ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

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

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

СОДЕРЖАНИЕ

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

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индекс по Объединенному каталогу "Пресса России" — **22765**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru/prin/rus E-mail: prin@novtex.ru
Журнал включен в систему Российского индекса научного цитирования и базу данных RSCI на платформе Web of Science.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

Vol. 10
N 5
2019

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

CONTENTS

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

УДК 004.4'2 DOI: 10.17587/prin.10.195-203

С. Д. Махортов, д-р физ.-мат. наук, зав. каф., e-mail: msd_exp@outlook.com,
А. А. Ногих, магистрант, e-mail: a.nogikh@yandex.ru, Воронежский государственный университет

Применение LP-структур для автоматизации рефакторинга объектно-ориентированных программ*

Рассмотрены вопросы автоматизации рефакторинга программ, основанных на парадигме объектно-ориентированного программирования. В качестве математической основы подхода предложен специальный класс алгебраических систем — LP-структуры на решетках типов. Обоснован способ автоматизации рефакторинга, учитывающий свойства отдельных типов и их атрибутов. Описаны условия, достаточные для проведения практической оптимизации иерархии типов программной системы.

Ключевые слова: *рефакторинг, объектно-ориентированное программирование, иерархия типов, LP-структуры, автоматизация, инструментальные средства разработки*

Введение

По некоторым оценкам, доля затрат на сопровождение программного обеспечения (ПО) превышает половину общей суммы ресурсозатрат на протяжении всего жизненного цикла ПО. Одним из ключевых факторов, определяющих трудоемкость сопровождения ПО, является качество исходного кода программ [1].

С течением времени ПО совершенствуется, адаптируется к новым требованиям, вынужденно уходит от изначально спроектированной структуры. Этот процесс неизбежно сопровождается ухудшением кода, что делает дальнейшие изменения программной системы все более сложными. Остановить деградацию помогает рефакторинг — такое преобразование программной системы, при котором ее внешнее поведение сохраняется, а внутренняя структура улучшается [2].

В общем случае рефакторинг состоит из множества этапов, а именно выбора частей системы и способов преобразования; оценки того, насколько при этом сохранится поведение модифицируемых частей, проведение изменений, оценки полученного результата и т. д. [3]. Часть этих этапов могут выполнять автоматизированные системы рефакторинга, позволяя тем самым существенно снизить затраты на сопровождение ПО.

В работе [4] показано, что математической основой решения задач рефакторинга может служить теория LP-структур, предоставляющая эффективные алгебраические модели для различного рода систем в информатике. Описанный подход позволяет проводить автоматическую оптимизацию иерархии типов программной системы, включая устранение дублирования кода путем "подъема" общих атрибутов по иерархии типов (*pull up field* в терминологии работы [2]).

Схожие задачи могут решаться с использованием метода FCA (*Formal Concept Analysis*) [5], когда элементам множества классов оптимально назначаются наборы атрибутов. Однако в случае подхода из работы [4] атрибуты типов не являются элементами отдельного независимого множества, а сами относятся к иерархии типов. Это обстоятельство предоставляет более широкие возможности для адекватного моделирования и преобразования иерархий типов.

Настоящая работа развивает подход, предложенный в работе [4]. Она посвящена основам реализации инструментальной системы рефакторинга, использующей LP-структуры на решетке типов в качестве их математической модели.

Обсуждаются ограничения, связанные с возможностями алгебраической модели типов [4]. Предлагается и обосновывается обобщение LP-структуры на решетке типов, снимающее значительную часть ограничений на допустимые для моделирования иерархии типов. На основе новой модели описывается рефакторинг методом подъема общих атрибутов и оптимизации иерархии, применимый для широкого класса объектно-ориентированных систем.

Подходы, представленные в настоящей работе, не привязаны к конкретным языкам программирования и опираются лишь на базовые понятия методологии объектно-ориентированного программирования (ООП).

1. Основные понятия

Ниже кратко представлены некоторые необходимые для дальнейшего изложения определения и результаты из работы [4].

* Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-07-00037.

Под LP-структурой подразумевается алгебраическая система, представляющая собой математическую решетку $\mathbb{F}(\leqslant \wedge, \vee)$ с дополнительно заданным на ней бинарным отношением $\leftarrow$, на которое накладываются следующие ограничения:

- транзитивность: из $a \leftarrow b$ и $b \leftarrow c$ следует $a \leftarrow c$ ($a, b, c \in \mathbb{F}$);
- дистрибутивность, в случае данной работы — $\vee$-дистрибутивность: из $b_1 \leftarrow a$ и $b_2 \leftarrow a$ следует $b_1 \vee b_2 \leftarrow a$;
- включение тавтологий: $\leqslant \subseteq \leftarrow$.

Между парами типов (классов) в ООП существует как минимум два вида связей — наследование и агрегация.

В работе [4] множество типов и связи, соответствующие наследованию, моделирует решетка типов $\mathbb{F}$: если тип b является наследником a, то $b \leqslant a$. В результате рассмотрению подлежат лишь такие иерархии типов, для которых после введения указанного частичного порядка алгебраическая система $\mathbb{F}$ удовлетворяет определению решетки. Аналогичное ограничение действует и в настоящей работе.

Дополнительно на решетке $\mathbb{F}$ вводится бинарное отношение R, соответствующее агрегации: если тип b агрегирует объект типа a, то $(b, a) \in R$. Отметим, что $(\mathbb{F}, R)$ еще не является LP-структурой, поскольку при таком построении нельзя гарантировать, что R будет удовлетворять требованиям определения LP-структуры.

Оба введенных отношения ($\leqslant$ и R) в предметной области имеют общую семантику — один тип получает возможности другого типа в виде доступа к его атрибутам. Семантически ясно, что такое отношение "обладания набором возможностей" (далее обозначено как $\xleftarrow{R}$) обязано быть рефлексивным и транзитивным.

Понятие $\vee$-дистрибутивности отношения $\xleftarrow{R}$ задает семантику монотонного логического вывода. При ее наличии из $b_1 \xleftarrow{R} a$ и $b_2 \xleftarrow{R} a$ следует $b_1 \vee b_2 \xleftarrow{R} a$. Это означает, что если тип b_1 обладает возможностями a и b_2 обладает возможностями a, то и наименьший общий предок b_1 и b_2 также обладает возможностями типа a. Если поместить атрибут a в $b_1 \vee b_2$, то b_1 и b_2 получат его в порядке наследования, что будет соответствовать рефакторингу методом подъема общих атрибутов.

Однако в случае LP-структур на решетке типов выполнение $\vee$-дистрибутивности в ряде случаев нецелесообразно и должно зависеть от контекста. Иначе возможны ситуации, нежелательные с точки зрения качества кода [4]. Например, тип b будет обладать возможностями своего потомка a; тип b будет обладать возможностями a одновременно по нескольким линиям наследования и т. д. С учетом таких ситуаций логический вывод возможностей типов в LP-структуре теряет свою монотонность.

Для формализации и исследования немонотонного логического вывода на рассмотренных структурах были введены понятия $\vee$-совместимости пар (b_1, a), $(b_2, a) \in R$, $\vee$-дистрибутивности тройки (c_1, c_2, a), где $c_1, c_2, a \in \mathbb{F}$ и некоторые другие понятия, с которыми можно детально ознакомиться в работе [4].

Логическое замыкание отношения R на решетке $\mathbb{F}$ представляет собой отношение, содержащее все такие пары (a, b), что в типе b доступны возможности типа a (с учетом выполнения ограниченной $\vee$-дистрибутивности).

Логической редукцией отношения R на решетке $\mathbb{F}$ называется минимальное эквивалентное ему отношение R_0. Логическая редукция рассматриваемой LP-структуры соответствует иерархии типов с минимальным эквивалентным набором связей, что способствует минимизации дублирования кода.

В работе [4] доказаны теоремы о существовании логической редукции и логического замыкания для LP-структур на решетке типов, а также обоснованы способы их построения.

2. Постановка задачи

Несмотря на то что семантика элементов LP-структуры на решетке типов четко сформулирована, преобразование в нее множества классов реальной программной системы не является тривиальной задачей.

При практическом применении рассмотренной методики возникают сложности, обусловленные следующими обстоятельствами.

- Бинарное отношение на решетке типов не позволяет полноценно представлять иерархии типов, в которых одним типом агрегируются несколько экземпляров другого типа.
- Логическая редукция лишь гарантирует, что по ее результатам все "достижимые" по отношению $R \cup \leqslant$ типы останутся таковыми. Поэтому преобразование, которое в одних случаях может рассматриваться как допустимая оптимизация иерархии типов, в других случаях может привести к LP-структуре, описывающей неработоспособное множество типов.

Очевидно, что такого рода обстоятельства влекут чрезмерное ограничение либо набора допустимых программных систем, либо подмножества атрибутов и типов, которые могут быть подвергнуты рефакторингу.

В настоящей работе представлен способ распространения указанной методики на произвольные иерархии типов. Для достижения этой цели предлагается расширить понятие LP-структуры на решетке типов, а также адаптировать процесс рефакторинга к учету свойств конкретных типов и особенностей случаев агрегации.

Случаем агрегации в этом разделе и далее называется кортеж $(T_1, T_2, name, I)$, где T_1 — это тип, содержащий в качестве атрибута экземпляр типа T_2, $name$ соответствует идентификатору атрибута, а I представляет дополнительную информацию о данном атрибуте. Для краткости введем отображения $agg\_source(A)$ и $agg\_target(A)$, которые случаю агрегации A ставят в соответствие агрегирующий (T_1) и

агрегируемый (T_2) типы соответственно. Будем также обозначать множество всех случаев агрегации в рассматриваемой программной системе как T_{agg}.

Далее классифицируем типы и случаи агрегации в зависимости от наличия свойств, существенных для проведения автоматизированного рефакторинга на основе LP-структур.

Предлагается разбить множество типов на две указанные ниже категории. Это разбиение должно зависеть только от поведения самих классов и не зависеть от того, как они используются в подвергающемся рефакторингу программном продукте.

1. *Типы, допускающие совмещение.* Это типы, чьи экземпляры можно использовать совместно без дополнительных условий. В частности, в данную категорию входят типы, все допустимые операции над которыми идемпотентны.

2. *Прочие типы.* К ним, например, относятся все примитивные изменяемые типы данных (строка, число и т. д.). Сюда также будем относить и такие типы, для которых не удается достоверно установить принадлежность к первой категории.

Однако такое определение допускающих совмещение типов еще не дает всей нужной информации для того, чтобы определить допустимые операции при оптимизации иерархии типов. Категория допускающих совмещение типов описывает свойство, присущее экземплярам в отдельности, и не гарантирует, что такие экземпляры взаимозаменяемы.

Для описания практической взаимозаменяемости экземпляров типа потребуется дополнительно задать отношение эквивалентности $\overset{eq}{\sim}$ на экземплярах, допускающих совмещение типов. Если $a \overset{eq}{\sim} b$, то допустимо заменить в исходном коде программы все случаи обращения к a на обращение к b, и наоборот. Такое разбиение дает достаточное условие для оптимизации иерархии путем удаления избыточных случаев агрегации из каждого такого класса эквивалентности. От $\overset{eq}{\sim}$ логично потребовать, что $a \overset{eq}{\sim} b \Rightarrow agg\_target(a) = agg\_target(b)$.

