Habibi S., Azgin K., Dogrusoz Y.S. The Development and Performance Characterization of Turbine Prototypes for a MEMS Spirometer. IEEE Sensors, vol. 16, no. 3, 2016, pp. 628-633.

[13]. Starshov A.M., Smirnov I.V. Spirography for professionals. Izd Poznavatel'naya kniga Press [Poznavatel'naya kniga Press publ.], Moscow, 2003, 77 p. (in Russian)

[14]. Chuchalin A.G., Chernyak A.V., Chikina S.Yu., Avdeev S.N., Naumenko Zh.K., Neklyudova G.V., Ajsanov Z.R., Kalmanova E.N. Functional Diagnostics in Pulmonology: A Practical Guide. Izdatel'skij holding Atmosfera [Publishing holding Atmosfera], Moscow, 2009, 192 p. (in Russian)

Обнаружение неисправностей в комбинационных схемах на основе самодвойственного дополнения до равновесных кодов

[1, 2] *Д.В. Ефанов <TrES-4b@yandex.ru>*
[3] *В.В. Сапожников <port.at.pgups@gmail.com>*
[3] *Вл.В. Сапожников <at.pgups@gmail.com>*
[3] *Д. В. Пивоваров <pivovarov.d.v.spb@gmail.com>*
[1] *ООО «ЛокоТех-Сигнал»,*
107113, Россия, г. Москва, 3-я Рыбинская ул., д. 18, стр. 22,
[2] *Российский университет транспорта (МИИТ),*
127994, РФ, Москва, ул. Образцова, д. 9
[3] *Петербургский государственный университет путей сообщения Императора Александра I,*
190031, Россия, г. Санкт-Петербург, Московский пр., д. 9

Аннотация. Рассматривается новый метод контроля комбинационных логических схем на основе дополнения рабочих функций объекта диагностирования до кодовых слов равновесных кодов с параллельным преобразованием функций их разрядов к виду самодвойственных функций и последующим контролем по данному признаку. Такой подход к организации систем функционального контроля позволяет повысить обнаруживающую способность по сравнению с традиционным контролем по равновесным кодам. При этом не обнаруживаемой может быть только такая ошибка, которая приведет к сохранению веса кодового слова, формируемого на выходах блока логического дополнения, при совпадении искажений всех функций разрядов на противоположных входных наборах. Данное условие более жесткое, чем контроль принадлежности вектора заранее выбранному коду, что дает большую вероятность обнаружения возникающего при неисправностях искажения. Представленный метод позволяет осуществлять контроль комбинационных схем на основе равновесных кодов с одинаковым числом единичных и нулевых разрядов. Приоритет в таком случае оказывается у широко используемого кода «2 из 4», структура тестера которого является наиболее простой из всех тестеров равновесных кодов. Приведено подробное описание новой структуры системы контроля, даны алгоритмы доопределения значений контрольных функций, обеспечивающие контролепригодность технических средств диагностирования.

Ключевые слова: комбинационная схема; обнаружение неисправностей; самодвойственное дополнение; равновесные коды; код «2 из 4».

DOI: 10.15514/ISPRAS-2019-31(1)-8

Для цитирования: Ефанов Д.В., Сапожников В.В., Сапожников Вл.В., Пивоваров Д.В. Обнаружение неисправностей в комбинационных схемах на основе самодвойственного дополнения до равновесных кодов. Труды ИСП РАН, том 31, вып. 1, 2019 г., стр. 115-132. DOI: 10.15514/ISPRAS-2019-31(1)-8

1. Введение

При работе современных микроэлектронных систем автоматики не исключены сбои, которые не должны приводить к нарушениям ответственных технологических процессов. По этой причине на этапе разработки и проектирования обосновываются мероприятия по повышению надежности и контролепригодности отдельных компонентов, блоков и узлов. В данные мероприятия входят периодическое тестирование, резервирование, функциональное диагностирование, контроль выполнения функций и т.д.

Важное значение в задаче парирования проявлений неисправностей имеют методы функционального диагностирования – такого вида технического диагностирования, при котором процедура определения технического состояния объекта диагностирования не требует отключения его от работы с целью подачи проверяющих воздействий – рабочие воздействия одновременно являются и тестовыми при таком подходе [1]. Своевременное определение неверно функционирующего объекта в составе устройства или целой системы позволяет блокировать результаты его вычислений и не использовать их при принятии управляющих решений. Сам же неисправный объект подвергается перезагрузке, а при повторном выявлении дефекта – отключается от работы и исследуется более тщательно при тестовом диагностировании.

При функциональном диагностировании часто осуществляется косвенный контроль возникающих неисправностей по результатам вычислений значений рабочих функций [2 – 6]. Это позволяет контролировать устройство в целом, а не отдельные его компоненты, и соответственно, уменьшить глубину диагностирования до отдельного блока с последующей заменой его при выявленном отказе. При организации систем функционального контроля объект диагностирования снабжается техническими средствами диагностирования, позволяющими осуществить проверку правильности вычисления значений его рабочих функций. Часто эти средства синтезируются на основании выбора в качестве базового какого-либо помехозащитного блочного равномерного кода [7, 8]. Соответствие формируемых кодовых векторов, включающих в себя разряды рабочих функций объекта диагностирования и разряды специальных контрольных функций, проверяется тестером [9]. Существует и другой подход при синтезе технических средств диагностирования, связанный с проверкой соответствия значений формируемых рабочих функций некоторому заранее выбранному классу функций алгебры логики [10]. Например, контроль монотонности или самодвойственности функций.

Данная статья посвящена изложению метода синтеза технических средств диагностирования для логических комбинационных схем на основе модифицированной структурной схемы логического дополнения до равновесных кодов с параллельным контролем принадлежности функций к классу самодвойственных функций.

2. Самодвойственные комбинационные схемы

Одной из первых работ, обращающих внимание на возможность использования при организации технического диагностирования логических устройств свойств специальных классов функций алгебры логики, является работа [11]. В ней показаны диагностические способности самодвойственных функций.

Определение 1. Функция алгебры логики является самодвойственной, если выполняется следующее условие:

$$f(x_1, x_2, ..., x_t) = \overline{f(\overline{x_1}, \overline{x_2}, ..., \overline{x_t})}. \qquad (1)$$

Другими словами, самодвойственная функция является функцией, которая двойственна сама к себе. У такой функции значения являются противоположными на противоположных друг другу значениях аргументов (на противоположных входных векторах) [12].

В табл. 1 приводится пример самодвойственной функции.

Табл. 1. Самодвойственная функция

Tab. 1. Self-dual function

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0

0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Существуют дискретные устройства, выходы которых описываются самодвойственными функциями [13]. Например, таковым является полный сумматор (рис. 1), функции выходов которого описываются системой [14]:

$$\begin{cases} S = x_1 \oplus x_2 \oplus x_3; \\ C = x_1x_2 \vee x_1x_3 \vee x_2x_3. \end{cases} \quad (2)$$

Неисправности внутренней структуры сумматора приводят к искажению вычисляемых им функций и к нарушению их самодвойственности (рис. 1). Например, при возникновении неисправности, приведенной на рис. 1, изменяются функции, реализуемые на выходах сумматора: $S = (x_1 \oplus x_2) \vee x_3$ и $C = x_1x_2$. Нетрудно проверить, что данные функции самодвойственными не являются. Следовательно, можно контролировать техническое состояние сумматора путем контроля самодвойственности его функций.

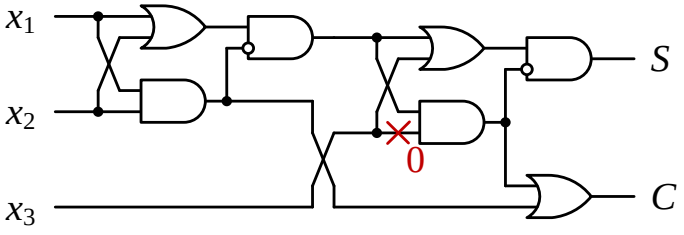


Рис. 1. Схема полного сумматора
Fig. 1. Full-adder circuit

Определение 2. Логическую комбинационную схему будем называть самодвойственной, если все ее выходные функции являются самодвойственными.

К самодвойственным относится небольшой класс комбинационных схем. Однако существуют способы преобразования их в схемы со структурами, обладающими свойством самодвойственности выходных функций [15, 16]. При этом никак не нарушаются основные функциональные характеристики самой схемы – она наделяется контролепригодной архитектурой, в которой внутренние логические элементы определенным образом связываются путями между собой, входами и выходами схемы.

Универсальным является следующий способ преобразования комбинационных схем в самодвойственные.

В основе метода лежит разложение Шеннона, применимое для любой функции алгебры логики [17]:

$$f(x_1,...,x_i,...,x_t) = \overline{x_i}\, f(x_1,...,0,...,x_t) \vee x_i\, f(x_1,...,1,...,x_t). \quad (3)$$

Для преобразования схемы в самодвойственную для каждой реализуемой ею функции вводится дополнительный сигнал, называемый альтернативным сигналом *a*. Этот сигнал представляет собой последовательность импульсов, формируемую внешним генератором сигналов [16].

Тогда, учитывая (3), можно преобразовать любую функцию в самодвойственную с учетом следующего выражения:

$$f^* = f^*(x_1, x_2, ..., x_t) = \bar{a} f(0, x_1, x_2, ..., x_t) \vee a f(1, x_1, x_2, ..., x_t) = \\ = \bar{a} f_1(x_1, x_2, ..., x_t) \vee a f_2(x_1, x_2, ..., x_t), \quad (4)$$

где $\boldsymbol{f_1(x_1, x_2, \ldots, x_t)}$ и $\boldsymbol{f_2(x_1, x_2, \ldots, x_t)}$ – двойственные функции.

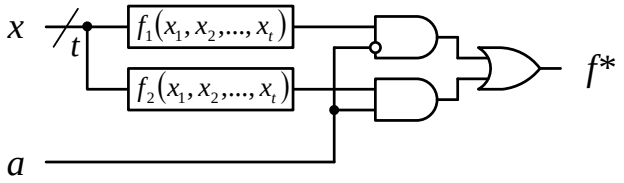


Рис. 2. Универсальная структура реализации самодвойственной схемы
Fig. 2. Universal structure of self-dual circuit realization

Приведем пример преобразования простейшей схемы в самодвойственную с использованием выражения (4). Имеется комбинационная схема, реализующая функцию логического сложения $f_1 = x_1 \vee x_2$. Для преобразования исходной комбинационной схемы в самодвойственную по формуле (4) требуется найти функцию, двойственную к функции f_1: $f_2 = \overline{\overline{x_1} \vee \overline{x_2}} = x_1 x_2$. Так как функции f_1 и f_2 являются двойственными, то при импульсном поступлении переменных x_1, x_2 и a будут последовательно отрабатывать сначала один элемент логического умножения, а затем второй (верхний и нижний элементы логического умножения на рис. 2). Соответственно, функция f^* будет принимать в одном такте сначала значение функции f_1, а затем, на противоположном входном наборе, значение функции f_2. Описание преобразованной схемы для наглядности дано в таблице 2.

Особенности метода самодвойственного дополнения и его использования при организации контроля как комбинационных, так и многотактных (последовательностных) схем изложены в монографиях [12, 13].

Исследования показывают, что перспективным оказывается сочетание особенностей методов контроля логических схем как по признаку самодвойственности функций, так и по принадлежности их в совокупности к заранее выбранному избыточному коду. Эффективным в этом плане является использование равновесных кодов с малой длиной кодовых слов. Одним из таких кодов, имеющим существенные преимущества перед всеми другими равновесными кодами, является 2/4-код [18]. Отметим основные особенности данного равновесного кода, позволяющие применять его при организации систем функционального контроля:

1. Тестер 2/4-кода (2/4-*TSC*) имеет простую структуру и требует для полной проверки всего четырех комбинаций: {0011; 1100; 1001; 0110}.
2. Для преобразования любого четырехбитного вектора, формируемого на выходах блока $F(x)$, в кодовое слово 2/4-кода потребуется изменение максимум двух функций, а значит, блок логического дополнения будет иметь наименьшую сложность, а блок контрольной логики – всего два выхода.
3. Эксперименты [19] показывают, что использование 2/4-кода для контроля многовыходных логических схем дает меньшую по сложности схему контроля, чем для любых других равновесных кодов.

Табл. 2. Описание самодвойственной схемы из рассматриваемого примера
Tab. 2. Description of the self-dual circuit of the example

a	x_1	x_2	f^*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1

1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Опишем структурные схемы систем функционального контроля комбинационных схем, основанных на контроле самодвойственности логических функций.