В качестве примера, иллюстрирующего определенные выше разбиения, рассмотрим класс, отвечающий за представление даты/времени в виде строки. UML-диаграмма такого класса изображена на рис. 1.

Если реализация такого класса соответствует ожиданиям — операция toString() на одних и тех же входных данных выдает одинаковый результат, то данный класс относится к допускающим совмещение.

Допустим, тип A агрегирует экземпляр DateTimeFormatter. Пусть класс-предок A также агрегирует экземпляр DateTimeFormatter с тем же форматом и часовым поясом, что и A. Тогда при выполнении рефакторинга допустимо удалить экземпляр рассматриваемого класса из A, и на поведение программной системы это не повлияет.

Эти рассуждения по сути описывают правило формирования отношения $\overset{eq}{\sim}$: если a и b — экземпляры DateTimeFormatter, то $a \overset{eq}{\sim} b$ при равенстве значений формата и часового пояса, использованных при создании экземпляров a и b.

Для типа DateTimeFormatter можно провести разбиение экземпляров на непересекающиеся группы по равенству строк format и timezone, передаваемых в конструктор класса.

Рассмотрим второй вариант класса DateTimeFormatter. Соответствующая ему UML-диаграмма классов изображена на рис. 2.

В данном варианте добавлены функции setTimezone() и setFormat(), позволяющие изменить часовой пояс и описание требуемого формата строки. Замена множества изначально одинаковых экземпляров такого типа на один экземпляр может очевидным образом повлечь изменение поведения системы.

В дальнейшем потребуется также разбиение множества случаев агрегации. Оно полностью зависит от того, как случаи агрегации используются в коде программной системы. Цель такого разбиения — выделить случаи агрегации, которые можно использовать для доступа к экземплярам допускающих совмещение типов.

1. *Простые случаи агрегации.* Это такие случаи агрегации (T_1, T_2, *name*, *I*), которые с момента создания экземпляра T_1 постоянно связаны с одним и тем же экземпляром T_2 (атрибутом класса T_1).

2. *Прочие случаи агрегации.* В такую категорию приводят, например, атрибуты, которым после создания экземпляра T_1 переприсваивается значение. Другой пример — атрибуты, которые не инициализи-

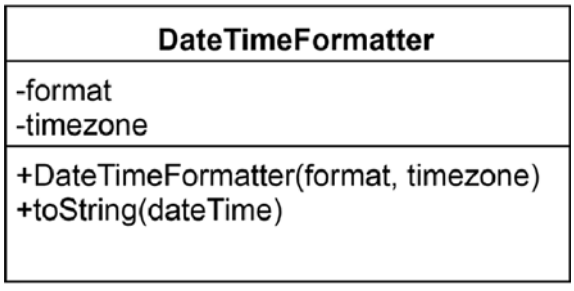

Рис. 1. UML-диаграмма класса DateTimeFormatter

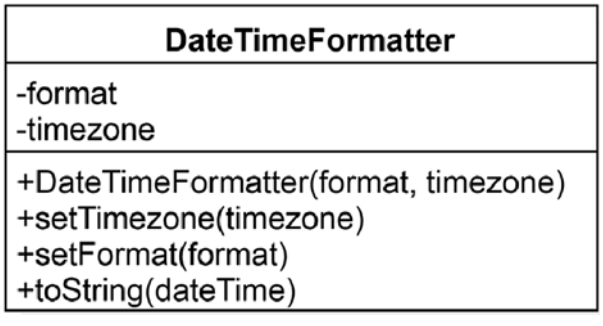

Рис. 2. UML-диаграмма альтернативного варианта класса DateTimeFormatter

руются при создании экземпляра T_1, либо допускают пустое значение (NULL).

Описанная классификация позволяет выделить достаточные условия для формального описания ситуаций, когда допустимо рассмотрение типов в рамках "обладания возможностями".

Пусть (b_i), $i = 1...n$ — последовательность случаев агрегации такая, что:

- $\forall i > 1$ выполняется $agg\_target(b_{i-1}) = T'$, где $T' = agg\_source(b_i)$ или $agg\_source(b_i)$ является типом-предком T';
- каждый из случаев агрегации b_i является простым;
- $C = agg\_target(b_n)$, причем тип C допускает совмещение.

Тогда будем констатировать, что тип $A = agg\_source(b_1)$ действительно *обладает возможностями* типа C. В исходном коде программы при наличии выбора из нескольких таких последовательностей (b_i) для доступа к экземпляру типа C можно воспользоваться любой из них (конечно, при условии, что заключительные элементы b_n таких последовательностей входят в один класс эквивалетности $\overset{eq}{\sim}$).

В соответствии с приведенной выше классификацией программная система сохранит свое поведение, если любые обращения к случаю агрегации $x \in T_{agg}$. При этом достаточно через экземпляр типа A обращаться к произвольному экземпляру C, следуя любой описанной выше цепочке случаев агрегации $\{B_i\}$.

Следует упомянуть еще одну проблемную ситуацию, связанную не с риском потери работоспособности программной системы, а с "читабельностью", полученной после рефакторинга иерархии типов.

Если при подъеме общих атрибутов ориентироваться только на тип переносимого атрибута, то возможны ситуации, когда в результате в класс-предок попадут атрибуты, имеющие различную семантику. UML-диаграмма классов для упрощенного примера такой ситуации изображена на рис. 3. Изображенные классы описывают передаваемые сообщения в некоторой многопользовательской системе. Сообщения могут поступать от одного пользователя другому

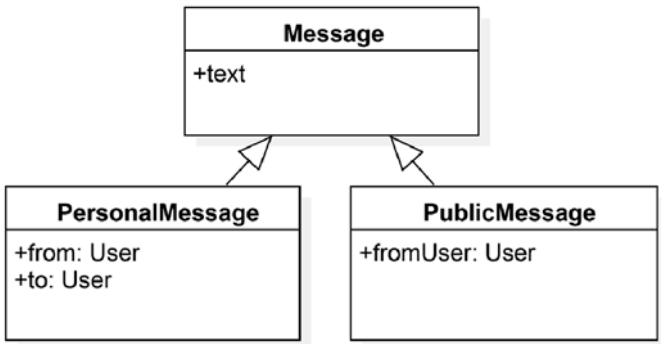

Рис. 3. UML-диаграмма классов, иллюстрирующая иерархию типов, для которой возможно осуществление подъема общих атрибутов

(PersonalMessage) и быть публичными, имеющими только автора (PublicMessage).

В случае, когда применением метода подъема общих атрибутов поля to из PersonalMessage и fromUser из PublicMessage будут перемещены в класс Message в виде одного общего поля, работоспособность программы сохранится. Однако такое изменение сделает код программы менее понятным для разработчика.

В настоящей работе такие ситуации будут учитываться с помощью вводимого отношения эквивалентности $\overset{agg}{\sim}$ на множестве случаев агрегации. Если $a \overset{agg}{\sim} b$ — случаи агрегации, входящие в один класс эквивалентности, то поднятие соответствующих общих атрибутов в тип-предок допустимо. Определение такого отношения эквивалентности оказывает влияние только на качество полученного в результате рефакторинга кода, но не на его работоспособность. Очевидно также следующее ограничение на указанное отношение эквивалентности: все случаи агрегации одного класса эквивалентности должны относиться к одному и тому же типу (см. разбиение выше).

При проведении рефакторинга потребуется перестроить исходный код программы, содержащий описание типов, а также заменить в коде все обращения к атрибутам, существовавшим ранее, на обращения к атрибутам новой иерархии типов. В случае, когда кроме подъема общих атрибутов проводится дополнительная оптимизация иерархии типов, возможны ситуации, в которых обращение к атрибуту заменится не одним, а цепочкой таких обращений в новой иерархии.

Опираясь на представленное выше разбиение случаев агрегации и типов, можно сформулировать следующую постановку задачи. Требуется описать формальный метод рефакторинга, на выходе предоставляющий пару $\left(T'_{agg}, f\right)$, где T'_{agg} — множество случаев агрегации в результирующей иерархии типов, а $f: T_{agg} \to (t'_n)$, $t'_i \in T'_{agg}$ — отображение, ставящее в соответствие каждому случаю агрегации исходной иерархии типов последовательность случаев агрегации новой иерархии типов (t'_n). Такая последовательность может состоять как из одного элемента (случай агрегации не перемещался или был перемещен в тип-предок), так и из нескольких (произошла оптимизация множества экземпляров, допускающих совмещение типов). T'_{agg} и f должны удовлетворять следующим требованиям.

1. Если $f(a) = f(b)$, то для случаев агрегации $a, b \in T_{agg}$ выполняется $agg\_target(a) = agg\_target(b)$. При этом a и b являются простыми случаями агрегации либо оба не являются таковыми.

2. Если $f(a) = f(b)$ и $t = agg\_target(a) = agg\_target(b)$ является типом, допускающим совмещение, то должно выполняться $a \overset{eq}{\sim} b$.

3. Если $f(a) = f(b)$ и $t = agg\_target(a) = agg\_target(b)$ не является типом, допускающим совмещение, или

a и b не являются простыми случаями агрегации, то должно выполняться $a \overset{agg}{\sim} b$.

4. Если $a \in T_{agg}$ не входил в категорию простых или класс $agg\_target(a)$ не являлся допускающим совмещение, то $|f(a)| = 1$. Это ситуация, когда оптимизация множества таких случаев агрегации недопустима.

5. Пусть $f(a) = (b_1,..., b_n)$. Тогда должно выполняться:

a) $agg\_source(a) = agg\_source(b_1)$, либо $agg\_source(b_1)$ является типом-предком $agg\_source(a)$;

b) $agg\_target(a) = agg\_target(b_n)$;

c) $\forall i > 1$: $agg\_target(b_{i-1}) = agg\_source(b_i)$, либо $agg\_source(b_i)$ является типом-предком $agg\_target(b_{i-1})$;

d) в случае, если $a \in T_{agg}$ является простым, а $agg\_target(a)$ — допускающим совмещение типом, должно также выполняться следующее: $\forall b_i$ если $f(t_{agg}) = Y$, где $|Y| = 1$ и $Y_1 = b_i$, то случай агрегации $t_{agg} \in T_{agg}$ является простым.

При решении задачи дополнительно к сформулированным выше обязательным условиям будут преследоваться следующие цели:

- увеличение $|T_{agg}| - |T'_{agg}|$, что соответствует задаче оптимизации иерархии типов;
- снижение избыточности результирующего кода.

3. Обобщение LP-структуры на решетке типов

Примем следующие обозначения: T — множество типов рассматриваемой системы; T_{agg} — как и прежде, множество всех случаев агрегации для рассматриваемой системы.

Требуется дополнить определение LP-структуры на решетке типов для достижения следующих целей:

- возможность моделирования агрегации одним типом нескольких экземпляров другого типа;
- возможность задания ограничений на подъем в общий тип-предок конкретных случаев агрегации.

Как уже упоминалось ранее, важным преимуществом подхода, предложенного в работе [4], является то, что агрегация рассматривается как дополнительная связь между элементами решетки типов, а не между двумя независимыми множествами. Основная цель расширения понятия LP-структуры — допустить моделирование ситуаций, когда один тип агрегирует несколько экземпляров другого типа, и при этом сохранить отмеченную особенность. Для ее достижения предлагается внести в иерархию дополнительные "искусственные" типы. Они будут представлять случаи агрегации (точнее, их множества), и при этом окажутся связанными с элементами модели, которые соответствуют агрегируемым типам данных.

Далее рассматривается формирование пары $(\mathbb{F}, R)$, используемой в работе [4] для построения LP-структуры на решетке типов. LP-структуру, построенную с использованием таких $(\mathbb{F}, R)$, далее будем называть *расширенной*.

Для формирования $(\mathbb{F}, R)$ потребуются:

- определенные ранее множества T и T_{agg};
- множество искусственных типов $\tilde{T}$;
- отображение случаев агрегации на искусственные типы, которые будут "представлять" эти случаи агрегации в расширенной LP-структуре $f_{agg}: T_{agg} \to \tilde{T}$.

Для того чтобы при рассматриваемых изменениях связи между существующими типами сохранились, дополнительно потребуем от f_{agg} следующее: $f_{agg}\left(t^{(1)}_{agg}\right) = f_{agg}\left(t^{(2)}_{agg}\right) \Rightarrow \text{agg\_target}\left(t^{(1)}_{agg}\right) = agg\_target\left(t^{(2)}_{agg}\right)$.

Это требование также позволит определить еще одно вспомогательное отображение $f_{orig}: \tilde{T} \to T$ следующим образом: если $f_{orig}(\tilde{t}) = t$, то существует $f_{agg}(t_{agg}) = \tilde{t}$ и $t = agg\_target(t_{agg})$. То есть $t = f_{orig}(\tilde{t})$ — это тип, вместо которого в отношении агрегации будет участвовать $\tilde{t}$.

Предлагается следующий способ построения $(\mathbb{F}, R)$.

1. Формируется решетка типов T, соответствующая отношению наследования, аналогично описанному в работе [4].

2. В решетку добавляются элементы $\tilde{T}$, причем отображение f_{orig} определяет отношение частичного порядка. Для добавленных $\tilde{t} \in \tilde{T}$ справедливо $\tilde{t} \leqslant f_{orig}(\tilde{t})$.

3. Бинарное отношение агрегации R строится следующим образом: для каждого случая агрегации $t_{agg} \in T_{agg}$ в R добавляется пара $(x, \tilde{t})$, где $\tilde{t} = f_{agg}(t_{agg})$, а $x = agg\_source(t_{agg})$.

Перечислим несколько свойств построенной алгебраической системы.

- Сохраняются связи исходной иерархии типов: для каждого $t_{agg} \in T_{agg}$, $a = agg\_source(t_{agg})$, $b = agg\_source(t_{agg})$ в LP-структуре сохранится связь $a \xleftarrow{R} b$ в силу $(a, \tilde{t}) \in R$ и $\tilde{t} \leqslant b$.
- Появляется возможность моделирования агрегации одним типом нескольких экземпляров другого типа. Для этого достаточно обеспечить, чтобы для таких $t^{(1)}_{agg}, t^{(2)}_{agg} \in T_{agg}$, что $agg\_target\left(t^{(1)}_{agg}\right) = agg\_target\left(t^{(2)}_{agg}\right)$ и $agg\_source\left(t^{(1)}_{agg}\right) = agg\_source\left(t^{(2)}_{agg}\right)$, выполнялось $f_{agg}\left(t^{(1)}_{agg}\right) \neq f_{agg}\left(t^{(2)}_{agg}\right)$.
- Появляется возможность контролировать подъем атрибутов. Условия $f_{agg}\left(t^{(1)}_{agg}\right) \neq f_{agg}\left(t^{(2)}_{agg}\right)$ достаточно, чтобы избежать подъема случаев агрегации $t^{(2)}_{agg}$ и $t^{(1)}_{agg}$ в общий тип-предок при проведении логической редукции [4].

4. Изоморфизм между элементами расширенной LP-структуры и иерархией типов системы

Для использования расширенной LP-структуры в целях рефакторинга необходимо научиться находить изоморфизм между элементами иерархии типов

исходной системы и элементами используемой математической модели.

Предварительно разобьем множество случаев агрегации на непересекающиеся группы. Такое разбиение преследует следующие цели.

- Объединение всех проведенных ранее классификаций в один критерий, определяющий необходимое условие для возможности поднятия в общий предок двух произвольных атрибутов.
- Устранение неоднозначностей, все еще остающихся после имеющихся классификаций. Например, не исключаются ситуации, когда a, $b \in T_{agg}$ принадлежат одному классу эквивалентности и $agg\_source(a)$ является потомком $agg\_source(b)$. Если не разрешить неоднозначность до проведения рефакторинга, проблемная ситуация возникнет при попытке провести соответствие между случаями агрегации исходной иерархии типов и случаями агрегации результирующей иерархии.

Важным исключением из этой ситуации являются случаи агрегации, являющиеся простыми и для которых агрегируемый тип допускает совмещение. Такие случаи агрегации могут рассматриваться в контексте "обладания возможностями", и для них разрешение неоднозначных ситуаций не вызывает затруднений — можно выбрать любой допустимый экземпляр агрегируемого типа.

Обозначим подмножество таких случаев агрегации $T_{comb} \subseteq T_{agg}$. Для T_{comb} искомое разбиение на группы, внутри которых допускается совмещение, считается уже заданным — в постановке задачи для этой цели использовалось отношение эквивалетности $\overset{eq}{\sim}$.

Для всех прочих случаев сформулируем следующие требования к разбиению T_{agg} на группы.

1. Если a, $b \in T_{agg} \backslash T_{comb}$ и a и b входят в разные классы эквивалентности $\overset{agg}{\sim}$, то a и b должны принадлежать разным группам.

2. Если a, $b \in T_{agg} \backslash T_{comb}$ и a является простым случаем агрегации (по классификации из постановки задачи), а b — не является, то a и b должны принадлежать разным группам. Такое требование вводится для сохранения классификации случаев агрегации на простые и прочие, проводимой для исходной иерархии типов.

3. Пусть $t \in T$, а $T^t_{agg} \subseteq T_{agg} \backslash T_{comb}$ — множество случаев агрегации, таких, что $\forall t_{agg} \in T^t_{agg}$: либо $agg\_source(t_{agg}) = t$, либо $agg\_source(t_{agg})$ является типом-предком t. Тогда все элементы T^t_{agg} должны принадлежать разным группам.

Первые два условия требуются для ограничения случаев агрегации, которые могут быть соединены в классе-предке при подъеме атрибутов. Требование 3 устраняет неоднозначность при рассмотрении иерархии типов.

Требования 1 и 2 используют уже имеющиеся разбиения множества случаев агрегации, и в силу этого обстоятельства добиться разбиения только по этим требованиям можно достаточно просто. Для этого нужно разбить на группы множество $T_{agg} \backslash T_{comb}$ по принадлежности классам эквивалентности, а затем каждую из полученных групп разбить по принадлежности к категории простых случаев агрегации.

После проведения разбиения $T_{agg} \backslash T_{comb}$ с применением требований 1 и 2 следует каждую полученную группу разбить по требованию 3. Однако разбиение по этому требованию не является тривиальной задачей.

С целью минимально ограничить подъем общих атрибутов такое разбиение должно выделять как можно меньшее число групп. В таком виде задача разбиения группы случаев агрегации по требованию 3 фактически приобретает вид задачи об оптимальной раскраске графа.

Напомним, правильной k-раскраской графа $G = (V, E)$ является отображение $c: V \to \{1, 2, ..., k\}$, для которого верно $\{u, v\} \in E \Rightarrow c(u) \neq c(v)$. Для заданного графа наименьшее k, при котором такая раскраска возможна, называется его хроматическим числом и обозначается $\chi(G)$. Оптимальная раскраска графа — такая правильная раскраска, для которой требуется ровно $\chi(G)$ цветов [6].

Пусть $\bar{T}_{agg} \subseteq T_{agg}$ — подмножество случаев агрегации, которые подлежат разбиению в соответствии с требованием 3. Построим граф $G = (V, E)$, для которого решение задачи об оптимальной раскраске будет соответствовать нахождению искомого разбиения на минимальное число групп.

Заполним множество вершин V элементами, изоморфными элементам $\bar{T}_{agg}$, и следующим образом обозначим биекцию между элементами этих множеств: $w : \bar{T}_{agg} \leftrightarrow V$. Множество ребер E стоится путем выполнения следующего алгоритма для каждого типа, имеющегося в иерархии $t \in T$.

1. Выделяется $T^t_{agg} \subseteq \bar{T}_{agg}$, где $\forall t_{agg} \in T^t_{agg}$: либо $agg\_source(t_{agg}) = t$, либо $agg\_source(t_{agg})$ является типом-предком t.

2. Для каждой пары вершин $a, b \in T^t_{agg}$ в E добавляется ребро между вершинами $w(a)$ и $w(b)$.

Решением задачи об оптимальной раскраске графа является отображение $c: V \to \{1, 2, ..., k\}$. Если разбить $\bar{T}_{agg}$ на группы элементов t_{agg}, имеющие одинаковое значение $c(w(t_{agg}))$, то такое разбиение и будет искомым в силу того, что при построении графа, ребрами были соединены все пары вершин, которые должны были оказаться в разных группах.

Алгоритм, трансформирующий задачу поиска разбиения $\bar{T}_{agg}$ к задаче об оптимальной раскраске графа работает, очевидно, за полиномиальное время. Однако сама задача оптимальной раскраски графа является NP-полной [6].

Если проводить разбиение случаев агрегации по требованию 3 после разбиения по требованиям 1 и 2, то на практике формируемые для раскраски графы могут иметь малое число вершин. Тогда экспонен-

циальная временная сложность алгоритма не будет создавать трудностей. В прочих случаях потребуется использование приближенных решений, описанных, например, в работе [6].

После получения разбиения случаев агрегации на группы $T_{groups} = \left\{T_{agg}^{(i)}\right\}$, соответствующие указанным трем требованиям, оказывается возможным построение LP-структуры на решетке типов. Процесс формирования $(\mathbb{F}, R)$ описан в разд. 3. Потребуются следующие входные данные: T, T_{agg}, множество $\tilde{T}$ и отображение f_{agg}. Ниже перечислены этапы построения LP-структуры.

Множества T и T_{agg} определены ранее.

Множество $\tilde{T}$ заполняется элементами, изоморфными множеству выделенных групп T_{groups}. Тогда между $\tilde{T}$ и T_{groups} существует биекция: $g : T_{groups} \leftrightarrow \tilde{T}$.

Отображение $f_{agg} : T_{agg} \to \tilde{T}$ задается описанным далее способом. Для каждого случая агрегации $t_{agg} \in T_{agg}$ пусть $T_{agg}^{(i)}$ — группа, в которую входит t_{agg}, тогда $f_{agg}\left(t_{agg}\right) = g\left(T_{agg}^{(i)}\right)$.