3. Контроль самодвойственных функций

Как отмечалось выше, учитывая возможности построения самодвойственных комбинационных схем, можно осуществлять и контроль возникающих в них неисправностей. С этой целью следует проверять соответствие реализуемых схемой функций классу самодвойственных функций алгебры логики. На рис. 3 изображена структурная схема системы функционального контроля, основанная именно на таком свойстве.

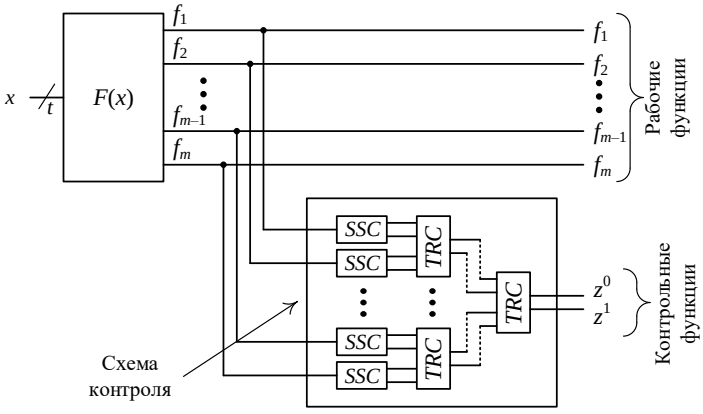


Рис. 3. Система функционального контроля для самодвойственной схемы
Fig. 3. Concurrent checking system for self-dual circuit

Для контроля принадлежности формируемых схемой функций классу самодвойственных функций алгебры логики используется специализированная схема контроля, включающая в себя тестеры самодвойственных сигналов *SSC* (*self-checking self-dual checker*) [12, 20] и модули сжатия парафазных сигналов *TRC* (*two-rail checker*) [21].

Структурная схема *SSC* приведена на рис. 4. В ней самодвойственный сигнал *f** при помощи линии задержки, равной одному такту импульсной последовательности *a*, преобразуется в двухфазный сигнал $<v_1\ v_2>$. *SSC* оборудован двумя выходами и при самодвойственности входного сигнала формирует парафазный сигнал <01> либо <10> на выходах. При нарушении самодвойственности поступающего сигнала на выходе формируется непарафазный сигнал.

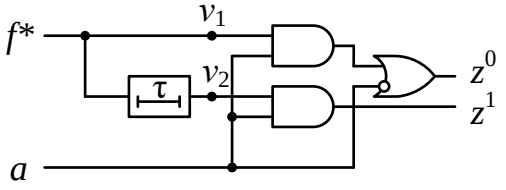


Рис. 4. Структурная схема SSC
Fig. 4. SSC structural circuit

Модуль сжатия парафазных сигналов позволяет преобразовывать два парафазных сигнала в один и используется для уменьшения числа наблюдаемых контрольных выходов. С этой

целью парафазные выходы всех *SSC* объединяются на входах самопроверяемых схем сжатия парафазных сигналов *TRC*, выходы которых каскадно соединяются таким образом, чтобы получить на выходе схемы контроля один парафазный сигнал. Модули *TRC* являются стандартными, один из вариантов реализации приведен на рис. 5 [21].

Для контроля несамодвойственных схем автоматики и вычислительной техники используется система функционального контроля на основе логического дополнения, приведенная на рис. 6 [22].

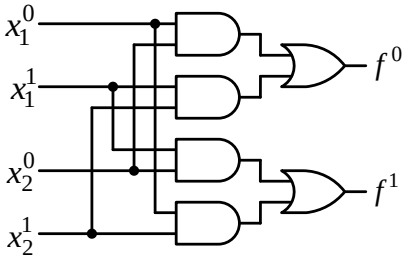


Рис. 5. Структурная схема TRC
Fig. 5. TRC structural circuit

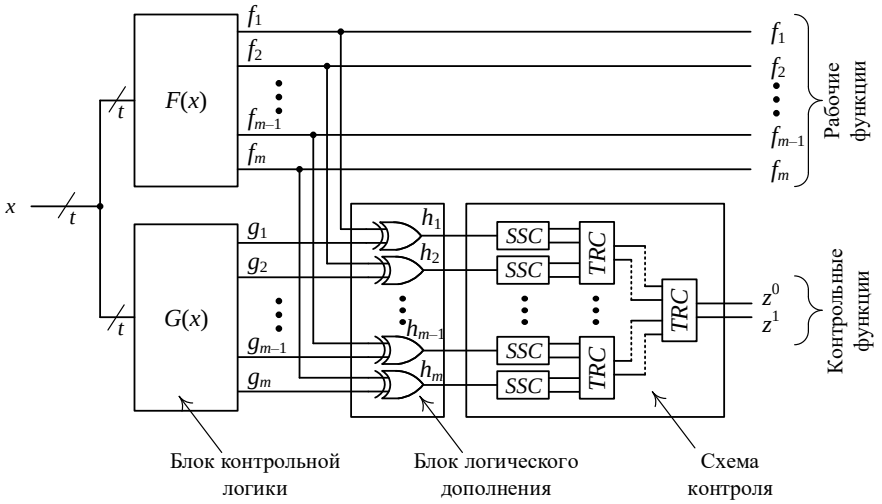


Рис. 6. Система функционального контроля для несамодвойственной схемы
Fig. 6. Concurrent checking system for non-self-dual circuit

В системе функционального контроля, приведенной на рис. 6, с целью контроля правильности вычислений каждая рабочая функция, вычисляемая объектом диагностирования, преобразуется в самодвойственную. Это преобразование осуществляется с использованием специального блока логического дополнения, образованного параллельно расположенными сумматорами по модулю два (элементами *XOR*). Преобразование каждой функции осуществляется по формуле:

$$h_i = f_i \oplus g_i, \quad i = \overline{1,m}. \qquad (5)$$

4. Метод самодвойственного дополнения до равновесных кодов

Развивая метод логического дополнения, применим для контроля правильности вычислений блоком $F(x)$ своих функций не только контроль по признаку самодвойственности, но и контроль по равновесному коду, часто используемому при организации систем

функционального контроля. Это позволит повысить обнаруживающую способность при функциональном контроле.

В системе функционального контроля, синтезированной по методу логического дополнения по равновесному коду, не будут обнаружены любые неисправности, вызывающие симметричную ошибку в векторе $<h_1\ h_2\ \ldots\ h_m>$. Повысить эффективность обнаружения ошибок можно за счет контроля еще какого-либо свойства функции, например, принадлежности ее к классу самодвойственных функций. Такой контроль двух видов (как принадлежности каждой функции к классу самодвойственных, так и контроль принадлежности вектора $<h_1\ h_2\ \ldots\ h_m>$ к равновесному коду) возможен только при использовании кодов вида «r из $2r$» ($r/2r$-кодов), где r – вес кодового вектора. К таким кодам относятся коды 1/2, 2/4, 3/6 и т.д.

Целесообразно рассмотреть использование 2/4-кода в системе функционального контроля, так как его тестер имеет наиболее простую структуру, а также требует для полной проверки подачи на входы только четырех кодовых комбинаций. На рис. 7 изображена модифицированная структурная схема системы функционального контроля как по равновесным кодам, так и по принадлежности каждой функции к классу самодвойственных функций. Объектом диагностирования является блок $F(x)$, реализующий четыре рабочие функции. Блок $F(x)$ снабжается специализированным аппаратным обеспечением, позволяющим в процессе его эксплуатации косвенно контролировать процесс возникновения неисправностей за счет фиксации неверных результатов вычислений.

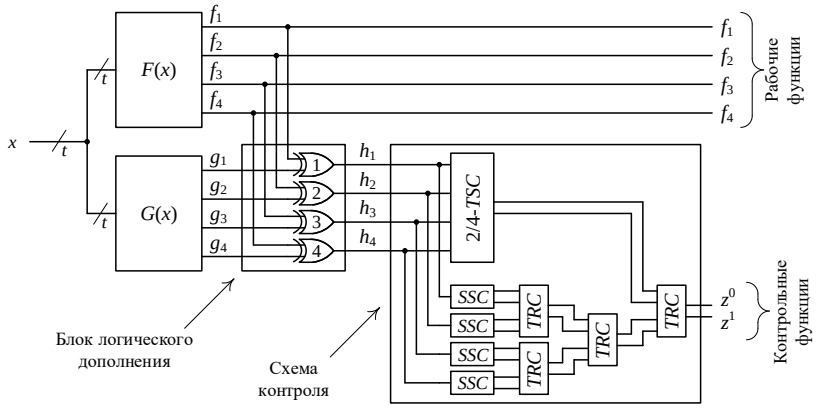


Рис. 7. Структурная схема системы функционального контроля по методу логического дополнения до 2/4-кода

Fig. 7. Concurrent checking system structural circuit, synthesis on self-dual complement to «2-out-of-4 code» method

Структурная схема, приведенная на рис. 7, является базовым вариантом организации системы функционального контроля. Для многовыходных логических схем следует выделять группы выходов для отдельного контроля с последующим объединением контрольных выходов на входах самопроверяемого компаратора.

Информационный вектор $<f_1\ f_2\ f_3\ f_4>$, образованный рабочими функциями объекта диагностирования, преобразуется в кодовое слово $<h_1\ h_2\ h_3\ h_4>$, принадлежащее 2/4-коду. Причем, преобразование осуществляется таким образом, чтобы каждая функция в кодовом слове являлась самодвойственной. Эта особенность кодового слова $<h_1\ h_2\ h_3\ h_4>$ достигается путем использования блока контрольной логики $G(x)$ и блока логического дополнения, образованного каскадом двухвходовых сумматоров по модулю два (элементов *XOR*). На этапе проектирования системы функционального контроля функции, реализуемые блоком $G(x)$, подбираются таким образом, чтобы были обеспечены требуемые свойства для реализации контроля вычислений и подача всех тестовых комбинаций на все элементы *XOR*.

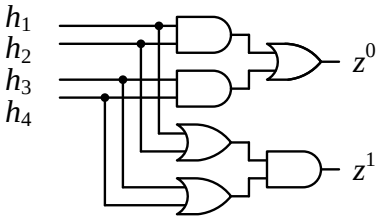


Рис. 8. Структурная схема 2/4-TSC
Fig. 8. 2/4-TSC structural circuit

Контроль принадлежности кодового слова <h_1 h_2 h_3 h_4> 2/4-коду осуществляется с использованием тестера 2/4-*TSC* (*totally self-checking checker*), наиболее простая структура которого изображена на рис. 8 [23]. 2/4-*TSC* снабжен двумя выходами, на которых формируется парафазный сигнал <01> или <10>, в случае, если поступающее на его входы кодовое слово принадлежит 2/4-коду. В противном случае на выходах тестера формируется непарафазный сигнал, свидетельствующий об ошибке в вычислениях и косвенно указывающий на наличие неисправностей в одном из блоков системы функционального контроля.

Контроль самодвойственности каждой функции кодового слова <h_1 h_2 h_3 h_4> осуществляется с использованием тестера самодвойственной функции *SSC*.

Эффект обнаружения ошибок в представленной на рис. 7 структуре достигается за счет того, что при наличии искажения в кодовом слове <h_1 h_2 h_3 h_4>, не зафиксированного схемой 2/4-*TSC*, возникает искажение функций h_1, h_2, h_3 и(или) h_4, что может быть выражено и в нарушении их самодвойственности.

Утверждение. *В системе функционального контроля на основе логического дополнения до равновесного r/2r-кода с самодвойственными функциями разрядов обнаруживается любая ошибка в кодовом слове <h_1 h_2 … h_m>, кроме ошибки, которая не нарушает его вес и при этом проявляется в виде одновременных искажений одинаковых разрядов на противоположных входных наборах.*

Справедливость формулировки утверждения вытекает из следующих соображений. В системе функционального контроля осуществляется контроль логической схемы на основе двух признаков. Первый признак – принадлежность кодового вектора <h_1 h_2 … h_m> к равновесному коду. В том случае, если при наличии неисправности в блоках $F(x)$, $G(x)$ или логического дополнения будет сформировано кодовое слово заранее выбранного равновесного $r/2r$-кода, неисправность по первому признаку зафиксирована не будет. Второй признак – это самодвойственность каждой контрольной функции. Неисправность по данному признаку не будет обнаружена в том случае, если вызовет искажения на противоположных входных наборах. При этом, если неисправность не обнаружена по первому признаку, она может быть обнаружена по второму признаку, и наоборот. Не обнаруженной окажется только такая неисправность, которая одновременно не нарушит веса кодового слова <h_1 h_2 … h_m> и самодвойственности каждой функции.

Рассмотрим пример, который позволит проиллюстрировать указанные особенности модифицированной схемы системы функционального контроля, изображенной на рис. 7. В табл. 3 задана комбинационная логическая схема, имеющая четыре входа и четыре выхода.

Получим значения контрольных функций g_1, g_2, g_3, g_4, позволяющие организовать контроль заданной комбинационной логической схемы на основе логического дополнения до равновесного кода с контролем самодвойственности каждой функции контрольного разряда кодового слова <h_1 h_2 h_3 h_4>. Расширим таблицу и доопределим вручную значения контрольных функций. При этом учтем необходимость выполнения следующих требований:

1. Каждое кодовое слово <h1 h2 h3 h4> должно иметь вес r=2.
2. Должны быть сформированы как минимум по одному разу тестовые комбинации для 2/4-

TSC: {0011; 1100; 0110; 1001} [23].

3. Должны быть сформированы как минимум по одному разу тестовые комбинации для каждого элемента XOR: {00; 01; 10; 11} [24].
4. Значения каждой функции h1, h2, h3 и h4 должны быть противоположными на противоположных входных наборах.

С учетом представленных требований получаются значения контрольных функций, формируемые блоком $G(x)$.

Табл. 3. Таблица истинности комбинационной логической схемы
Tab. 3. Truth table of combinational logic circuit

№	x_1	x_2	x_3	x_4	f_1	f_2	f_3	f_4
0	0	0	0	0	0	1	0	1
1	0	0	0	1	1	1	1	1
2	0	0	1	0	1	1	1	1
3	0	0	1	1	1	0	0	0
4	0	1	0	0	1	0	1	1
5	0	1	0	1	0	0	0	0
6	0	1	1	0	1	1	0	1
7	0	1	1	1	0	1	0	1
8	1	0	0	0	1	0	1	1
9	1	0	0	1	0	0	1	0
10	1	0	1	0	0	1	1	1
11	1	0	1	1	0	1	0	1
12	1	1	0	0	1	0	0	1
13	1	1	0	1	0	0	1	0
14	1	1	1	0	0	1	1	1
15	1	1	1	1	1	0	0	0

Существует большое число способов доопределения значений контрольных функций, позволяющих обеспечить выполнение требований контролепригодности структуры системы функционального контроля. Как показывают исследования, наиболее простой является следующая последовательность доопределения значений контрольных функций.

Алгоритм 1. *Доопределение значений контрольных функций с учетом выполнения требований контролепригодности схемы контроля с постпроверкой тестируемости блока логического дополнения:*

5. Проверяется наличие как минимум двух нулевых и двух единичных значений каждой функции f_1, f_2, f_3 и f_4 на всех входных наборах.
6. В столбцы первой (или второй) половины таблицы относительно середины, соответствующие кодовым словам $<h_1\ h_2\ h_3\ h_4>$, произвольным образом, но равномерно, заносятся комбинации <0011> и <0110> или <0011> и <1001>, или <1100> и <0110>, или <1100> и <1001>.
7. Путем заполнения противоположными значениями на противоположных наборах относительно середины таблицы заполняются столбцы второй половины таблицы, соответствующие кодовым словам $<h_1\ h_2\ h_3\ h_4>$.
8. Вычисляются значения контрольных функций $g_i = f_i \oplus h_i, i = \overline{1,4}$.
9. Определяются формируемые комбинации на элементах сложения по модулю два в блоке логического дополнения.
10. Исходя из п. 2 и п. 3, требования о формировании тестового множества для 2/4-*TSC* и

наделения функций h_1, h_2, h_3 и h_4 свойством самодвойственности будут выполнены. Требуется проверить формирование полного множества тестовых комбинаций для элементов сложения по модулю два блока логического дополнения.

Если для какого-либо элемента сложения по модулю два блока логического дополнения не удается сформировать требуемую тестовую комбинацию, требуется изменить заполнение строк на этапе выполнения п. 2. Для этого осуществляется поиск двух входных противоположных наборов, которые позволят доопределить значения функций логического дополнения до требуемых комбинаций для полной проверки элементов *XOR*. Затем вновь потребуется проделать шаги 3 – 6 алгоритма.

Алгоритм 1 удобно использовать при большом числе входных переменных, так как вероятность достижения результата по формированию тестовых комбинаций элементов сложения по модулю два блока логического дополнения с увеличением числа входных переменных возрастает.

Пользуясь описанным алгоритмом, получим табл. 4. Данная таблица содержит описание рабочих и контрольных функций системы функционального контроля. Блок контрольной логики $G(x)$ синтезируется по значениям полученных контрольных функций. Кроме того, в таблице 4 представлены все формируемые тестовые комбинации для элементов сложения по модулю два в блоке логического дополнения.

Алгоритм 1 в большинстве случаев дает возможность построения контролепригодной структуры схемы контроля.

Альтернативным вариантом для построения системы функционального контроля является использование другого алгоритма, основанного на первоначальной проверке контролепригодности элементов сложения по модулю два.

Табл. 4. Сигналы на линиях схемы системы функционального контроля
Tab. 4. Signals on the lines of the functional control system circuit

№	Значения рабочих функций				Значения контрольных функций				Кодовые слова на выходе блока логического дополнения				Формируемые тестовые комбинации элементов сложения по модулю два			
	f_1	f_2	f_3	f_4	g_1	g_2	g_3	g_4	h_1	h_2	h_3	h_4	XOR_1	XOR_2	XOR_3	XOR_4
0	0	1	0	1	0	0	1	1	0	1	1	0	00	10	01	11
1	1	1	1	1	1	0	0	1	0	1	1	0	11	10	10	11
2	1	1	1	1	1	0	0	1	0	1	1	0	11	10	10	11
3	1	0	0	0	1	1	1	0	0	1	1	0	11	01	01	00
4	1	0	1	1	1	0	0	0	0	0	1	1	11	00	10	10
5	0	0	0	0	0	0	1	1	0	0	1	1	00	00	01	01
6	1	1	0	1	1	1	1	0	0	0	1	1	11	11	01	10
7	0	1	0	1	0	1	1	0	0	0	1	1	00	11	01	10
8	1	0	1	1	0	1	1	1	1	1	0	0	10	01	11	11
9	0	0	1	0	1	1	1	0	1	1	0	0	11	01	11	00
10	0	1	1	1	1	0	1	1	1	1	0	0	01	10	11	11
11	0	1	0	1	1	0	0	1	1	1	0	0	11	10	00	1
12	1	0	0	1	0	0	0	0	1	0	0	1	00	00	00	10
13	0	0	1	0	1	0	1	1	1	0	0	1	11	00	11	01
14	0	1	1	1	1	1	1	0	1	0	0	1	01	11	11	10
15	1	0	0	0	0	0	0	1	1	0	0	1	10	00	00	01

Алгоритм 2. *Последовательное доопределение значений контрольных функций с учетом выполнения требований контролепригодности:*

1. Проверяется наличие как минимум двух нулевых и двух единичных значений каждой функции f1, f2, f3 и f4 на всех входных наборах.
2. Осуществляется доопределение значений функций h1, h2, h3 и h4 с учетом необходимости формирования тестовых комбинаций для элементов сложения по модулю два в блоке логического дополнения:
   - хотя бы для одного случая f_i=0 значение функции h_i доопределяется до 0 и хотя бы для одного случая f_i=0 – до 1;
   - хотя бы для одного случая f_i=1 значение функции h_i доопределяется до 0 и хотя бы для одного случая f_i=1 – до 1.
3. Если доопределение значений функций hi на одних и тех же входных наборах привело к формированию не кодового слова 2/4-кода, осуществляется иное доопределение; если ни один вариант доопределения не удовлетворяет поставленному условию, то самопроверяемая схема на основе представленного подхода построена быть не может.
4. Заполняются значения функций h1, h2, h3 и h4 в строках, расположенных симметрично относительно середины таблицы: уже заполненные значения инвертируются и помещаются в соответствующие строки, расположенные симметрично.
5. Заполняются столбцы кодовых слов <h1 h2 h3 h4> для тех строк, в которых уже занесены значения.
6. Проверяется наличие всех четырех тестовых комбинаций 2/4-TSC.
7. С учетом требований по формированию кодовых слов 2/4-кода, необходимых для проверки 2/4-TSC и самодвойственности функций h1, h2, h3 и h4 заполняются оставшиеся столбцы.

Представленный алгоритм дает возможность синтеза самопроверяемой структуры комбинационной схемы.

Полученные по алгоритмам 1 и 2 структурные схемы будут наделены искомыми свойствами принадлежности кодового вектора $<h_1\ h_2\ h_3\ h_4>$ 2/4-коду, а также самодвойственности каждой функции, что вытекает из последовательности шагов алгоритма. На каждом шаге обеспечивается выполнение нескольких свойств для каждой функции h_i. Возможности изменения значений каждой рабочей функции объекта диагностирования позволяют выполнить необходимые преобразования для большинства логических комбинационных схем за исключением таких схем, для которых на всех входных наборах не формируется как минимум по два нулевых и два единичных значения каждой функции f_1, f_2, f_3 и f_4 (это не даст сформировать все четыре тестовые комбинации для элементов *XOR* в блоке логического дополнения).

5. Моделирование работы структуры логического дополнения

Для исследования возможностей в обнаружении ошибок в предложенной в данной работе структурной схеме системы функционального контроля были проведены эксперименты с моделированием одиночных константных неисправностей в заданной в табл. 3 комбинационной схеме, реализованной в базисе типовых логических элементов в среде моделирования *Multisim.*

Процесс реализации структуры оказался не тривиальным, так как в стандартных средствах *Multisim* трудно реализуемы элементы задержки сигналов, применяющиеся в тестере самодвойственности функций. Было решено сымитировать его работу за счет использования двух JK-триггеров, работающих как D-триггеры (U15A и U9A). Аналог тестера самодвойственности, реализованный в среде *Multisim*, представлен на рис. 9.

К входам синхронизации триггеров подключен генератор тактовых импульсов частотой 10 Гц с небольшим сдвигом по фазе. К одному из триггеров генератор подключен через инвертор

(U16D). Таким образом, триггеры срабатывают в разные моменты времени. Отметим, что к тестеру подключен еще один генератор тактовых импульсов с частотой 10 Гц (U13).

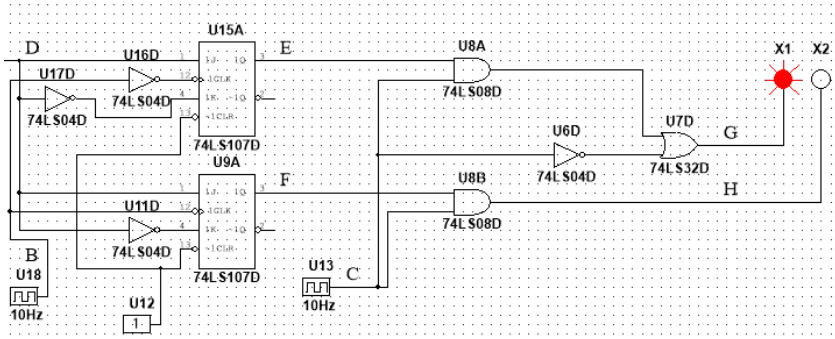


Рис. 9. Моделирование работы тестера самодвойственности
Fig. 9. Self-dual checker simulation

Как видно из схемы на рис. 9, если на выходе генератора U13 будет сигнал <1>, то на индикаторы X1 и X2 транслируется сигнал с выходов триггеров. Если же на выходе генератора U13 будет сигнал <0>, то на индикаторах установится сигнал <10>. Таким образом, тестер вырабатывает парафазный сигнал тогда, когда на выходе генератора появляется сигнал <0> или когда на выходе генератора присутствует сигнал <1>, а триггеры находятся в противоположных состояниях. Для дальнейшего описания приведем простую схему, контролируемую данным тестером (рис. 10).