В силу предъявляемых к T_{groups} требований, для каждой выделенной группы $T_{agg}^{(i)}$ справедливо:

$$t_{agg}^{(1)}, t_{agg}^{(2)} \in T_{agg}^{(i)} \Rightarrow agg\_target\left(t_{agg}^{(1)}\right) = agg\_target\left(t_{agg}^{(2)}\right).$$

Следовательно, f_{agg} удовлетворяет требованию, накладываемому процедурой, описанной в разд. 3.

Таким образом, все необходимые параметры заданы, и далее можно выполнить шаги построения $(\mathbb{F}, R)$ из разд. 3.

5. Осуществление подъема общих атрибутов

Данный раздел посвящен непосредственному проведению рефакторинга иерархии типов. Пусть имеются определенные в постановке задачи классификация типов, классификация случаев агрегации, а также алгебраическая система $(\mathbb{F}, R)$. Они получены в результате выполнения действий, описанных в разд. 4. Требуется сформировать множество случаев агрегации T'_{agg} для результирующей иерархии типов и отображение $f: T'_{agg} \to \{t'_i\}$, $t'_i \in T'_{agg}$. Требования к ним были определены в разд. 2.

Несмотря на то что пара $(\mathbb{F}, R)$ уже определена, воспользоваться для решения задачи напрямую подходом из работы [4] не дает следующая особенность логической редукции. Подразумеваемая ею оптимизация иерархии типов опирается на "обладание возможностями", что в соответствии с принятой в настоящей работе классификацией допустимо только для ограниченного набора случаев агрегации и типов.

Для того чтобы рефакторинг смог затронуть иерархию типов целиком, предлагается проводить данный процесс в два этапа.

1. Этап, касающийся иерархии типов целиком: поднятие общих атрибутов.

2. Этап, касающийся только части иерархии типов: проведение логической редукции.

Для осуществления первого этапа удобно воспользоваться концепцией неконфликтных $\vee$-дистрибутивных троек. В работе [4] одним из принципов формирования множества таких $\vee$-дистрибутивных троек является избегание ситуаций наследования одним типом другого типа сразу по нескольким линиям. С одной стороны, эта стратегия положительно влияет на качество кода, который будет получен в результате такого рефакторинга, с другой стороны, позволит избежать неоднозначностей при проведении отображения между прежней и новой иерархиями типов.

Ниже приведена последовательность шагов, реализующих указанные этапы.

Шаг 1. $(\mathbb{F}, R)$ сформирована в результате выполнения действий из разд. 4. Далее потребуются отображения f_{agg} и f_{orig}, связанные с таким построением расширенной LP-структуры.

Шаг 2. Изначально $T'_{agg} = \varnothing$.

Шаг 3. Построим $\mathbb{T}$ — множество неконфликтных $\vee$-дистрибутивных троек для $(\mathbb{F}, R)$, как это описано в работе [4].

Шаг 4. Для каждой тройки $(c_1, c_2, a) \in \mathbb{T}$, такой, что $f_{orig}(a)$ не является типом, допускающим совмещение, или существует $t_{agg} \in T_{agg}$, что $f_{agg}(t_{agg}) = a$ и случай агрегации t_{agg} не относится к простым, выполним следующие действия.

4.1. Добавим в t'_{agg} случай агрегации $t'_{agg} = (T_1, T_2, name, I)$, для которого $T_1 = c_1 \vee c_2$, $T_2 = f_{orig}(a)$, а $name$ и I определяются на основе множества таких t_{agg}, что $f_{agg}(t_{agg}) = a$, способ определения $name$ и I выходит за рамки настоящей работы и должен быть рассмотрен отдельно;

4.2. Сохраним соответствие между рассматриваемой $\vee$-дистрибутивной тройкой и добавленным t'_{agg} в биекции $\mathbb{T}_{added}$: $\mathbb{T} \leftrightarrow T'_{agg}$.

Шаг 5. Для каждого $t_{agg} \in T_{agg}$, такого, что случай агрегации t_{agg} не является простым или $agg\_target(t_{agg})$ не является типом, допускающим совмещение:

5.1. Если существует $x = (c_1, c_2, a) \in \mathbb{T}$, где $a = f_{agg}(t_{agg})$ и $agg\_source(t_{agg})$ является потомком типа $c_1 \vee c_2$, то сохраним $f(t_{agg}) = t'_{seq}$, где t'_{seq} — последовательность, единственным элементом которой является $\mathbb{T}_{added}(x)$.

5.2. Иначе добавим t_{agg} в T'_{agg} и сохраним $f(t_{agg}) = t'_{seq}$, где t'_{seq} — последовательность, единственным элементом которой является t_{agg}.

Шаг 6. Удалим из R все пары $(a, b) \in R$, такие, что существует $t_{agg} \in T_{agg}$, для которого $f_{agg}(t_{agg}) = b$ и случай агрегации t_{agg} не относится к простым.

Шаг 7. Построим LP-структуру на основе $(\mathbb{F}, R)$ и проведем для нее логическую редукцию в соответствии с работой [4], используемое при этом множество неконфликтных $\vee$-дистрибутивных троек должно являться подмножеством $\mathbb{T}$. С учетом шага 6 такая LP-структура моделирует иерархию типов только с простыми случаями агрегации.

Шаг 8. Пусть R'— бинарное отношение LP-структуры после проведения логической редукции. Для каждого $(a, b) \in R'$, где $f_{orig}(b)$ является типом, допускающим совмещение, выполним следующее. Добавим в T'_{agg} случай агрегации $t'_{agg} = (T_1, T_2, name, I)$, для которого $T_1 = a$, $T_2 = f_{orig}(b)$. Способ определения *name* и I выходит за рамки данной работы и должен быть рассмотрен отдельно.

Шаг 9. Для каждого $t_{agg} \in T_{agg}$ такого, что случай агрегации t_{agg} является простым и $agg\_target(t_{agg})$ является типом, допускающим совмещение, выполним следующие действия.

9.1. Сформируем последовательность $(a_1, b_1), ..., (a_n, b_n) \in R'$, для которой $agg\_source(t_{agg}) \leqslant a_1$, $b_n = f_{agg}(t_{agg})$, и $\forall i > 1$: $b_{i-1} \leqslant a_i$;

9.2. Каждому (a_i, b_i) из такой последовательности поставим в соответствие элемент $t'_{agg} \in T'_{agg}$, такой, что $agg\_target(t'_{agg}) = f(b_i)$, $agg\_source(t'_{agg})$ равен или является предком a_i, а также для всех таких t_{agg}, что $f(t_{agg}) = t'_{agg}$, t_{agg} является простым случаем агрегации.

9.3. Пусть $t'_{seq} = \{t'_1, t'_2, ..., t'_n\}$ — последовательность элементов T'_{agg}, для которых t'_i поставлен в соответствие (a_i, b_i) на шаге 2. Тогда положим $f(t_{agg}) = t'_{seq}$.

Приведем некоторые комментарии к описанной выше процедуре.

- Шаг 9.1 всегда возможен в силу того, что для описанного $t_{agg} \in T_{agg}$ должно существовать $(agg\_source(t_{agg}), f_{agg}(t_{agg})) \in R$, а проведение логической редукции гарантирует, что тип продолжит "обладать возможностями" тех типов, чьими возможностями он обладал до проведения логической редукции.
- Шаг 9.2 также всегда возможен по двум причинам.
  - До шага 9 в T'_{agg} были добавлены элементы для каждого $(a, b) \in R'$, часть — в шаге 8, часть (простые случаи агрегации типов, не допускающих совмещение) — в шаге 4.
  - В силу того, что множество неконфликтных $\vee$-дистрибутивных троек, используемое при проведении логической редукции LP-структуры, входит в множество, использованное в шагах 4 и 5, простые случаи агрегации не могли в ходе логической редукции быть подняты по иерархии типов выше, чем это было уже учтено в шагах 4 и 5.

Построенные таким образом T'_{agg} и f удовлетворяют требованиям, сформулированным при формальной постановке задачи.

- Требования 1—3 удовлетворяются автоматически в силу свойств разбиения T_{agg} на группы, описанного в разд. 4.
- Для всех случаев агрегации, не являющихся простыми, или же в которых агрегируемый тип не является допускающим совмещение, $|f(t_{agg})| = 1$ в силу шага 5. Следовательно, f удовлетворяет требованию 4.
- Требования 5.a—5.c выполняются в силу шагов 5.1 и 9.2.
- Требование 5.d выполняется в силу того, что при построении $f(t_{agg})$ для таких t_{agg}, что $agg\_target(t_{agg})$ допускают совмещение и t_{agg} является простым случаем агрегации, в шагах 9.1, 9.2 используется LP-структура, в которой нет связей, соответствующих случаям агрегации, не являющихся простыми.

Заключение

Предложено обобщение LP-структуры на решетке типов. На основе новой модели описан формализованный процесс рефакторинга ООП-систем методом подъема общих атрибутов.

Описанный подход может быть положен в основу автоматизированной инструментальной системы рефакторинга. Представленные результаты гарантируют корректность результирующей иерархии типов при отсутствии ошибок в предварительно проведенной классификации типов и случаев агрегации.

Для реализации указанной автоматизированной системы также потребуется решение следующих задач.

- Считывание иерархии типов из исходного кода программы.
- Выбор способа формирования имен для совмещенных в общем предке атрибутов.
- Реализация разбиения случаев агрегации и типов в соответствии с описанной в настоящей статье классификацией.
- Обновление определений типов в исходном коде.
- Замены обращений к атрибутам в исходном коде.

Поскольку перечисленные задачи могут существенно зависеть от конкретного языка программирования, их решение в данном общем исследовании не рассматривалось. Оно планируется в качестве последующего развития представленной тематики.

Перспективным представляется также распространение изложенного подхода на реализацию других методов рефакторинга, в частности, метод "совмещения атрибутов", впервые опубликованный в работе [7].

Список литературы

1. **Nexhati A.** Justification of Software Maintenance Costs // International Journal of Advanced Research in Computer Science and Software Engineering. 2017. Vol. 7, Issue 3. P. 15—23.
2. **Фаулер М.** Рефакторинг: улучшение существующего кода / Пер. с англ. СПб.: Символ-Плюс, 2003. 432 с.
3. **Mens T., Tourwe T.** A survey of software refactoring // Software Engineering IEEE Transactions. 2005. Vol. 30. P. 126—139.
4. **Махортов С. Д.** LP-структуры на решетках типов и некоторые задачи рефакторинга // Программирование. 2009. Т. 35, № 4. С. 5—14.
5. **Godin R., Valtchev P.** Formal Concept Analysis-Based Class Hierarchy Design in Object-Oriented Software Development// Formal Concept Analysis / eds. B. Ganter, G. Stumme, R. Wille // Lecture Notes In Computer Science. Springer Berlin/Heidelberg. 2005. Vol. 3626. P. 304—323.
6. **Kosowski A., Manuszewski K.** Classical coloring of graphs // Contemporary Mathematics. 2004. Vol. 352. P. 1—20.
7. **Махортов С. Д.** LP-структуры для обоснования и автоматизации рефакторинга в объектно-ориентированном программировании //Программная инженерия. 2010. № 2. С. 15—21.

Application of LP-Structures to Object-Oriented Code Refactoring

S. D. Makhortov, msd_exp@outlook.com, **A. A. Nogikh,** a.nogikh@yandex.ru, Voronezh State University, Voronezh, 394018, Russian Federation

Corresponding author:

Makhortov Sergey D., Head of Department of Applied and System Software, Voronezh State University, Voronezh, 394018, Russian Federation,
E-mail: msd_exp@outlook.com

Received on March 04, 2019
Accepted on March 20, 2019

The article describes an approach of automatized refactoring that adopts lattice-based algebraic structures as a model for type hierarchy presentation and optimization. A distinctive feature of the adopted algebraic structures is their ability to aggregate models not as a relation between two independent sets, but as a means of access to the services provided by the aggregated objects. This property makes it possible to perform a much deeper optimization of type hierarchy. The described refactoring approach focuses on redundant attributes removal and on the relocation of identical attributes into their common superclasses ("Pull Up Field" technique). The crucial requirement for these transformations is that such modifications must not change the external behavior of the software. Also it is important that such modifications do not increase the complexity of the source code e.g. by combining attributes with differing semantics. The proposed approach fulfills these requirements by defining a comprehensive classification of types and attributes. Such classification enables it to take into consideration the properties of individual types and the compatibility of individual attributes. An algorithm has been described that, given a type hierarchy with an already performed classification, returns a new type hierarchy alongside with a mapping between attributes of these hierarchies. The proposed approach employs only the most fundamental ideas of the Object-Oriented programming and does not depend on any specific programming language.

Keywords: *refactoring, object-oriented programming, type hierarchy, LP-structures, automatization, software development tools*

Acknowledgements: *This work was funded by RFBR according to the research project 19-07-00037*

For citation:

Makhortov S. D., Nogikh A. A. Application of LP-Structures to Object-Oriented Code Refactoring, *Programmnaya Ingeneria*, 2019, vol. 10, no. 5, pp. 195—203

DOI: 10.17587/prin.10.195-203

References

1. **Nexhati A.** Justification of Software Maintenance Costs, *International Journal of Advanced Research in Computer Science and Software Engineering*, 2017, vol. 7, issue 3, pp. 15—23.

2. **Fowler M.** *Refactoring: Improving the Design of Existing Code.* Boston, MA, USA: Addison-Wesley, 1999.

3. **Mens T.** A survey of software refactoring, *IEEE Transactions on Software Engineering*, 2005, vol. 30, issue 3, pp. 126—139.

4. **Makhortov S.** LP structures on type lattices and some refactoring problems, *Programming and Computer Software*, 2009, vol. 35, no. 4, pp. 183—189. DOI: 10.1134/S0361768809040021.

5. **Godin R., Valtchev P.** Formal Concept Analysis-Based Class Hierarchy Design in Object-Oriented Software Development. Eds., B. Ganter, G. Stumme, R. Wille. *Formal Concept Analysis. Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2005, vol. 3626, pp. 304—323.

6. **Kosowski A., Manuszewski M.** Classical Coloring of Graphs, *Graph Colorings, AMS Contemporary Math*, 2004, vol. 352, pp. 1—20.

7. **Makhortov S.** LP-Structures for Justification and Automation of Refactoring in Object-Oriented Programming, *Programmnaya Ingeneria*, 2010, no. 2, pp. 15—21 (in Russian).

УДК 519.724.6 DOI: 10.17587/prin.10.204-218

И. Б. Казаков, аспирант, e-maill: i_b_kazakov@mail.ru, Московский государственный университет имени М. В. Ломоносова

Разностный код и протокол циклической поблочной передачи в скрытом канале по памяти

Исследован скрытый канал передачи данных, который порождается действиями организатора канала, позволяющими изменять объем памяти, обеспечивающий передачу. В канале передачи возможны ошибки, приводящие к склеиванию нескольких значений в одно. Исправление ошибок осуществляется путем многократной передачи одних и тех же данных. Предложена стратегия, минимизирующая число переданных блоков при заданном уровне надежности, а также оценены характеристики надежности канала. Адекватность модели проверена с помощью эксперимента, проведенного с использованием программной реализации канала.

Ключевые слова: *скрытые каналы, пропускная способность, разностный код, многократная передача данных, вероятность ошибки, виртуальная память*

Введение

Скрытым каналом передачи данных (далее для краткости изложения — скрытый канал) называется способ пересылки информации между злоумышленниками незаметно от законных, т. е. регламентированных принятой политикой безопасности, пользователей. В настоящей работе рассматривается скрытый канал по памяти, а именно канал передачи данных, который порождается действиями злоумышленника, позволяющими изменять объем памяти и обеспечивающими эту передачу. О других скрытых каналах можно узнать из обзора [1], где и было впервые введено это понятие. Суть рассматриваемого далее канала заключается в следующем: через диспетчер задач видны характеристики запущенных процессов. Процесс (далее — процесс-передатчик), очевидно, может манипулировать своими характеристиками, такими как степень загрузки центрального процессора и объем занимаемой памяти. Отличный от него процесс (далее — процесс-приемник) может читать данные, доступные диспетчеру задач, и, следовательно, считывать параметры первого процесса. Таким образом, процесс-передатчик может передавать процессу-приемнику информацию посредством манипуляции собственными параметрами.

В настоящей работе рассматриваются манипуляции объемом памяти, обеспечивающие передачу данных. Манипуляции такого рода происходят путем вызова в процессе-передатчике системных процедур занятия/освобождения памяти. Процесс-приемник периодически считывает объем памяти, занимаемый процессом-передатчиком. Этот объем представляет собой (с дальнейшими оговорками) сообщение, передаваемое по скрытому каналу.

Предварим изложение небольшой исторической справкой. Понятие скрытого канала уже представлено выше. Кратко перечислим и отметим особенности скрытых каналов. В уже упомянутом обзоре [1] с достаточной степенью подробности описаны примеры скрытых каналов. К их числу относятся модели передачи данных через содержание памяти; через чтение и запись во временные файлы; через флаги прав доступа; через сообщения между процессами; через манипуляции скоростью ввода/вывода. Следует выделить один весьма специфический и в настоящее время мало используемый канал передачи данных через суммы счетов оплаты.

Что касается первых трех каналов, а также последнего из перечисленных, то можно констатировать, что они устроены простейшим из возможных способов. Его суть в том, что имеются передатчик и приемник, а также некий реальный или виртуальный носитель информации, на который передатчик записывает информацию, а приемник ее считывает. Каналы такого рода будем называть "статическими", потому что сама по себе взятая среда передачи всегда находится в каком-то определенном состоянии. Это состояние, конечно же, в ходе передачи информации изменяется, но оно изменяется не само по себе, а по команде приемника.

От отмеченной группы скрытых каналов по природе среды передачи отличается канал передачи данных между процессами: он подобен обычной передаче информации как некоторого потока. В противоположность "статическим" каналам, в контексте настоящего исследования предлагается называть такие каналы "динамическими". Перечень скрытых каналов не ограничивается данными. Со способами организации других, в том числе связанных с сетью скрытых каналов можно ознакомиться в работе [2]. Очевидно, что стеганография (т. е. включение некоего скрытого сообщения в основное, передаваемое между законными пользователями) является одним

из способов организации скрытого канала. По технике организации таких каналов написана большая монография [3], посвященная также и цифровым водяным знакам.

В полном объеме перечислять все известные на настоящее время способы организации скрытого канала в одной статье не представляется возможным. Однако для четкого понимания цели исследования необходимо упомянуть о двух конкретных способах.

Во-первых, это канал передачи по загруженности процессора. Особенность данного канала состоит в том, что передатчик периодически нагружает процессор, а приемник считывает степень загруженности процессора. В простейшем варианте речь идет о полной загрузке процессора. Канал, рассматриваемый в настоящей работе, является его модификацией: используется тот же принцип обмена информацией между процессами посредством данных, предоставляемых диспетчером задач. Однако реализовать такой канал затруднительно, так как сложно манипулировать временем загрузки, а еще труднее заставить центральный процессор (ЦП) быть загруженным на некое точно определенное время. Поэтому в контексте исследования, результаты которого представлены в настоящей работе, было принято решение сменить параметр загруженности ЦП на объем выделенной виртуальной памяти.

Во-вторых, следует упомянуть скрытый канал перестановки пакетов и посвященные ему работы автора [4, 5]. В этих работах исследуется построение кода, исправляющего ошибки, т.е. вопрос о передаче информации с наибольшей скоростью в условиях, когда в скрытом канале существуют помехи. Согласно информации, полученной автором по результатам библиографических исследований, ранее изучался вопрос только о способности скрытых каналов передавать информацию вообще, т. е. о возможности передать 1 бит информации, но не вопрос максимизации пропускной способности. Таким образом, работы [4, 5] представляют результаты поисковых исследований, принадлежащие циклу работ, целью которого является восполнение данного пробела, а именно переход от "качественного" состояния теории скрытых каналов к "количественному".

Настоящая статья принадлежит циклу исследований на последнем из обозначенных направлений. Однако несмотря на общность условий, заключающуюся в том, что при передаче информации по скрытому каналу происходят искажения, метод решения задачи будет несколько иной. Здесь правильность приема обеспечивается не исправлением ошибки, а многократными передачами одной и той же информации, пока, наконец, она не будет передана безошибочно. Подход же на основе исправления ошибок также возможен, однако предполагается, что он будет рассматриваться в последующих работах.

Сначала будет представлен главный результат статьи о среднем числе попыток. Будет дано не только точное, но и практически значимое приближенное выражение для этого среднего. Далее будет доказана теорема о "неулучшаемости циклической передачи". В дополнение к этому будут представлены оценки, а также для дисперсии числа попыток будут приведены два выражения — в виде конечной и в виде бесконечной сумм. Под оценкой будем иметь в виду число передач цикла блоков, необходимое, чтобы вероятность того, что после выполненной передачи этих циклов какой-либо блок не был безошибочно принят, не превышала ε.

Перечисленные выше результаты применимы не только к рассматриваемому скрытому каналу. Они применимы к решению задач, которые сводятся к следующей абстрактной постановке. Пусть есть некий передатчик, передающий m разных блоков, причем блок передается правильно с некой вероятностью p. Есть также некий приемник, сохраняющий блок при условии его успешного приема. Требуется найти число переданных блоков до того, как приемник соберет "полную коллекцию" из m различных блоков.

Далее будут представлены: принятая автором схема кодирования/декодирования передаваемой информации; доказательство ее корректности; характер ошибок, возникающих при передаче информации. Также будут описаны результаты экспериментов по измерению частоты ошибок, а также дана их теоретическая интерпретация.

1. Среднее число попыток передачи блока

Для оценки среднего времени передачи оценим среднее число циклов передач блоков, идущих до тех пор, пока все без исключения блоки не будут переданы безошибочно. Свою роль играет также поправка на стоимость блока — затраты времени на передачу его заголовка. Введем главный объект изучения — случайную величину L, принимающую значение числа циклов передачи, произошедших до момента полного приема. Будет дан ответ об основном интересном с точки зрения практики свойстве, а именно о его математическом ожидании, свидетельствующем о том, сколько раз в среднем нужно передать цикл блоков. Так как в рамках рассматриваемой модели принимаем предположение о том, что все блоки передаются одинаковое время, то среднее время передачи пропорционально этому математическому ожиданию EL.