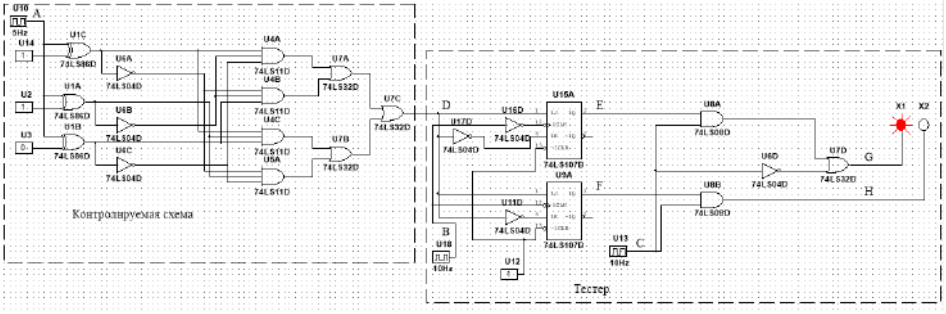


Рис. 10. Пример применения тестера самодвойственности
Fig. 10. Example of self-dual checker used

Тестер самодвойственности сравнивает текущий сигнал, приходящий на его вход с предыдущим сигналом. Поэтому на входах контролируемой схемы установлены элементы «сложение по модулю два» (U1C, U1A, U1B). Один вход этих элементов подключен к соответствующему источнику входного сигнала, а другой к генератору тактовых импульсов (U10) с частотой импульсов в два раза меньшей, чем у генератора тестера. Как упоминалось выше, генераторы тестера имеют небольшой сдвиг по фазе. Без этого сдвига возникает момент, когда все генераторы переключаются одновременно, в результате чего возникает состязание. Это состязание может приводить к ложному сигналу ошибки на выходе тестера. Таким образом, в момент времени, когда на выходе генератора U10 сформирован сигнал логического нуля, на входы схемы подаются заданные входные наборы, а когда на выходе генератора установлен сигнал логической единицы, на входы схемы подаются значения, противоположные заданным. В результате этого, если схема является самодвойственной, то на ее выходе генерируется самодвойственный сигнал. На рис. 11 приводится временная диаграмма работы схемы, приведенной на рис. 10.

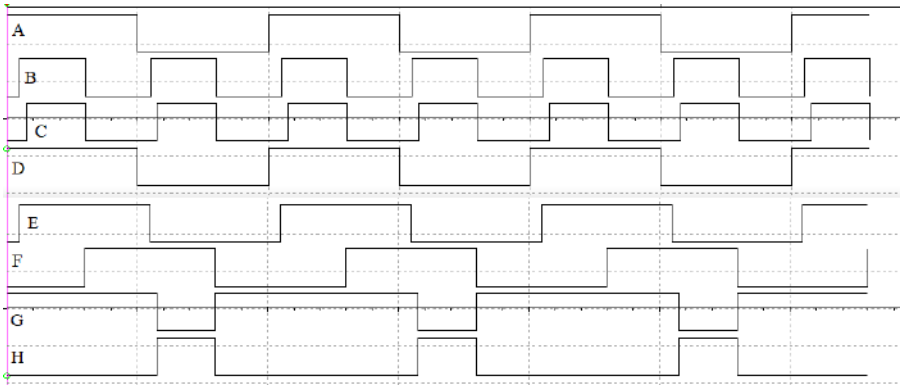


Рис. 11. Временная диаграмма работы схемы
Fig 11. Timing diagram of the circuit

На рис. 11 буквами A – H подписаны диаграммы, соответствующие проводам на рис. 10. В начальный момент времени на выходе генератора A появляется сигнал логической единицы. То есть на входы схемы подается входной набор противоположный установленному. На выходе контролируемой схемы появляется сигнал логической единицы. При этом, на выходах тестера появляется парафазный сигнал, так как на выходах генераторов B и C присутствуют сигналы <0>. Далее генераторы B и C меняют свои выходные состояния с нуля на единицу. В этот момент выходной сигнал контролируемой схемы записывается в триггер F. На выходах тестера формируется парафазный сигнал, так как на выходах триггеров присутствуют разные сигналы. Далее генераторы B и C меняют свои выходные состояния с <1> на <0>. В этот момент выходное состояние схемы записывается в триггер F. Оба триггера находятся в одинаковых состояниях, но на выходе тестера все равно будет парафазный сигнал, так как генератор C вырабатывает сигнал <0>. Далее генератор A меняет свое состояние с единицы на ноль, то есть на входы схемы поступают установленные сигналы. В этот момент на выходе схемы (поскольку она является самодвойственной) появляется сигнал, противоположный предыдущему. Спустя небольшой промежуток времени этот сигнал записывается в триггер E, и оба триггера будут находиться в противоположных состояниях, что приводит к возникновению парафазного сигнала на выходе тестера. Далее, при смене состояния генератора B, информация с выхода контролируемой схемы записывается в триггер F. В результате этого триггеры переходят в одинаковые состояния, но так как на выходе генератора C фиксируется сигнал <0>, тестер вырабатывает парафазный сигнал. Далее генератор A переключается в состояние <1>, и цикл повторяется.

Таким образом, если контролируемая схема самодвойственная, то она будет менять выходное состояние на противоположное тогда, когда на выходе генератора A поменяется сигнал на противоположный. В свою очередь, если на выходе схемы будут попеременно появляться противоположные сигналы, то триггеры будут находиться в противоположных состояниях тогда, когда на выходе генератора C будет сигнал <1>. В этом случае противоположные сигналы с триггеров будут транслироваться на выходы тестера, и таким образом будет получен парафазный сигнал. Также парафазный сигнал будет появляться в тот момент, когда на выходе генератора C появляется сигнал нуля.

Если схема не самодвойственная, то на выходе будет один и тот же сигнал до тех пор, пока установленные входные сигналы не поменяются. В этом случае триггеры установятся в одинаковые состояния, и тестер будет вырабатывать непарафазный сигнал тогда, когда на выходе генератора C будет сигнал единицы.

Здесь стоит обратить внимание на то, что если генератор U13 сделать аналогичным генератору U18, то в момент его переключения в состояние единицы на выходы схемы будут транслироваться сигналы с триггеров, а в триггер U15A будет записано показание с входа

тестера. Однако триггер меняет свое состояние с некоторой задержкой, вследствие этого возникает короткий момент времени, когда на выходах тестера возникает ложный сигнал ошибки. Поэтому у генератора U13 сдвиг больше, чем у U18. При этом в состояние нуля также быстрее переходит генератор U18. В результате этого возникает момент времени, когда в триггер U9A записывается сигнал с выхода тестера и оба триггера устанавливаются в одинаковые состояния. В это же время на выходе генератора U13 сигнал единицы, и сигналы с выходов триггеров транслируются на выходы тестера. Таким образом, возникает ложный сигнал ошибки. Для того, чтобы этого избежать необходимо, чтобы Генератор U13 переходил в состояние нуля раньше или одновременно с генератором U18. Чтобы этого добиться был уменьшен коэффициент заполнения генератора U13.

Представленную на рис. 9 схему тестера самодвойственности не рекомендуется применять в реальных схемах, так как она содержит элементы памяти. Однако для процесса моделирования это не имеет значения, так как целью является только получение данных по характеристикам обнаружения ошибок.

Тестирование моделируемой схемы происходило аналогично для каждой функции *h*. К каждому тестеру подключался генератор с частотой 10 Гц и небольшим сдвигом по фазе. Далее выходы тестеров объединялись при помощи схем *TRC*. Получившаяся схема тестера объединялась с тестером равновесного кода «2 из 4» также при помощи модулей *TRC*. На каждом входе всей схемы были установлены элементы «сложение по модулю 2», один вход которых подсоединялся к генератору входных сигналов, а другой – к генератору тактовых импульсов с частотой 5 Гц. Таким образом, если хотя бы одна функция *h* оказывалась несамодвойственной, то на выходе всего тестера вырабатывался непарафазный сигнал с частотой 10 Гц.

Для симуляции схема, заданная табл. 3, была реализована в среде *Multisim*. В результате минимизации функций, заданных таблицей истинности, получились следующие формулы:

$$f_1 = \overline{x_1}\,\overline{x_2}x_4 \vee x_1\overline{x_3}\,\overline{x_4} \vee x_2\overline{x_3}\,\overline{x_4} \vee \overline{x_1}x_3\overline{x_4} \vee x_1x_2x_3x_4;$$
$$f_2 = x_3\overline{x_4} \vee x_1\overline{x_2}x_3 \vee \overline{x_1}x_2x_3 \vee \overline{x_1}\,\overline{x_2}\,\overline{x_3};$$
$$f_3 = \overline{x_2}\,\overline{x_3}x_4 \vee x_1\overline{x_2}\,\overline{x_3} \vee x_1x_3\overline{x_4} \vee \overline{x_2}x_3\overline{x_4} \vee x_1\overline{x_3}x_4 \vee \overline{x_1}x_2\,\overline{x_3}\,\overline{x_4};$$
$$f_4 = \overline{x_1}\,\overline{x_4} \vee \overline{x_1}x_2x_3 \vee x_1\overline{x_2}x_3 \vee \overline{x_1}\,\overline{x_2}\,\overline{x_3} \vee x_1\overline{x_4}.$$

Представленные формулы описывают реализуемые схемой функции в виде двухуровневой дизъюнктивной нормальной формы записи. Для имитации кратных ошибок на выходах схемы формулы были преобразованы к виду:

$$f_1 = yx_4 \vee x_1\overline{x_3}\,\overline{x_4} \vee x_2\overline{x_3}\,\overline{x_4} \vee zx_3 \vee ak;$$
$$f_2 = p \vee x_1\overline{x_2}x_3 \vee \overline{x_1}k \vee y\overline{x_3};$$
$$f_3 = bx_4 \vee x_1b \vee x_1p \vee \overline{x_2}p \vee a\overline{x_3} \vee zx_2\overline{x_3};$$
$$f_4 = z \vee \overline{x_1}k \vee x_1\overline{x_2}x_3 \vee y\overline{x_3} \vee x_1\overline{x_4};$$
$$y = \overline{x_1}\,\overline{x_2};$$
$$z = \overline{x_1}\,\overline{x_4};$$
$$k = x_2x_3;$$
$$a = x_1x_4;$$
$$b = \overline{x_2}\,\overline{x_3}.$$

По конечным формулам была построена комбинационная схема, а также реализована система функционального контроля по разработанному в данной статье методу в *Multisim*. На выходы элементов, использующихся в нескольких функциях (y, z, k, a), поочередно вводились константные неисправности. Это производилось отсоединением соответствующего элемента от всей схемы и подсоединением на его место источника логических сигналов. Далее имитировалась поочередная подача всех возможных входных наборов. Значения на выходах контролируемой схемы сравнивались со значениями в таблице 3. Если значения не совпадали, то фиксировалась ошибка в информационном векторе. Если при этом тестер вырабатывал парафазный сигнал, то ошибка классифицировалась как необнаруженная.

Всего при 8 неисправностях в схеме возникло 45 ошибок. Из них 16 –многократные ошибки. Остальные ошибки – однократные. Все эти ошибки были обнаружены в системе функционального контроля.

Стоит отметить, что при таком способе контроля тестер показывал сигнал ошибки и тогда, когда ошибка не транслировалась на выходы, но неисправность присутствовала. Это происходило в том случае, когда неисправность не проявляла себя на данном входном наборе, но проявлялась на противоположном. В результате этого нарушалась самодвойственность какой-либо из функций, и на выходе тестера появлялся сигнал ошибки. Другими словами, предлагаемый метод повышает контролепригодность синтезируемой системы функционального контроля.

6. Заключение

Использование при функциональном контроле логических комбинационных схем нескольких признаков функций алгебры логики, формируемых на выходах объекта диагностирования, позволяет повысить обнаруживающую способность метода. Одним из эффективных вариантов может оказаться совместное использование самодвойственного дополнения и контроля по равновесным кодам «r из 2r» (r/2r-кодов), где r – вес кодового вектора. К таким кодам относится код «2 из 4». Для полной проверки самого простого из тестеров данных кодов требуется всего четыре тестовые комбинации. Эти обстоятельства определяют преимущества применения кодов «2 из 4» перед всеми остальными r/2r-кодами при совместном самодвойственном преобразовании рабочих функций объекта диагностирования.

Обнаруживающая способность при использовании совместного контроля по равновесным кодам с самодвойственным дополнением функций повышается за счет контроля сразу же нескольких признаков, однако вопрос того, насколько сильно повышается эффективность применения такого подхода остается неисследованным. При этом представленный метод синтеза системы функционального контроля логических комбинационных схем приводит к незначительному усложнению технических средств диагностирования за счет применения тестеров самодвойственности и модулей сжатия парафазных сигналов в компараторе. Тем не менее, их структуры не являются слишком сложными. Сам метод дает более эффективные по показателю структурной избыточности системы функционального контроля для сложных комбинационных схем с большим числом выходов.

Так как на входах модулей сжатия парафазных сигналов *TRC* также формируются векторы кода «2 из 4» из множества {0101; 0110; 1001; 1010}, контроль схем по модифицированной структуре, предложенной в данной статье, может быть осуществлен без использования 2/4-*TSC* [25]. В этом случае методика построения системы функционального контроля остается той же.

Список литературы

[1]. Kharchenko V., Kondratenko Yu., Kacprzyk J. Green IT Engineering: Concepts, Models, Complex Systems Architectures. Springer Book series «Studies in Systems, Decision and Control», vol. 74, 2017, 305 p.

[2]. Pradhan D.K. Fault-Tolerant Computer System Design. New York, Prentice Hall, 1996, 560 p.

[3]. Nicolaidis M., Zorian Y. On-Line Testing for VLSI – A Compendium of Approaches. Journal of Electronic Testing: Theory and Application, vol. 12, issue 1-2, 1998, pp. 7-20.
[4]. Fujiwara E. Code Design for Dependable Systems: Theory and Practical Applications. John Wiley & Sons, 2006, 720 p.
[5]. Borecký J., Kohlík M., Kubátová H. Parity Driven Reconfigurable Duplex System. Microprocessors and Microsystems, vol. 52, issue C, 2017, pp. 251-260.
[6]. Гаврилов С.В., Тельпухов Д.В., Жукова Т.Д., Гуров С.И. Использование информационной избыточности при построении сбоеустойчивых комбинационных схем. Таврический вестник информатики и математики, том 2, №39, 2018 г., стр. 29-44.
[7]. Согомонян Е.С., Слабаков Е.В. Самопроверяемые устройства и отказоустойчивые системы. М, Радио и связь, 1989 г., 208 стр.
[8]. Efanov D., Sapozhnikov V., Sapozhnikov Vl. Generalized Algorithm of Building Summation Codes for the Tasks of Technical Diagnostics of Discrete Systems. In Proc. of the 15th IEEE East-West Design & Test Symposium (EWDTS`2017), 2017, pp. 365-371, doi: 10.1109/EWDTS.2017.8110126.
[9]. Сапожников В.В., Сапожников Вл.В. Самопроверяемые дискретные устройства. СПб., Энергоатомиздат, 1992 г., 224 стр.
[10]. Яблонский С.В. Введение в дискретную математикую Учеб. пособие для вузов. Под ред. В.А. Садовничего, 4-е изд., стер. М., Высшая школа, 2003 г., 384 стр.
[11]. Reynolds D.A., Meize G. Fault Detection Capabilities of Alternating Logic. IEEE Transactions on Computers, vol. C-27, issue 12, 1978, pp. 1093-1098.
[12]. Сапожников В.В., Сапожников Вл.В., Гессель М. Самодвойственные дискретные устройства. СПб., Энергоатомиздат, 2001 г., 331 стр.
[13]. Сапожников В.В., Сапожников Вл.В., Валиев Р.Ш. Синтез самодвойственных дискретных систем. СПб., Элмор, 2006 г., 220 стр.
[14]. Piestrak S.J. Design of Self-Testing Checkers for Unidirectional Error Detecting Codes. Wrocław, Oficyna Wydawnicza Politechniki Wrocławskiej, 1995, 111 p.
[15]. Гессель М., Мошанин В.И., Сапожников В.В., Сапожников Вл.В. Обнаружение неисправностей в самопроверяемых комбинационных схемах с использованием свойств самодвойственных функций. Автоматика и телемеханика, №12, 1997 г., стр. 193-200.
[16]. Гессель М., Дмитриев А.В, Сапожников В.В, Сапожников Вл.В. Самотестируемая структура для функционального обнаружения отказов в комбинационных схемах. Автоматика и телемеханика, №11, 1999 г., стр. 162-174.
[17]. Поспелов Д.А. Логические методы анализа и синтеза схем. 3-е изд., перераб. и доп., М., Энергия», 1974 г., 368 стр.
[18]. Сапожников В.В., Сапожников Вл.В., Ефанов Д.В. Построение самопроверяемых структур систем функционального контроля на основе равновесного кода «2 из 4». Проблемы управления, №1, 2017 г., стр. 57-64.
[19]. Efanov D., Sapozhnikov V., Sapozhnikov Vl. Method of Self-Checking Concurrent Error Detection System Development Based on Constant-Weight Code «2-out-of-4». In Proc. of the 3rd International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), 2017, pp. 1-6, doi: 10.1109/ICIEAM.2017.8076374.
[20]. Lala P.K. Self-Checking and Fault-Tolerant Digital Design. San Francisco, Morgan Kaufmann Publishers, 2001, 216 p.
[21]. Carter W.C., Duke K.A., Schneider P.R. Self-Checking Error Checker for Two-Rail Coded Data. United States Patent Office, filed July 25, 1968, ser. No. 747533, patented Jan. 26, 1971, N. Y., 10 p.
[22]. Гессель М., Дмитриев А.В., Сапожников В.В., Сапожников Вл.В. Обнаружение неисправностей в комбинационных схемах с помощью самодвойственного контроля. Автоматика и телемеханика, №7, 2000 г., стр. 140-149.
[23]. Сапожников В.В., Сапожников Вл.В. Самопроверяемые тестеры для равновесных кодов. Автоматика и телемеханика, №3, 1992 г., стр. 3-35.
[24]. Аксенова Г.П. Необходимые и достаточные условия построения полностью проверяемых схем свертки по модулю 2. Автоматика и телемеханика, №9, 1979 г., стр. 126-135.
[25]. Сапожников В.В., Сапожников Вл.В., Ефанов Д.В. Организация систем функционального контроля с обеспечением полной самопроверяемости структуры на основе модулей сжатия парафазных сигналов. Известия вузов. Приборостроение, том. 60, №5, 2017 г., стр. 404-411, DOI: 10.17586/0021-3454-2017-60-5-404-411.

Fault detection in combinational circuits based on self-dual complement to constant-weight code

[1, 2] *D.V. Efanov <TrES-4b@yandex.ru>*
[3] *V.V. Sapozhnikov <port.at.pgups@gmail.com>*
[3] *Vl.V. Sapozhnikov <at.pgups@gmail.com>*
[3] *D.V. Pivovarov <pivovarov.d.v.spb@gmail.com>*
[1] *OOO «LocoTech-Signal»,*
107113, Russia, Moscow, 3ed Rybinskaya st., 18, building 22
[2] *Russian University of Transport,*
127994, Russia, Moscow, Obraztsova st., 9
[3] *Emperor Alexandex I St. Petersburg state transport university,*
190031, Russia, St. Petersburg, Moscowsky ave., 9

Abstract. A new method of combinational circuits concurrent checking is considered based on Boolean complement of the working functions of the diagnosis object to the constant-weight code words with parallel checkout of the bits functions belonging to the self-dual functions class of the Boolean logic. The described approach to the organization of concurrent error-detection (CED) systems allows to increase the detection ability in comparison with the traditional checkout by constant-weight code. In this case, the undetectable error can be a such error, which will lead to the preservation of the code word weight generated at the outputs of the Boolean complement block, if the distortions of all the functions bits on the opposite sets is coincide. This condition is more stringent than the checkout of the vector belonging a pre-selected code, it means that it gives a greater probability of detecting distortion arising from errors. The described method allows checkout of combinational circuit based on constant-weight codes with the same number of single and zero bits. In this case, the priority is in the widely used «2-out-of-4» code, which has a simple checker structure. A detailed description of the new CED system structure is given, and algorithms for the extension of definition of the values of the checkout functions that provide testability of technical diagnostic tools are given.

Keywords: combinational circuits; fault detection; self-dual complement; constant-weight code; «2-out-of-4» code.

DOI: 10.15514/ISPRAS-2019-31(1)-8

For citation: Efanov D.V., Sapozhnikov V.V., Sapozhnikov Vl.V., Pivovarov D.V. Fault detection in combinational circuits based on self-dual complement to constant-weight code. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 1, 2019. pp. 115-132 (in Russian). DOI: 10.15514/ISPRAS-2019-31(1)-8

References

[1]. Kharchenko V., Kondratenko Yu., Kacprzyk J. Green IT Engineering: Concepts, Models, Complex Systems Architectures. Springer Book series «Studies in Systems, Decision and Control», vol. 74, 2017, 305 p.
[2]. Pradhan D.K. Fault-Tolerant Computer System Design. New York, Prentice Hall, 1996, 560 p.
[3]. Nicolaidis M., Zorian Y. On-Line Testing for VLSI – A Compendium of Approaches. Journal of Electronic Testing: Theory and Application, vol. 12, issue 1-2, 1998, pp. 7-20.
[4]. Fujiwara E. Code Design for Dependable Systems: Theory and Practical Applications. John Wiley & Sons, 2006, 720 p.
[5]. Borecký J., Kohlík M., Kubátová H. Parity Driven Reconfigurable Duplex System. Microprocessors and Microsystems, vol. 52, issue C, 2017, pp. 251-260.
[6]. Gavrilov S.V., Tel'puhov D.V., Zhukova T.D., Gurov S.I. Synthesis of fault-tolerant combination schemes by introducing information redundancy, Taurida Journal of Computer Science Theory and Mathematics [Tavricheskij vestnik informatiki i matematiki], vol. 2, issue 39, , 2018, pp. 29-44 (in Russian).
[7]. Sogomonyan E.S., Slabakov E.V. Self-Checking Devices and Fault-Tolerance Systems. Moscow, Radio and Communication [Radio i svyaz`], 1989, 208 p. (in Russian).

[8]. Efanov D., Sapozhnikov V., Sapozhnikov Vl. Generalized Algorithm of Building Summation Codes for the Tasks of Technical Diagnostics of Discrete Systems. In Proc. of the 15th IEEE East-West Design & Test Symposium (EWDTS`2017), 2017, pp. 365-371, doi: 10.1109/EWDTS.2017.8110126.

[9]. Sapozhnikov V.V., Sapozhnikov Vl.V. Self-Checking Discrete Devices. St. Petersburg, Energoatomizdat, 1992, 224 p. (in Russian).

[10]. Yablonskij S.V. Introduction to Discrete Mathematics: Textbook, Ed. V.A. Sadovnichiy, 4th edition, Moscow, High School [Vyschaya shkola], 2003, 384 p. (in Russian).

[11]. Reynolds D.A., Meize G. Fault Detection Capabilities of Alternating Logic. IEEE Transactions on Computers, vol. C-27, Issue 12, 1978, pp. 1093-1098.

[12]. Sapozhnikov V.V., Sapozhnikov Vl.V., Gyossel' M. Self-dual discrete devices. St. Petersburg, Energoatomizdat, 2001, 331 p. (in Russian).

[13]. Sapozhnikov V.V., Sapozhnikov Vl.V., Valiev R.Sh. Synthesis of self-dual systems, St. Petersburg, Elmor, 2006, 220 p. (in Russian).

[14]. Piestrak S.J. (1995) Design of Self-Testing Checkers for Unidirectional Error Detecting Codes. Wrocław: Oficyna Wydawnicza Politechniki Wrocławskiej, 1995, 111 p.

[15]. Gessel M., Moshanin V.I., Sapozhnikov V.V., Sapozhnikov Vl.V. Fault Detection in Self-Test Combination Circuits Using the Properties of Self-Dual Functions. Automation and Remote Control [Avtomatika i telemekhanika], issue 12, 1997, pp. 193-200 (in Russian).

[16]. Gessel' M., Dmitriev A.V, Sapozhnikov V.V, Sapozhnikov Vl.V. A functional fault-detection self-test for combinational circuits. Automation and Remote Control [Avtomatika i telemekhanika], issue 11, 1999, pp. 162-174 (in Russian).

[17]. Pospelov D.A. Logical methods of analysis and synthesis of circuits. 3-ed edition, Moscow, Energy [Energiya], 1974, 368 p. (in Russian).

[18]. Sapozhnikov V.V., Sapozhnikov Vl.V., Efanov D.V. Design of self-checking concurrent error detection systems based on «2-out-of-4» constant-weight code, Control Science [Problemy upravlenija], issue 1, 2017, pp. 57-64 (in Russian).

[19]. Efanov D., Sapozhnikov V., Sapozhnikov Vl. Method of Self-Checking Concurrent Error Detection System Development Based on Constant-Weight Code "2-out-of-4", In Proc. of the 3ed International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), 2017, pp. 1-6, doi: 10.1109/ICIEAM.2017.8076374.

[20]. Lala P.K. Self-Checking and Fault-Tolerant Digital Design. San Francisco, Morgan Kaufmann Publishers, 2001, 216 p.

[21]. Carter W.C., Duke K.A., Schneider P.R. Self-Checking Error Checker for Two-Rail Coded Dataю United States Patent Office, filed July 25, 1968, ser. No. 747533, patented Jan. 26, 1971, N. Y., 10 p.

[22]. Gessel' M., Dmitriev A.V., Sapozhnikov V.V., Sapozhnikov Vl.V. Detection of faults in combinational circuits by a self-dual test, Automation and Remote Control [Avtomatika i telemekhanika], issue 7, 2000, pp. 140-149 (in Russian).

[23]. Sapozhnikov V.V., Sapozhnikov Vl.V. Self-Checking Constant-Weights Codes Checkers, Automation and Remote Control [Avtomatika i telemekhanika], issue 3, 1992, pp. 3-35 (in Russian).