Пусть всего передается m блоков. Вероятность передать блок безошибочно — p. Успешность передачи каждого блока независима от успешности передачи любого другого блока.

Прежде всего, дадим формальное определение такой случайной величины. Формальное определение ее будем давать через определение более простой случайной величины — того же самого смысла, но только для отдельно взятого блока с номером i среди блоков с номерами 1, ..., m.

Обозначения. $P()$ обозначает вероятностную меру, определенную на событиях. Например, $P(l^i = k)$ — это вероятность события "случайная величина l^i принимает значение k".

Для произвольной случайной величины η ее математическое ожидание обозначается как $E\eta$, а дисперсия — как $D\eta$.

Для биномиального коэффициента используется традиционное обозначение $\binom{r}{m} = m!/(r!(m-r)!)$.

Определение. Случайная величина l^i — величина, значение которой указывает, с какой попытки передался i-й блок.

Утверждение. $P(l^i = k) = (1-p)^{k-1}p$.

Действительно, передать блок с k-й попытки — это не передать его на первых $k-1$ попытках и передать именно на k-й.

Утверждение. $P(l^i \leqslant k) = (1-p)^k$,

$$P(l^i \leqslant k) = P(l^i = 1) + P(l^i = 2) + ... + P(l^i = k) =$$
$$= (1-p)^0 p + (1-p)^1 p + ... + (1-p)^{k-1} p =$$
$$= p(1 + ...(1-p)^{k-1}) = p\frac{1-(1-p)^k}{1-(1-p)} = 1-(1-p)^k.$$

Определение. Определим $L = \max(l^1, ..., l^m)$ как случайную величину, имеющую значение среднего числа циклов передачи.

Замечание. На самом деле число переданных блоков меньше, чем $m \times L$, так как на последнем цикле после передачи последнего, еще не подтвержденного блока, далее блоки не передаются. Пусть l' — число блоков в этом остатке последнего цикла. Это означает, что всего передалось $m \times L - l'$ блоков. Однако тогда $E(mL) \leqslant E(mL - L') = mEL - E(l') \leqslant mEL - m$ (так как $0 \leqslant l' \leqslant m$), т. е. $EL \leqslant \frac{1}{m}E(mL - l') \leqslant EL - 1$. Для цели получения оценки это отклонение несущественно.

Вычислим точное значение EL, которое сформулируем в виде следующей теоремы.

Теорема. Математическое ожидание

$$EL = \sum_{r=1}^{m} \frac{\binom{r}{m}(-1)^{r+1}}{1-(1-p)^r}.$$

Доказательство.

Для начала обозначим некоторые подготовительные вычисления, опираясь на введенные выше определения.

Прежде всего установим в виде утверждения следующий общеизвестный факт.

Утверждение. $EL = \sum_{s=1}^{\infty} P(L \geqslant s)$.

Действительно,

$$\sum_{s=1}^{\infty} P(L \geqslant s) = \sum_{s=1}^{\infty}\sum_{k=s}^{\infty} P(L = k) = P(L=1) +$$
$$+ 2P(L=2) + 3P(L=3) + ... = EL.$$

Порядок суммирования изменен математически корректно. Так как все слагаемые положительные, то порядок суммирования можно менять как угодно.

Утверждение. $P(L \leqslant k) = (1-(1-p)^k)^m$.

Утверждение верно в силу того, что $P(L \leqslant k) = P(\max(l^1, ..., l^m) \leqslant k) = P(l^1 \leqslant k, l^2 \leqslant k, ..., l^m \leqslant k) = P(l^1 \leqslant k)...P(l^m \leqslant k) = (1-(1-p)^k)^m$, так как случайные величины $l^1, ..., l^m$ независимы.

Проведем суммирование, используя формулу бинома Ньютона.

Утверждение. $P(L \geqslant s) = \sum_{r=1}^{m}\binom{r}{m}(1-p)^{sr-r}(-1)^{r+1}$.

Действительно,

$$P(L \geqslant s) = 1 - P(L \leqslant s-1) = 1 - \left(1-(1-p)^{s-1}\right)^m =$$
$$= 1 - \sum_{r=0}^{m}\binom{r}{m}\left(-(1-p)^{s-1}\right)^r = 1 - \sum_{r=0}^{m}\binom{r}{m}((1-p)^{s-1})^r(-1)^r =$$
$$= 1 + \sum_{r=0}^{m}\binom{r}{m}(1-p)^{rs-r}(-1)^{r+1} =$$
$$= 1 + (-1) + \sum_{r=1}^{m}\binom{r}{m}(1-p)^{rs-r}(-1)^{r+1} =$$
$$= \sum_{r=1}^{m}\binom{r}{m}(1-p)^{rs-r}(-1)^{r+1}.$$

Теперь все готово к непосредственному вычислению суммы. Вычислим ее, применив перестановку конечного и бесконечного суммирований.

Тогда

$$EL = \sum_{s=1}^{\infty}P(L \geqslant s) = \sum_{s=1}^{\infty}\sum_{r=1}^{m}\binom{r}{m}(1-p)^{sr-r}(-1)^{r+1} =$$
$$= \sum_{r=1}^{m}\sum_{s=1}^{\infty}\binom{r}{m}(1-p)^{sr-r}(-1)^{r+1} =$$
$$= \sum_{r=1}^{m}\binom{r}{m}(-1)^{r+1}\sum_{s=1}^{\infty}\left((1-p)^r\right)^{s-1} =$$
$$= \sum_{r=1}^{m}\binom{r}{m}(-1)^{r+1}\sum_{s=0}^{\infty}\left((1-p)^r\right)^s = \sum_{r=1}^{m}\frac{\binom{r}{m}(-1)^{r+1}}{1-(1-p)^r}.$$

Замечание (о перестановке суммирований). Конечное число рядов можно складывать почленно. Суммарный результат при этом равен сумме сумм этих рядов. Однако необходимо проверить, что каждый из них сходится сам по себе. В приведенном выше вычислении именно это и было сделано — после перестановки конечного и бесконечного пределов суммирования была подсчитана сумма каждого ряда самого по себе.

Теорема доказана.

2. Приближенная оценка *EL*

Полученное выше выражение является точным, но неудобным для оценки среднего времени передачи

на практике. По этой причине необходимо найти достаточно простую приближенную оценку. Для ее получения будет использован метод оценки суммы ряда интегралом. Предварительно введем необходимые обозначения.

Положим $q = 1 - p$, $f(k) = 1 - (1 - q^k)^m$.

Тогда $P(L \geqslant s) = f(s-1)$, $EL = \sum\limits_{s=1}^{\infty} P(L \geqslant s) = \sum\limits_{s=1}^{\infty} f(s-1) = \sum\limits_{s=0}^{\infty} f(s)$.

Теорема. Пусть $f(k)$ непрерывна (на всех значениях $k \geqslant 0$), $f \geqslant 0$, f монотонно убывает, ряд $\sum\limits_{k=0}^{\infty} f(k)$ сходится.

Тогда:

1) сходится несобственный интеграл $\int\limits_0^{\infty} f(k)\,dk$;

2) $0 \leqslant \sum\limits_{k=0}^{\infty} f(k) - \int\limits_0^{\infty} f(k)dk \leqslant f(0)$.

Доказательство.

1. Оценим интеграл по отрезку длины 1. $f(n+1) = \int\limits_n^{n+1} f(n+1)dk \leqslant \int\limits_n^{n+1} f(k)dk \leqslant \int\limits_n^{n+1} f(n)dk = f(n)$ по монотонности f.

2. Сложим эти оценки при $n = 0...s$. Получим $\sum\limits_{n=0}^{s} f(n+1) \leqslant \int\limits_0^{s+1} f(k)\,dk \leqslant \sum\limits_{n=0}^{s} f(n)$.

3. Устремим это неравенство к пределу $s \to \infty$. При этом автоматически следует, что несобственный интеграл сходится, так как, во-первых, он ограничен сверху суммой ряда, а во-вторых, в силу $f \geqslant 0$ он монотонно возрастает. Имеется в виду несобственный интеграл как функция $F(x) = \int\limits_0^x f(x)dx$.

Получим: $\sum\limits_{n=0}^{\infty} f(n+1) \leqslant \int\limits_0^{\infty} f(k)\,dk \leqslant \sum\limits_{n=0}^{\infty} f(n)$.

4. Заключительные равносильные преобразования:

$$\sum_{n=0}^{\infty} f(n+1) - \sum_{n=0}^{\infty} f(n) \leqslant \int_0^{\infty} f(k)dk - \sum_{n=0}^{\infty} f(n) \leqslant 0.$$

$$-f(0) \leqslant \int_0^{\infty} f(k)dk - \sum_{n=0}^{\infty} f(n) \leqslant 0,$$

$$0 \leqslant \sum_{n=0}^{\infty} f(n) - \int_0^{\infty} f(k)\,dk \leqslant f(0).$$

Теорема доказана.

Теперь можно, пользуясь приближением ряда интегралом, получить искомую оценку как утверждение следующей теоремы.

Теорема.

$EL = -\dfrac{1}{\ln(1-p)} H_m + \gamma(m,p)$, где $\gamma(m,\ p) \in [0,1]$, а $H_m = 1 + \dfrac{1}{2} + \dfrac{1}{3} + ... + \dfrac{1}{m}$ — сумма гармонического ряда.

Заметим, что для самого гармонического ряда известна его оценка логарифмом: $H_m = \ln m + \varepsilon_n + R$, где $\varepsilon_n \to 0$, а $R = 0{,}5772...$ — так называемая постоянная Эйлера-Маскерони [6].

Доказательство.

Проверим условия сформулированной выше теоремы для нашей функции $f(k) = 1 - (1 - q^k)^m$. Непрерывность очевидна. Так как $0 \leqslant q^k \leqslant 1$, то $0 \leqslant 1 - q^k \leqslant 1$, $0 \leqslant (1 - q^k)^m \leqslant 1$, $0 \leqslant 1 - (1 - q^k)^m \leqslant 1$. Неотрицательность также установлена.

Так как $0 \leqslant q \leqslant 1$, то (по k) q^k монотонно убывает. Следовательно, $1 - q^k$, а также $(1 - q^k)^m$ монотонно возрастает, и, как следствие, $f(k) = 1 - (1 - q^k)^m$ монотонно убывает.

Сходимость ряда следует из того, что ранее уже подсчитана его сумма EL.

Следовательно, $0 \leqslant \sum\limits_{k=0}^{\infty} f(k) - \int\limits_0^{\infty} f(k)dk \leqslant f(0) = 1$,

т. е. $EL = \sum\limits_{k=0}^{\infty} f(k) = \int\limits_0^{\infty} f(k)dk + \gamma(m,\ p)$, где $\gamma(m,p) \in [0,\ 1]$.