[24]. Aksyonova G.P. Necessary and sufficient conditions for the design of totally checking circuits of compression by modulo 2, Automation and Remote Control [Avtomatika i telemekhanika], Issue 9, 1979, pp. 126-135 (in Russian).

[25]. Sapozhnikov V.V., Sapozhnikov Vl.V., Efanov D.V. Organization of functional control systems with totally self-checking structure based on two-rail signals compression modules, Journal of Instrument Engineering [Izvestiya vuzov. Priborostroenie], vol. 60, issue 5, 2017, pp. 404-411, doi: 10.17586/0021-3454-2017-60-5-404-411 (in Russian).

Улучшение ранее известной верхней оценки для задачи Multiple Strip Packing и вероятностный анализ алгоритма для большого числа полос[1]

[1] *Д.О. Лазарев <dennis810@mail.ru>*
[1,2] *Н.Н. Кузюрин <nnkuz@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
[2] *Московский физико-технический институт,*
141700, Московская область, г. Долгопрудный, Институтский пер., 9

Аннотация. В работе рассмотрена задача упаковки прямоугольников в полосы единичной ширины Multiple Strip Packing. Рассмотрен аналог ранее предложенного алгоритма, алгоритм упаковки в области ограниченной высоты Limited Hash Packing и произведён его вероятностный анализ. Алгоритм онлайновый и работает в режиме closed-end, когда число прямоугольников, которые нужно упаковать известно алгоритму до начала работы. Лучшая из ранее известных оценок для математического ожидания площади полос, не заполненной прямоугольниками для задачи Multiple Strip Packing при числе полос $k \leq \sqrt{N}$, составляющая $E(S_{sp}) = O(\sqrt{N}\ln N)$ была улучшена до $E(S_{sp}) = O(\sqrt{N \ln N})$. Также был произведён анализ алгоритма в случае $k = \omega(\sqrt{N})$. Было доказано, что при любых k, N выполнено $E(S_{sp}) \geq \frac{k}{8} - \frac{\sqrt{N}}{4}$.

Ключевые слова: Strip Packing; Multiple Strip Packing; онлайновый алгоритм; режим closed-end; вероятностный анализ; алгоритм упаковки прямоугольников в ограниченные области; Limited Hash Packing.

DOI: 10.15514/ISPRAS-2019-31(1)-9

Для цитирования: Лазарев Д.О., Кузюрин Н.Н. Улучшение ранее известной верхней оценки для задачи Multiple Strip Packing и вероятностный анализ алгоритма для большого числа полос. Труды ИСП РАН, том 31, вып. 1, 2019 г., стр. 133-142. DOI: 10.15514/ISPRAS-2019-31(1)-9

1. Введение

Задача упаковки прямоугольников в полосу Strip Packing Problem и её обобщение – задача упаковки прямоугольников в несколько полос одинаковой ширины Multiple Strip Packing Problem(MSP) активно изучаются в последнее время благодаря их многочисленным практическим применениям.

При решении задачи построения расписания выполнения параллельных вычислительных задач на кластере или группе кластеров, согласно [1], можно представить каждую из задач в виде прямоугольника, высоте которого соответствует время выполнения задачи на кластере, а ширине соответствует число процессоров, необходимое для решения задачи на кластере. Таким образом, при наличии одного кластера задача построения расписания выполнения задач сводится к задаче Strip Packing, при наличии нескольких кластеров [2] задача построения расписаний для параллельных заданий сводится либо к задаче Multiple Strip Packing, либо к её обобщениям[3].

[1] Работа выполнена при финансовой поддержке РФФИ, проект 17-07-01006

Также интерес к задаче Strip Packing подкреплён практическими применениями задачи и её частных случаев в таких перспективных областях, как задачи раскроя и перевозки материалов, грид-технологии[4], [5], облачные вычисления [6] и распределение памяти [7].

2. Постановка задачи Multiple Strip Packing и известные результаты

Задача Strip Packing является геометрическим обобщением классической задачи Bin Packing или задачи упаковки в контейнеры. Задача Bin Packing - одна из первых известных NP-полных в сильном смыле задач [8]. Для неё было составлено множество приближённых алгоритмов для анализа в худшем случае [9], [10] и в среднем [11], [12].

При анализе задачи в среднем сначала рассматривались шельфовые алгоритмы, разбивающие полосу на слои(шельфы) и распределяющие по шельфам прямоугольники в зависимости от их высоты и внутри шельфа пакующие прямоугольники некой эвристикой для задачи Bin Packing. Однако в [13] Коффманом и Шором было показано, что для любого шельфового алгоритма математическое ожидание не заполненной прямоугольниками площади полосы $E(S_{sp}) = \Omega\left(N^{\frac{2}{3}}\right)$.

В работе [13] был предложен closed-end алгоритм с точностью $O\left(N^{\frac{2}{3}}\right)$, а 2011 году в [14] Кузюриным и Поспеловым был предложен open-end алгоритм с точностью $\theta\left(N^{\frac{2}{3}}\right)$. В работе [15] Трушниковым был предложен новый closed-end алгоритм, точность которого была исследована в работе [16], где было показано, что $E(S_{sp}) = O(N^{\frac{1}{2}}\log^{\frac{3}{2}} N)$. В [17] оценка была улучшена до $E(S_{sp}) = O\left(N^{\frac{1}{2}}\ln\ N\right)$ и обобщен3

а на случай задачи Multiple Strip Packing с числом полос $k \le \sqrt{N}$. В настоящей работе показано, что

$$E(S_{sp}) = O\left(\sqrt{N\ln N}\right) \text{ при числе полос } k \le \sqrt{N} \qquad (1)$$

Значительное уменьшение величины $E(S_{sp})$ в алгоритме из [15] по сравнению с ранее известными алгоритмами обусловлено ограничением общей высоты разбиения на области и усовершенствованным способом упаковки прямоугольники в области.

Постановка задачи Multiple Strip Packing:

В набор из k полубесконечных прямоугольных полос $C_1, .., C_k$ единичной ширины требуется упаковать без пересечений и без вращений открытые прямоугольники $a_1, \ldots, a_N$ со сторонами, параллельными основаниям или сторонам полос, длины сторон прямоугольников не превосходят 1. Требуется минимизировать высоту упаковки H, или наибольшую высоту верхней стороны замыканий одного из прямоугольников.

Будем рассматривать онлайновые алгоритмы упаковки, не знающие длин сторон прямоугольников $a_{i+1}, \ldots, a_N$ при размещении i-го прямоугольника a_i.

Алгоритм работает в режиме closed-end, т.е. ему известно число N прямоугольников, которые нужно упаковать до начала работы.

Производим вероятностный анализ алгоритма, т.е. считаем, что длины сторон прямоугольников – независимые случайные величины, имеющие равномерное распределение на отрезке $[0, 1]$. Ошибку алгоритма оцениваем через математическое ожидание $E(S_{sp})$, где S_{sp} – площадь незаполненной прямоугольниками части полос от нижнего основания полос до наибольшей высоты верхней стороны H. Верно, что

$$E(S_{sp}) = kE(H) - E(S_r) = kE(H) - \frac{N}{4}$$

где S_r- суммарная площадь всех прямоугольников, а k- число полос.

При наличии всего одной полосы $k = 1$ задача Multiple Strip Packing вырождается в задачу Strip Packing.

3. Алгоритм упаковки в области ограниченной высоты (Limited Hash Packing)

1. Разбиение на области. Выберем $\boldsymbol{s} = \left\lfloor \frac{\sqrt{N}}{\boldsymbol{k}} \right\rfloor$. Верно, что

$$s \in \begin{cases} \left[\frac{\sqrt{N}}{k}, 2\frac{\sqrt{N}}{k} \right], k \le \sqrt{N} \\ 1, k > \sqrt{N} \end{cases}$$

Разобьём нижнюю часть всех полос высотой $\frac{N}{4k}$ на $2sk - 1$ области с высотами, равными $\frac{N}{4sk}$, и значениями ширины областей, равными $\forall j = \overline{1, 2sk - 1}$, как показано на рис. 1.

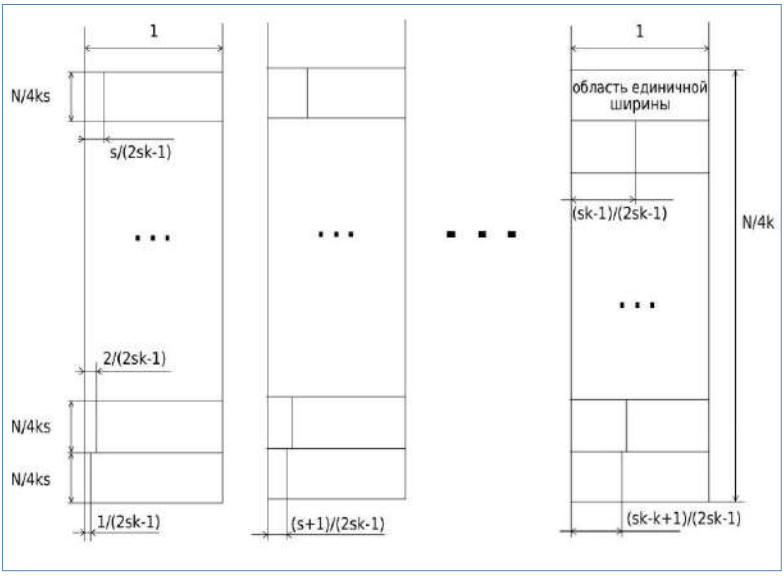


Рис. 1. Разбиение нижней части полос на прямоугольные области
Fig. 1. Breaking the bottom of the strips into rectangular areas

Пронумеруем области: область имеет номер $i (i = \overline{1, 2sk - 1})$, если её ширина равна $\frac{i}{2sk-1}$. Так, самая тонкая область имеет номер один, а самая толстая, единичной ширины, имеет номер $2sk - 1$.

2. Упаковка прямоугольников. Скажем, что прямоугольник R имеет тип $j, j = \overline{1, 2sk - 1}$, если его ширина $w(R) \in \left(\frac{i-1}{2sk-1}, \frac{i}{2sk-1} \right]$. Минимальная область, в которую прямоугольник типа i вмещается, есть область с номером i.

Алгоритм упаковки в области ограниченной **высоты (Limited Hash Packing):**

1) Если выпал прямоугольник R типа i, и он помещается в область с номером i, то размещаем R в эту область на верх текущей упаковки в данной области.
2) Если прямоугольник R типа i не помещается в область с номером i, то размещаем его на верх текущей упаковки в область с минимальным номером, куда его можно поместить, если это возможно.

3) Иначе, если R не помещается ни в одну область (такие прямоугольники назовём **выпавшими**), то помещаем его на верх текущей упаковки в полосу (т.е. нижняя сторона R помещается не ниже части полос, разбитой на области, и лежит на самой нижней (по всем полосам) верхней стороне самого высокого прямоугольника в полосе, помещённого, также как и R, выше всех областей, если такой прямоугольник есть, и на верх самой высокой области в одной из полос, если ту полосу ещё не упаковано выпавших прямоугольников), заполненную не выше всех других полос. Для прямоугольника в полосе выбираем такое положение, чтобы его левая сторона касалась левого края полосы.

4. Верхняя оценка математического ожидания площади незаполненной части полос $E(S_{sp})$

Следующая Лемма была доказана в случае $\alpha \leq 1$ в работе [16].

Лемма 1. Пусть случайная величина $X = X_1 + X_2 + \cdots + X_k$, где $X_i = \xi_i \eta_i$, ξ_i принимает значение 1 с вероятностью p и 0 с вероятностью $1 - p$,- η_i — равномерно распределенная на отрезке $(0, 1]$ случайная величина, причем все случайные величины $\xi_i, \eta_i, i = \overline{1, \dots, k}$ независимы в совокупности. Тогда для любого α из интервала $(0, 1]$ выполняется неравенство:

$$\mathbb{P}\{X > (1+\alpha)EX\} \leq e^{-\frac{5}{9}\alpha^2 EX}$$

При $\alpha > 1$ верно, что

$$\mathbb{P}\{X > (1+\alpha)EX\} \leq e^{-\frac{1}{3}\alpha EX}$$

Доказательство.