Теперь вычислим интеграл:

$I = \int\limits_0^{\infty} f(k)dk = \int\limits_0^{\infty} \left(1 - \left(1 - q^k\right)^m\right) dk$. Проведем следующую замену переменных: $y = 1 - q^k$. Тогда $dy = -(\ln q) q^k dk$, $dk = -\dfrac{1}{(\ln q) q^k} dy = -\dfrac{1}{(\ln q)(1-y)} dy$, $y(0) = 0$, $y(\infty) = 1$.

$$I = -\frac{1}{\ln q} \int_0^1 \frac{1 - y^m}{1 - y} dy =$$

$$= -\frac{1}{\ln q} \int_0^1 \left(1 + y + y^2 + ... + y^{m-1}\right) dy =$$

$$= -\frac{1}{\ln q} \left(1 + \frac{y^2}{2}\Big|_0^1 + \frac{y^3}{3}\Big|_0^1 + ... \frac{y^m}{m}\Big|_0^1 \right) =$$

$$= -\frac{1}{\ln q} \left(1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{m} \right) = -\frac{H_m}{\ln q}.$$

Теорема доказана.

Таким образом, с увеличением числа блоков среднее число попыток передать блок возрастает лишь логарифмически, т. е. достаточно медленно для использования этого способа передачи на практике.

3. ε-оценки

Для оценки пропускной способности канала передачи данных интерес представляет ответ на следующий вопрос: сколько раз нужно передавать цикл блоков, чтобы вероятность успешно передать файл целиком, т. е. успешно передать все блоки, была больше некоего наперед заданного значения $1 - \varepsilon$.

Теорема. Пусть $\varepsilon \in \mathbb{R}$, $0 < \varepsilon < 1$. Тогда для того чтобы вероятность успешной передачи каждого блока была не меньше, чем $1 - \varepsilon$, необходимо и достаточно передать цикл блоков $k_{\min} = \left\lceil \frac{\ln\left(1 - (1-\varepsilon)^{\frac{1}{m}}\right)}{\ln q} \right\rceil$ раз, где $\lceil x \rceil$ — округление вверх значения x.

Доказательство.

Случайная величина L принимает значение числа циклов передачи, произошедших до полного приема всех блоков. Значит, событие вида $L \leqslant k$ происходит тогда и только тогда, когда по результатам передачи цикла блоков k раз подряд все блоки оказываются переданными, т. е. каждый из них хотя бы один (из k раз) был передан корректно.

Следовательно, в формальном виде задача может быть сформулирована так: найти такие k, что $P(L \leqslant k) \geqslant 1 - \varepsilon$, а точнее минимальное из этих k, так как, если $k_1 \geqslant k_2$, то $P(L \leqslant k_1) \geqslant P(L \leqslant k_2)$.

Распишем наше выражение: $P(L \leqslant k) = (1 - q^k)^m \geqslant 1 - \varepsilon$. После его преобразования получим:

$$k \geqslant \frac{\ln\left(1 - (1-\varepsilon)^{\frac{1}{m}}\right)}{\ln q},$$

что в итоге дает $k_{\min} = \left\lceil \frac{\ln\left(1 - (1-\varepsilon)^{\frac{1}{m}}\right)}{\ln q} \right\rceil$. Теорема доказана.

Следствие. Если $c_1 \leqslant p \leqslant c_2$, то

$$\frac{\ln\left(1 - (1-\varepsilon)^{\frac{1}{m}}\right)}{\ln(1-c_2)} \leqslant k_{\min} \leqslant \left\lceil \frac{\ln\left(1 - (1-\varepsilon)^{\frac{1}{m}}\right)}{\ln(1-c_1)} \right\rceil.$$

Это выражение следует из того, что согласно представленному выше выражению, $k_{\min}(p)$ является убывающей функцией.

В данном выражении $k_{\min}$ содержатся три параметра: ε, q, и m. Приведем таблицы его значений для $\varepsilon = 0{,}5;\ 1{,}0;\ 0{,}05$; $m = 1,\ 2,\ 5,\ 10,\ 20,\ 50,\ 75,\ 100$; $q = 0{,}1;\ 0{,}5;\ 0{,}7$. Общая таблица значений является трехмерной, поэтому она будет записана как три двумерные таблицы 1, 2, 3, взятые для трех заданных значений ε соответственно. Согласно представленным в табл. 1—3 данным легко видеть, что даже при значительной вероятности наличия ошибок (большей 0,5) и при довольно длинном сообщении число циклов, гарантирующее успешную передачу с вероятностью, близкой к 1, не превосходит нескольких десятков.

Таблица 1

Значения $k_{\min}$ для $\varepsilon = 0{,}5$

q	m							
	1	2	5	10	20	50	75	100
0,1	1	1	1	2	2	2	3	3
0,5	1	2	3	4	5	7	7	8
0,7	2	4	6	8	10	13	14	14

Таблица 2

Значения $k_{\min}$ для $\varepsilon = 0{,}1$

q	m							
	1	2	5	10	20	50	75	100
0,1	1	2	2	2	3	3	3	3
0,5	4	5	6	7	8	9	10	10
0,7	7	9	11	13	15	18	19	20

Таблица 3

Значения $k_{\min}$ для $\varepsilon = 0{,}05$

q	m							
	1	2	5	10	20	50	75	100
0,1	2	2	2	3	3	3	4	4
0,5	5	6	7	8	9	10	11	11
0,7	9	11	13	15	17	20	21	22

4. Дисперсия *DL*

Другой величиной, которая может представлять интерес в контексте рассматриваемой модели скрытого канала, является дисперсия *DL*. Значение этой величины почти так же важно, как и значение математического ожидания. Она предоставляет информацию о том, насколько (в среднем) реальное время передачи отклоняется от среднего времени, пропорционального математическому ожиданию.

Далее будет дано выражение *DL* как бесконечной, а также как конечной суммы. Однако такое представление не дает единственную форму оценки порядка дисперсии.

Теорема. Верны следующие равенства:

$$1)\ DL = \sum_{k=0}^{\infty} \left(1 - q^k\right)^m \left(1 - \left(1 - q^k\right)^m\right) + 2\sum_{k=0}^{\infty} \left(1 - q^k\right)^m \times$$

$$\times \sum_{p=1}^{\infty} \left(1 - \left(1 - q^{k+p}\right)^m\right);$$

2) $$DL = 2\sum_{r=1}^{m}\frac{\binom{r}{m}(-1)^{r+1}}{(1-(1-p)^r)^2} - \sum_{r=1}^{m}\frac{\binom{r}{m}(-1)^{r+1}}{1-(1-p)^r} - \left(\sum_{r=1}^{m}\frac{\binom{r}{m}(-1)^{r+1}}{1-(1-p)^r}\right)^2.$$

Доказательство равенства 1.

Прежде всего, запишем:

$$L = I_{L\geqslant 1} + I_{L\geqslant 2} + I_{L\geqslant 3} + \ldots = \sum_{k=1}^{\infty} I_{L\geqslant k}.$$

Действитетельно, если $L(\omega) = r$, то

$$\sum_{k=1}^{\infty} I_{L\geqslant k}(\omega) = \sum_{k=1}^{\infty}[r \geqslant k] = \sum_{k=1}^{r} 1 = r.$$

Здесь использовано следующее обозначение: [условие] = 1, если условие истинно, [условие] = 0, если условие ложно.

По свойствам ковариаций:

$$DL = \operatorname{cov}(L, L) = \operatorname{cov}\left(\sum_{k_1=1}^{\infty} I_{L\geqslant k_1}, \sum_{k_2=1}^{\infty} I_{L\geqslant k_2}\right) =$$

$$= \sum_{k_1=1}^{\infty}\sum_{k_2=1}^{\infty}\operatorname{cov}\left(I_{L\geqslant k_1}, I_{L\geqslant k_2}\right) = \sum_{k_1=1}^{\infty}\sum_{k_2=1}^{\infty}\operatorname{cov}\left(1 - I_{L<k_1}, 1 - I_{L<k_2}\right) =$$

$$= \sum_{k_1=1}^{\infty}\sum_{k_2=1}^{\infty}\operatorname{cov}\left(I_{L\leqslant k_1-1}, I_{L\leqslant k_2-1}\right) = \sum_{k_1=0}^{\infty}\sum_{k_2=0}^{\infty}\operatorname{cov}\left(I_{L\leqslant k_1}, I_{L\leqslant k_2}\right) =$$

$$= \sum_{k_1=0}^{\infty}\sum_{k_2=0}^{\infty} E\left(I_{L\leqslant k_1} I_{L\leqslant k_2}\right) - E\left(I_{L\leqslant k_1}\right) E\left(I_{L\leqslant k_2}\right) =$$

$$= \sum_{k_1=0}^{\infty}\sum_{k_2=0}^{\infty} P\left(L\leqslant \min(k_1, k_2)\right) - P(L\leqslant k_1)\,P(L\leqslant k_2).$$

Возможны три случая: $k_1 = k_2$, $k_1 < k_2$ и $k_1 > k_2$. В соотвествии с ними распишем двойную сумму и раскроем вероятности. В силу симметрии k_1 и k_2 можно менять местами, поэтому при суммировании по условиям двух последних случаев получаются одинаковые суммы:

$$DL = \sum_{k=0}^{\infty} P(L\leqslant k) - P(L\leqslant k)P(L\leqslant k) + 2\sum_{k_1<k_2} P\left(L\leqslant k_1\right) - P\left(L\leqslant k_1\right)P\left(L\leqslant k_2\right) =$$

$$= \sum_{k=0}^{\infty} P(L\leqslant k)(1 - P(L\leqslant k)) + 2\sum_{k=0}^{\infty} P(L\leqslant k) \times$$

$$\times \sum_{p=1}^{\infty}(1 - P(L\leqslant k+p)) = \sum_{k=0}^{\infty}\left(1-q^k\right)^m\left(1-\left(1-q^k\right)^m\right) +$$

$$+ 2\sum_{k=0}^{\infty}\left(1-q^k\right)^m \sum_{p=1}^{\infty}\left(1-\left(1-q^{k+p}\right)^m\right).$$

Первое слагаемое представленной суммы поддается дальнейшей оценке, которую сформулируем в виде утверждения.

Утверждение.

$$-\frac{1}{2\ln(1-p)} - 1 \leqslant S_1 \leqslant -\frac{1}{\ln(1-p)} + 1,$$

где $S_1 = \sum_{k=0}^{\infty}\left(1-q^k\right)^m\left(1-\left(1-q^k\right)^m\right)$.

Действительно,

$$S_1 = \sum_{k=0}^{\infty} 1 - \left(1-q^k\right)^{2m} - \sum_{k=0}^{\infty} 1 - \left(1-q^k\right)^m =$$

$$= EL_{2m} - EL_m = -\frac{1}{\ln(1-p)}(H_{2m} - H_m) + \gamma_1 - \gamma_2.$$

Очевидно, что $\gamma_1 - \gamma_2 \in [-1, 1]$.