Докажем утверждение в случае $\alpha > 1$. В работе [16] было показано, что для любых $\alpha, t \geq 0$

$$\mathbb{P}\{X > (1+\alpha)EX\} \leq \exp\left\{\left(2\frac{e^t - 1}{t} - 2 - t(1+\alpha)\right)\frac{pk}{2}\right\}$$

В данную формулу подставим $t = 1$, учитывая, что $\alpha > 1$, Получим:

$$\mathbb{P}\{X > (1+\alpha)EX\} \leq \exp\left\{\frac{pk}{2}(2e - 5 - \alpha)\right\} \leq \exp\left\{\frac{pk}{2}\left(-\frac{\alpha}{3}\right)\right\} = e^{-\frac{1}{3}\alpha EX} \blacksquare$$

Теорема 1 (Оценки E_{sp} для алгоритма Limited Hash Packing) Для алгоритма Limited Hash Packing при упаковке N прямоугольников в k полос,

$$E(S_{sp}) \leq 3k + 6\sqrt{N \ln N} = O(\max\{k, \sqrt{N \ln N}\})$$

Обозначения.

H- высота упаковки прямоугольников в полосы.

$h_{max} = H - \frac{N}{4k}$ – максимальная суммарная высота выпавших(т.е. упакованных не в области, а выше областей) прямоугольников, упакованных в одну полосу.

h_{min} – минимальная суммарная высота выпавших прямоугольников, упакованных в одну полосу.

h – суммарная высота всех выпавших прямоугольников.

l_i – суммарная высота прямоугольников типов $2sk - 1, \dots, 2sk - i$.

Прямоугольник R имеет тип $j, j = \overline{1, 2sk - 1}$, если его ширина $w(R) \in \left(\frac{i-1}{2sk-1}, \frac{i}{2sk-1}\right]$.

Область имеет номер $i (i = \overline{1, 2sk - 1})$, если её ширина равна $\frac{i}{2sk-1}$.

Область с номером j- **не переполнялась**, если и только если любой прямоугольник типа $k \in \{1, 2, \dots, j\}$ был упакован в область с номером $m \in \{k, k+1, \dots, j\}$.

Доказательство. Так как суммарная площадь областей равна $\frac{N}{4}$, то по формуле 1 для доказательства достаточно показать, что

$$kE(H) - \frac{N}{4} = E(h_{max})k \le 3k + 6\sqrt{N \ln N} \quad (2)$$

Так как по построению $h_{min} + 1 \ge h_{max}$, а $h \ge kh_{min}$, то

$$h_{max} \le \frac{h}{k} + 1 \quad (3)$$

Оказывается, что верна оценка

$$h \le \max_i \left\{ l_i - \frac{iN}{4sk} + i \right\} \quad (4)$$

Действительно, скажем, что область с номером j- не переполнялась, если и только если любой прямоугольник типа $p \in \{1, 2, \dots, j\}$ был упакован в область с номером $m \in \{p, p+1, \dots, j\}$. Если в область упаковали прямоугольники суммарной высотой не более, чем $\frac{N}{4sk} - 1$, что ровно на единицу меньше, чем высота области, то область не переполнена(хотя обратное – не верно).

Рассмотрим $j_0 =$ наибольшее j: область с номером $j -$ не переполнялась. Если такой области нет, то полагаем $j = 0$. Тогда, во-первых, ни в областях с номерами $k \in \{j_0 + 1, j_0 + 2, \dots, 2sk - 1\}$, ни среди выпавших прямоугольников нет прямоугольников типа $j' \le j_0$. И, во-вторых, любая из областей с номером $j'' > j_0$ заполнена не менее, чем на $\frac{N}{4sk} - 1$, иначе, $j'' = j_0$. Таким образом, верно, что $h \le l_{2ks-j_0-1} - \left(2ks - j_0 - \frac{1N}{4sk}\right) + 2ks - j_0 - 1$, стало быть, верна формула 4.

Из формул 3 и 4 получаем, что $kh_{max} \le h + k \le \max_i \left\{ l_i - \frac{iN}{4sk} + i \right\} + k$. По формуле 2 с учётом того, что $i \le max\{2k, 2\sqrt{N}\} < 2k + 2\sqrt{N}$ и того, что,

$$E(h_{max})k \le E\left(\max_i \left\{ l_i - \frac{iN}{4sk} + i \right\}\right) + k \le 3k + 6\sqrt{N \ln N} \Leftrightarrow$$

$$\Leftrightarrow E\left(\max_i \{l_i + i\}\right) \le 2k + 6\sqrt{N \ln N} + \frac{iN}{4sk}$$

достаточно доказать, что верна формула 5:

$$E\left(\max_i \left\{ l_i - \frac{iN}{4sk} \right\}\right) \le 4\sqrt{N \ln N} \quad (5)$$

Докажем, что при достаточно больших N

$$\mathbb{P}\left(l_i \le \frac{Ni}{4sk} + 3\sqrt{N \ln N} \ \forall i = \overline{1,2sk-1}\right) \ge 1 - N^{-2} \quad (6)$$

Заметим, что $E(l_i) = \frac{Ni}{4sk}$. Значит, по лемме 1,

$$\mathbb{P}\left(l_i \ge \frac{Ni}{4sk} + 3\sqrt{N \ln N}\right) = \mathbb{P}\left(l_i \ge E(l_i) + 3\sqrt{N \ln N}\right) = P\left(L_i \ge (1+\alpha)E(l_i)\right),$$

$$\alpha = 3\sqrt{N \ln N}\frac{4ks}{Ni} = \frac{12ks\sqrt{\ln N}}{i\sqrt{N}}$$

Рассмотрим 2 случая:

1) $\alpha \le 1$. По лемме 1), учитывая, что $i \le 2ks - 1 < 2ks$ имеем:

$$\mathbb{P}\left(l_i \ge \frac{Ni}{4sk} + 3\sqrt{N \ln N}\right) = \mathbb{P}\left(l_i \ge \left(1 + \frac{12ks\sqrt{\ln N}}{i\sqrt{N}}\right)E(l_i)\right) \le$$

$$\le \exp\left\{-\frac{5}{9}\frac{144k^2s^2 \ln N}{Ni^2}\frac{Ni}{4ks}\right\} = \exp\left\{-\frac{20ks \ln N}{i}\right\} < \exp\left\{-\frac{20ks \ln N}{2ks}\right\} =$$

$$= N^{-10} < N^{-3}$$

2) $\alpha > 1$. $E(l_i)\alpha = 3\sqrt{N \ln N}$. Значит, по Лемме 1, для достаточно больших N имеем:

$$\mathbb{P}\left(l_i \geq \frac{Ni}{4sk} + 3\sqrt{N \ln N}\right) \leq \exp\left\{-\frac{E(l_i)\alpha}{3}\right\} = \exp\{-\sqrt{N \ln N}\} \leq N^{-3}$$

Так как $i \leq N$, получаем формулу 6:

$$\mathbb{P}\left(l_i \geq \frac{Ni}{4sk} + 3\sqrt{N \ln N}\right) \leq N^{-3} \Rightarrow$$

$$\mathbb{P}\left(l_i \leq \frac{Ni}{4sk} + 3\sqrt{N \ln N} \ \forall i = \overline{1,2sk-1}\right) \geq 1 - N^{-2}$$

Значит, так как в любом случае $l_i \leq N$, для достаточно больших N имеем:

$$E\left(\max_i\{l_i\} - \frac{iN}{4ks}\right) \leq 3\sqrt{N \ln N} + \mathbb{P}\left(l_i \geq \frac{Ni}{4sk} + 3\sqrt{N \ln N} \ \forall i = \overline{1,2sk-1}\right)N \leq$$

$$\leq 4\sqrt{N \ln N}$$

Отсюда следует формула 5, из которой, в свою очередь, следует утверждение теоремы. ■

5. Нижние оценки $E(S_{sp})$ для алгоритма Limited Hash Packing при упаковке в $k = \Omega(\sqrt{N})$ полос

Теорема 2. Покажем, что после выпадения N прямоугольников,

$$E(S_{sp}) \geq \frac{k}{8} - \frac{\sqrt{N}}{4}$$

Обозначения.

$X = X_1 + \cdots + X_N$- сумма высот всех прямоугольников, где X_i-имеет равномерное на $[0, 1]$ распределение.

H – высота упаковки алгоритмом Limited Hash Packing.

Прямоугольник назовём **выпавшим**, если он помещён алгоритмом выше части полос, разбитой на области.

Доказательство.

Покажем, что с вероятностью $P \geq \frac{1}{4}$ один из выпавших прямоугольников имеет высоту $\geq \frac{1}{2}$ и сумма высот всех прямоугольников $X \geq \frac{N}{2}$. Покажем, что отсюда будет следовать утверждение теоремы.

Ввиду симметричности распределения X относительно математического ожидания $E(X) = \frac{N}{2}$, верно, что $\frac{N}{2} - E\left(X \middle| X \leq \frac{N}{2}\right) = E(|X - EX|) \leq \sqrt{Var(X)} < \frac{\sqrt{N}}{2}$.

Таким образом, обозначив за $E(X|A)$ математическое ожидание X при условии того, что случайное событие A произошло и учитывая, что, если $X > \frac{N}{2}$, то хотя бы 1 прямоугольник выпал, и, следовательно, так как высота упаковки H не менее, чем $\frac{N}{4k}$, $E(H)k - \frac{N}{4} > 0$, по формуле полной вероятности, имеем:

$$E(S_{sp}) = E(H)k - \frac{N}{4} = \mathbb{P}\left(X \leq \frac{N}{2}\right)\left(kE\left(H \middle| X \leq \frac{N}{2}\right) - \frac{N}{4}\right) +$$

$$+\mathbb{P}\left(\left(X > \frac{N}{2}\right) \cap \left(H < \frac{N}{4k} + \frac{1}{2}\right)\right)\left(kE\left(H \middle| \left(X > \frac{N}{2}\right) \cap \left(H < \frac{N}{4k} + \frac{1}{2}\right)\right) - \frac{N}{4}\right) +$$

$$+\mathbb{P}\left(\left(X > \frac{N}{2}\right) \cap \left(H \geq \frac{N}{4k} + \frac{1}{2}\right)\right)\left(kE\left(H \middle| \left(X > \frac{N}{2}\right) \cap \left(H \geq \frac{N}{4k} + \frac{1}{2}\right)\right) - \frac{N}{4}\right) \geq$$

$$-\frac{1}{4}\sqrt{N} + 0 + \frac{1}{4}\frac{k}{2} = -\frac{\sqrt{N}}{4} + \frac{k}{8}$$

при $\mathbb{P}\left(\left(X \geq \frac{N}{2}\right) \cap \left(H \geq \frac{N}{4k} + \frac{1}{2}\right)\right) \geq \frac{1}{4}$

Докажем, что с вероятностью $P \geq \frac{1}{4}$ верно, что $\left(X \geq \frac{N}{2}\right)$и $\left(H \geq \frac{N}{4k} + \frac{1}{2}\right)$, и тем самым докажем теорему.

Рассмотрим 2 типа случайных событий в предположении, что $X \geq \frac{N}{2}$:

1) $A(i)$- первый выпавший прямоугольник имел номер $i, 1 \leq i \leq N$ и его высота была больше $\frac{1}{2}$;
2) $B(i)$- первый выпавший прямоугольник имел номер $i, 1 \leq i \leq N$ и его высота была не больше $\frac{1}{2}$.

Верно, что $\mathbb{P}(A(i)) \geq \mathbb{P}(B(i))$, так как если мы рассмотрим набор длин сторон прямоугольников $h_1, w_1, \dots, h_N, w_N$, такой, что событие $B(i)$ имеет место, то для набора сторон $h_1, w_1, \dots, h_i + \frac{1}{2}, w_i, \dots, h_N, w_N$ имеет место событие $A(i)$. Действительно, если при упаковке прямоугольников алгоритмом Limited Hash Packing i-ый прямоугольник- первый вылевший и его высота $h_i < \frac{1}{2}$, то при тех же параметрах остальных прямоугольников $h_i \geq \frac{1}{2}$, то прямоугольник с номером i- также первый вылетевший.

Верно, что $A(i) \cap A(j) = B(i) \cap B(j) = \emptyset, i \neq j$ и $A(i) \cap B(j) = \emptyset$ и

$$\bigcup_{i=1}^{N} A(i) \in \left(X \geq \frac{N}{2}\right) \cap \left(H \geq \frac{N}{4k} + \frac{1}{2}\right) \text{ и}$$

$$\left(X \geq \frac{N}{2}\right) \cap \left(H < \frac{N}{4k} + \frac{1}{2}\right) \in \bigcup_{i=1}^{N} B(i)$$

Следовательно, $\mathbb{P}\left(\left(X \geq \frac{N}{2}\right) \cap \left(H \geq \frac{N}{4k} + \frac{1}{2}\right)\right) \geq \mathbb{P}\left(\left(X \geq \frac{N}{2}\right) \cap \left(H < \frac{N}{4k} + \frac{1}{2}\right)\right)$. Так как сумма последних двух вероятностей равна $\frac{1}{2}$, верно, что $\mathbb{P}\left(\left(X \geq \frac{N}{2}\right) \cap \left(H \geq \frac{N}{4k} + \frac{1}{2}\right)\right) \geq \frac{1}{4}$, откуда и следует утверждение теоремы. ■

6. Заключение

Для задачи упаковки прямоугольников в полосы одинаковой ширины Multiple Strip Packing и её частного случая – задачи упаковки прямоугольников в полосу единичной ширины Strip Packing был произведён вероятностный анализ алгоритма упаковки в области ограниченной высоты Limited Hash Packing. Данный алгоритм является обобщением на случай упаковки в несколько полос алгоритма, предложенного Трушниковым в [15]. Алгоритм онлайновый, работает в режиме closed-end, т.е. ему известно число прямоугольников N, которое требуется разместить до начала работы.

Во многих алгоритмах для задач Strip Packing и Multiple Strip Packing прямоугольники разбиваются на группы по ширине, прямоугольники из одной группы упаковываются по высоте в области одного типа. Принципиальным отличием алгоритма, рассматриваемого в настоящей работе и алгоритмов того же типа, который мы называем алгоритмами упаковки в области ограниченной высоты, является тот факт, что после переполнения одной из областей, новая область не создаётся, а прямоугольник кладётся в самую узкую область, куда он помещается, если таковая существует.

В этой работе лучшая из ранее известных для задачи Multiple Strip Packing при числе полос $k \leq \sqrt{N}$, составляющая $E(S_{sp}) = O(\sqrt{N} \ln N)$ была улучшена до $E(S_{sp}) = O(\sqrt{N \ln N})$.

Также был произведён анализ алгоритма в случае $k = \Omega(\sqrt{N})$. Было доказано, что при любых k, N выполнено $E(S_{sp}) \geq \frac{k}{8} - \frac{\sqrt{N}}{4}$.

Список литературы

[1]. Кузюрин Н., Грушин Д., Фомин С. Проблемы двумерной упаковки и задачи оптимизации в распределенных вычислительных системах. Труды ИСП РАН, том 26, вып. 1, 2014 г., стр. 483–502, DOI: 10.15514/ISPRAS-2014-26(1)-21.

[2]. С.Н. Жук. О построении расписаний выполнения параллельных задач на группах кластеров с различной производительностью. Труды ИСП РАН, том 23, 2012, стр. 447-454, DOI: 10.15514/ISPRAS-2012-23-27.

[3]. S.N. Zhuk. Approximate algorithms for packing rectangles into several strips. Discrete Mathematics and Applications, vol 18, issue 1, 2006, pp. 91-105.

[4]. Tchernykh A., Schwiegelshohn U., Yahyapour R., Kuzjurin N. On-line hierarchical job scheduling on grids with admissible allocation. Journal of Scheduling, vol. 13, issue 5, 2010, pp. 545-552.

[5]. Tshernykh A., Ramirez J.M., Avetisyan A., Kuzjurin N., Grushin D., Zhuk S. Two-Level Job-Scheduling strategies for a Computational Grid. Lecture Notes in Computer Science, vol. 3911, 2005, pp. 774-781.

[6]. Cohil B., Shah S., Goleshha Y., Patel D. A Comparative Analysis of Virtual Machine Placement Techniques in the Cloud Environment. International Journal of Computer Applications, vol. 156, no. 14, 2016, pp. 12-18.

[7]. Garey M.R. Graham R.L., Ullman J.D., Worst-case analysis of memory allocation algorithms. In Proc. of the fourths annual ACM symposium on theory of computing (STOC ’72), 1972, pp. 143-150.

[8]. Garey M.R., Johnson D.S. Computers and intractability: a guide to the theory of NP-completeness. Freeman, San Francisco, 1979, 338 p.

[9]. Johnson D.S. Near-optimal Bin Packing Algorithms. PhD Thesis, Massachusetts Institute of Technology, Department of Mathematics, Cambridge, 1973, 401 p.

[10]. Vliet A. An improved lower bound for on-line bin packing algorithms. Journal of information Processing Letters, vol. 43, issue 5, 1992, pp. 277-284.

[11]. Coffman E.G., Courcoubetis C., Garey M.R., Johnson D.S., McGeoch L.A., Shor P.W., Weber R. and Yannakakis M. Fundamental discrepancies between average-case analysis under discrete and continuous distributions: a bin packing study. In Proc. of the twenty-third annual ACM symposium on Theory of Computing (STOC '91), 1991, pp. 230-240.

[12]. Shor P.W. How to pack better than Best Fit: tight bounds for average-case online Bin Packing. In Proc. of the 32nd Annual Symposium of foundations of Computer Science, 1991, pp. 752-759.

[13]. Coffman E.G., Shor P.W. Packing in two dimensions: Asymptotic average-case analysis of algorithms. Algorithmica, vol. 9, issue 3, 1993, pp. 253-277.

[14]. Кузюрин Н.Н., Поспелов А.И. Вероятностный анализ нового класса алгоритмов упаковки прямоугольников в полосу. Журнал вычислительной математики и математической физики, том 51, no. 10, 2011, стр. 1931-1936.

[15]. Трушников М.А. Об одной задаче Коффмана-Шора, связанной с упаковкой прямоугольников в полосу. Труды ИСП РАН, том 22, 2012 г., стр. 456-462, doi: 10.15514/ISPRAS-2012-22-24.

[16]. Трушников М.А. Вероятностный анализ нового алгоритма упаковки прямоугольников в полосу. Труды ИСП РАН, том 24, 2013 г., стр. 457-468, doi: 10.15514/ISPRAS-2013-24-21/

[17]. Лазарев Д.О., Кузюрин Н.Н. Алгоритм упаковки прямоугольников в несколько полос и анализ его точности в среднем. Труды ИСП РАН, том 29, выпуск 6, 2017 г., стр. 221-228, doi:: 10.15514/ISPRAS-2017-29(6)-13

An improvement of previously known upper bound of Multiple Strip Packing problem and probabilistic analysis of algorithm in case of large number of strips given

[1] *D.O. Lazarev <dennis810@mail.ru>*
[1,2] *N.N. Kuzyurin <nnkuz@ispras.ru>*
[1] *Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
[2] *Moscow Institute of Physics and Technology,*
Dolgoprudnyj, Institutskij alley, Moscow region, 141700, Russia

Abstract. In this article, an analog of previously proposed algorithm Limited Hash Packing for Multiple Strip Packing Problem is studied using probabilistic analysis. Limited Hash Packing is an on-line algorithm, which works in closed-end mode, knowing the number $\boldsymbol{N}$ of rectangles it has to pack before knowing the heights and width of the first rectangle. The algorithm proposes that width and heights of all rectangles have a uniform on $[\mathbf{0},\mathbf{1}]$ distribution and works in two stages. Firstly, it divides the k strips into $\boldsymbol{d}=\boldsymbol{\Theta}(\mathbf{max}\{\boldsymbol{k},\sqrt{\boldsymbol{N}}\})$ rectangular areas width of which equal $\frac{\boldsymbol{i}}{\boldsymbol{d}},\forall \boldsymbol{i}=\overline{\mathbf{1},\ldots,\boldsymbol{d}}$ such that the sum space of all this areas equals the expected space of all rectangles, $\frac{\boldsymbol{N}}{\mathbf{4}}$. Secondly, it packs a rectangle area of minimal width, in which it fits, or, if rectangle doesn't fit in any area, above all areas. It was shown, that for any number of strips $\boldsymbol{k}$ and any number of rectangles $\boldsymbol{N}$, the expected value of space not filled with rectangles of all strips from their lowest point to the highest point of the highest rectangle, $\boldsymbol{E}(\boldsymbol{S_{sp}}) \leq \mathbf{6}\sqrt{\boldsymbol{N}\ln\boldsymbol{N}}+\mathbf{3}\boldsymbol{k}$. It was also shown, that $\boldsymbol{E}(\boldsymbol{S_{sp}}) \geq \frac{\boldsymbol{k}}{\mathbf{8}}-\frac{\sqrt{\boldsymbol{N}}}{\mathbf{4}}$. This result proves that the previous bound is asymptotically tight in case when packing $\boldsymbol{N}$ rectangles into $\boldsymbol{k} \geq \sqrt{\boldsymbol{N}\ln\boldsymbol{N}}$ strips.

Keywords: on-line algorithm; closed-end; probabilistic analysis; closed-end mode; Multiple Strip Packing; an algorithm for packing into limited areas Limited Hash Packing.

DOI: 10.15514/ISPRAS-2019-31(1)-9

For citation: Lazarev D.O., Kuzyurin N.N. An improvement of previously known upper bound of Multiple Strip Packing problem and probabilistic analysis of algorithm in case of large number of strips given. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 1, 2019. pp. 133-142 (in Russian). DOI: 10.15514/ISPRAS-2019-31(1)-9

References

[1]. Two-dimensional packing problems and optimization in distributed computing systems N.N. Kuzyurin, D.A. Grushin, S.A. Fomin, Trudy ISP RAS/Proc. ISP RAS, vol. 26, issue 1, 2015, pp. 483–502(in Russian). DOI: 10.15514/ISPRAS-2014-26(1)-21

[2]. Zhuk S.N. On-line algorithm for scheduling parallel tasks on a group of related clusters. Trudy ISP RAN/Proc. ISP RAS, vol. 23, 2012, pp. 447-454 (in Russian). DOI: 10.15514/ISPRAS-2012-23-27

[3]. S.N. Zhuk. Approximate algorithms for packing rectangles into several strips. Discrete Mathematics and Applications, vol 18, issue 1, 2006, pp. 91-105.

[4]. Tchernykh A., Schwiegelshohn U., Yahyapour R., Kuzjurin N. On-line hierarchical job scheduling on grids with admissible allocation. Journal of Scheduling, vol. 13, issue 5, 2010, pp. 545-552.

[5]. Tshernykh A., Ramirez J.M., Avetisyan A., Kuzjurin N., Grushin D., Zhuk S. Two-Level Job-Scheduling strategies for a Computational Grid. Lecture Notes in Computer Science, vol. 3911, 2005, pp. 774-781.

[6]. Cohil B., Shah S., Goleshha Y., Patel D. A Comparative Analysis of Virtual Machine Placement Techniques in the Cloud Environment. International Journal of Computer Applications, vol. 156, no. 14, 2016, pp. 12-18.

[7]. Garey M.R. Graham R.L., Ullman J.D., Worst-case analysis of memory allocation algorithms. In Proc. of the fourths annual ACM symposium on theory of computing (STOC '72), 1972, pp. 143-150.

[8]. Garey M.R., Johnson D.S. Computers and intractability: a guide to the theory of NP-completeness. Freeman, San Francisco, 1979, 338 p.

[9]. Johnson D.S. Near-optimal Bin Packing Algorithms. PhD Thesis, Massachusetts Institute of Technology, Department of Mathematics, Cambridge, 1973, 401 p.

[10]. Vliet A. An improved lower bound for on-line bin packing algorithms. Journal of information Processing Letters, vol. 43, issue 5, 1992, pp. 277-284.

[11]. Coffman E.G., Courcoubetis C., Garey M.R., Johnson D.S., McGeoch L.A., Shor P.W., Weber R. and Yannakakis M. Fundamental discrepancies between average-case analysis under discrete and continuous distributions: a bin packing study. In Proc. of the twenty-third annual ACM symposium on Theory of Computing (STOC '91), 1991, pp. 230-240.

[12]. Shor P.W. How to pack better than Best Fit: tight bounds for average-case online Bin Packing. In Proc. of the 32nd Annual Symposium of foundations of Computer Science, 1991, pp. 752-759.

[13]. Coffman E.G., Shor P.W. Packing in two dimensions: Asymptotic average-case analysis of algorithms. Algorithmica, vol. 9, issue 3, 1993, pp. 253-277.

[14]. Kuzjurin N.N., Pospelov A.I. Probabilistic analysis of a new class of strip packing algorithms. Computational Mathematics and Mathematical Physics, vol. 51, no. 10, 2011, pp. 1817–1822.

[15]. Trushnikov M.A. On one problem of Koffman-Shor connected to strip packing problem. Trudy ISP RAN/Proc. ISP RAS, vol. 22, 2012, pp. 456-462, doi: 10.15514/ISPRAS-2012-22-24

[16]. Trushnikov M.A. Probabilistic analysis of a new strip packing algorithm. Trudy ISP RAN/Proc. ISP RAS, vol. 24, 2013, pp. 457-468, doi: 10.15514/ISPRAS-2013-24-21

[17]. Lazarev D.O., Kuzyrin N.N. An algorithm for Multiple Strip Package and its average case evaluation. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 6, 2017. pp. 221-228 (in Russian). DOI: 10.15514/ISPRAS-2017-29(6)-13