Оценим разность гармонических сумм:

$$\frac{1}{2} = \frac{m}{2m} = \frac{1}{2m} + \ldots + \frac{m}{2m} \leqslant H_{2m} - H_m =$$

$$\frac{1}{m+1} + \ldots + \frac{m}{2m} \leqslant \frac{1}{m+1} + \ldots + \frac{m}{m+1} = \frac{m}{m+1} \leqslant 1.$$

Утверждение доказано.

Доказательство равенства 2.

Сделаем некоторые подготовительные замечания технического характера перед расчетом второго момента EL^2 (записи в виде конечной суммы), необходимого для подсчета дисперсии $DL = EL^2 - (EL)^2$.

Во-первых, так как L принимает только целые значения, то условие вида $L \geqslant x$ равносильно условию $L \geqslant \lceil x\rceil$. Во-вторых, найдем число решений уравнения вида $\lceil\sqrt{s}\rceil = t$, где t — это целое число.

Утверждение. Уравнение $\lceil\sqrt{s}\rceil = t$ имеет $2t-1$ решений в целых числах.

Доказательство.

1. Перепишем уравнение в виде двойного неравенства: $t - 1 < \sqrt{s} \leqslant t$. После возведения в квадрат: $(t-1)^2 < s \leqslant t^2$.

2. В целых числах оно имеет решения: $s = (t-1)^2 + 1$, $(t-1)^2 + 2$, ..., t^2. Всего $t^2 - (t-1)^2 = 2t - 1$ решений. Утверждение доказано.

В-третьих, напомним достаточно легко проверяемый факт (о сумме бесконечного ряда), сформулированный в виде следующего утверждения.

Утверждение. При $q \in (0, 1)$ выполнено $\sum_{s=1}^{\infty} s q^{s-1} = \frac{1}{(1-q)^2}$.

Действительно, $\sum_{s=0}^{\infty} q^s = \frac{1}{1-q}$. Почленно дифференцируя степенной ряд, получаем требуемое.

Итак, раскроем выражение для второго момента:

$$EL^2 = \sum_{s=1}^{\infty} P\left(L^2 \geqslant s\right) = \sum_{s=1}^{\infty} P\left(L \geqslant \sqrt{s}\right) = \sum_{s=1}^{\infty} P\left(L \geqslant \lceil \sqrt{s} \rceil\right) =$$

$$= \sum_{t=1}^{\infty} \sum_{\lceil \sqrt{s} \rceil = t} P(L \geqslant t) = \sum_{t=1}^{\infty} (2t-1) P(L \geqslant t) =$$

$$= 2\sum_{t=1}^{\infty} tP(L \geqslant t) - \sum_{t=1}^{\infty} P(L \geqslant t) = 2\sum_{s=1}^{\infty} sP(L \geqslant s) - EL.$$

Теперь раскроем $P(L \geqslant s)$:

$$\sum_{s=1}^{\infty} sP(L \geqslant s) = \sum_{s=1}^{\infty} s \sum_{r=1}^{m} \binom{r}{m} (1-p)^{sr-r} (-1)^{r+1} =$$

$$= \sum_{s=1}^{\infty} \sum_{r=1}^{m} s \binom{r}{m} q^{sr-r} (-1)^{r+1} = \sum_{r=1}^{m} \sum_{s=1}^{\infty} s \binom{r}{m} (q^r)^{s-1} (-1)^{r+1} =$$

$$= \sum_{r=1}^{m} (-1)^{r+1} \binom{r}{m} \sum_{s=1}^{\infty} s(q^r)^{s-1} = \sum_{r=1}^{m} \frac{\binom{r}{m} (-1)^{r+1}}{\left(1-q^r\right)^2}.$$

В итоге получаем:

$$DL = EL^2 - (EL)^2 = 2\sum_{s=1}^{\infty} sP(L \geqslant s) - EL - EL^2 =$$

$$= 2\sum_{r=1}^{m} \frac{\binom{r}{m} (-1)^{r+1}}{\left(1-q^r\right)^2} - \sum_{r=1}^{m} \frac{\binom{r}{m} (-1)^{r+1}}{1-q^r} - \left(\sum_{r=1}^{m} \frac{\binom{r}{m} (-1)^{r+1}}{1-q^r} \right)^2.$$

Теорема доказана.

Замечание. Аналогично можно вывести подобные формулы для старших моментов. Такой вывод сводится к суммированию бесконечного ряда вида $\sum_{s=1}^{\infty} s^n q^{s-1}$.

5. Неулучшаемость стратегии передачи блоков

В рамках рассматриваемой модели скрытого канала блоки передаются циклически. Сначала первый блок, потом второй, третий и так до последнего, после чего они снова передаются в том же порядке. Возникает вопрос, а что если передавать блоки не в таком, а в каком-то ином порядке? Например, передадим два раза подряд первый блок, потом два раза подряд второй и так далее по тому же циклу.

Передача блока два раза подряд — это все равно что передача блока безошибочно с вероятностью $1-(1-p)^2$ вместо вероятности p. Однако при этом вдвое увеличивается число циклов передачи. Для множителя из выражения для среднего: $2\frac{1}{\ln(1-p)^2} = \frac{1}{\ln(1-p)}$. При этом сколько выигрываем на повышении вероятности успешной передачи, ровно столько и проигрываем на том, что эта передача равноценна двум передачам блоков, а не одной. Так же обстоит дело с тремя и более передачами блоков подряд. Как следствие, возникает вопрос, будет ли так же дело обстоять в случае "произвольного порядка" передачи блоков, т. е. не может ли существовать такая стратегия передачи, которая лучше, чем представленный выше традиционный цикл? Интуитивно очевидно, что так быть не должно. Представим доказательство этого утверждения. Начнем с формально строгого определения стратегии.

Определение. Стратегия передачи m блоков — это последовательность чисел 1...m (т. е. функция $f:\mathbb{N} \to \{1...m\}$) такая, что каждое из этих чисел встречается в этой последовательности бесконечное число раз.

Определение. Стратегия циклической передачи: $c(n) = (n-1) \bmod m + 1$. Именно согласно данной стратегии передавались блоки.

Определение. С определенной стратегией f связаны функции h_k^f, $k = 1...m$: $h_k^f(i) = \left|\{j \mid j \leqslant i,\ f(j) = k\}\right|$, т. е. число блоков с номером k среди первых i переданных блоков.

Заметим, что $h_1^f(i) + ... + h_m^f(i) = i$.

Со стратегиями очевидным образом связаны определенные события, которые могут произойти при передаче очередного блока. При передаче i-го (по счету) блока передается блок с номером $f(i)$. Для него есть две возможности: он был передан успешно или неуспешно. Таким образом, можно определить последовательность событий, соответствующих передаче очередного блока. На основе данной последовательности можно определить также и последовательность событий, означающих, что при передаче i-го блока (по счету) блок с номером k уже был успешно передан. Наконец, определяются события, означающие "при передаче i-го (по счету) блока передача уже завершена". Для доказательства того, что некая стратегия f не лучше циклической стратегии c, требуется лишь доказать, что вероятности данных событий, связанных с f, не больше, чем вероятности аналогичных событий, связанных с c.

Определение. A_i — событие "передача i-го (по счету) блока произошла успешно". Как и ранее, полагаем $P(A_i) = p$, $q = 1-p$. Данные события, согласно базовому предположению, являются совместно независимыми:

$P\left(A_{i_1} \cap A_{i_2} \cap ... \cap A_{i_m}\right) = P\left(A_{i_1}\right) P\left(A_{i_2}\right) ... P\left(A_{i_m}\right)$ для любого конечного набора событий A_{i_1}, A_{i_2}, ..., A_{i_m}.

Утверждение. События $\overline{A_i}$ также совместно независимы.

Действительно, по формуле включений-исключений:

$$P\left(\overline{A_{i_1}} \cap \overline{A_{i_2}} \cap ... \cap \overline{A_{i_m}}\right) = 1 - P\left(A_{i_1} \cup A_{i_2} \cup ... \cup A_{i_m}\right) =$$

$$= 1 - \left(\sum_{\{k_1...k_t\} \subset \{i_1,...,i_m\}, t>0} (-1)^{t+1} P\left(A_{k_1} \cap A_{k_2} ... \cap A_{k_t}\right) \right) =$$

$$= \sum_{\{k_1...k_t\} \subset \{i_1,...,i_m\}} (-1)^t P\left(A_{k_1}\right) P\left(A_{k_2}\right) ... P\left(A_{k_t}\right) =$$

$$= \left(1 - P\left(A_{i_1}\right)\right)\left(1 - P\left(A_{i_2}\right)\right) ... \left(1 - P\left(A_{i_m}\right)\right) =$$

$$= P\left(\overline{A_{i_1}}\right) P\left(\overline{A_{i_2}}\right) ... P\left(\overline{A_{i_m}}\right).$$

Определение. $B_i^{f,k}$ — событие "при передаче первых i блоков по стратегии f блок с номером k хотя бы один раз будет передан успешно". Формализовано: $\omega \in B_i^{f,k} \Leftrightarrow \exists j \leqslant i;\ \omega \in A_j \wedge f(j) = k$.

Откуда $B_i^{f,k} = \bigcup\limits_{1 \leqslant j \leqslant i, f(j)=k} A_j$.

Утверждение. $P\left(B_i^{f,k}\right) = 1 - q^{h_k(i)}$.

Действительно,

$$P(B_i^{f,k}) = P\left(\bigcup_{1\leqslant j\leqslant i,\ f(j)=k} A_j\right) = P\left(\overline{\bigcap_{1\leqslant j\leqslant i,\ f(j)=k} \overline{A_j}}\right) =$$

$$= 1 - P\left(\bigcap_{1\leqslant j\leqslant i,\ f(j)=k} \overline{A_j}\right) = 1 - \prod_{1\leqslant j\leqslant i,\ f(j)=k} P(\overline{A_j}) =$$

$$= 1 - \prod_{1\leqslant j\leqslant i,\ f(j)=k} q = 1 - q^{h_k(i)}.$$

Лемма. Пусть $\{A \cup B\} \cup \{C_i\}$ — множество совместно независимых событий. Тогда $\{A \cup B\} \cup \{C_i\}$ также является множеством совместно независимых событий.

1. Положим $C = C_{i_1} \cap ... \cap C_{i_k}$. Тогда, по совместной независимости, $P(A \cap C) = P(A)P(C)$, $P(B \cap C) = P(B)P(C)$, $P(A \cap B \cap C) = P(A)P(B)P(C)$, $P(A \cap B) = P(A)P(B)$.

$$2.\ P\left((A \cup B) \cap C\right) = P\left((A \cap C) \cup (B \cap C)\right) =$$

$$= P(A \cap C) + P(B \cap C) - P(A \cap B \cap C) =$$

$$= P(A)P(C) + P(B)P(C) - P(A)P(B)P(C) =$$

$$= P(C)\left(P(A) + P(B) - P(A)P(B)\right) = P(C)P(A \cup B).$$

3. В силу произвола в выборе C_{i_1}, ..., C_{i_k} данных проверок достаточно для доказательства искомого утверждения. Лемма доказана.

Следствие. События $B_i^{f,1}$, ..., $B_i^{f,m}$ совместно независимы.

Действительно, возьмем множество событий $\{A_1, A_2, ..., A_i\}$, сгруппируем их по значению f на индексе, а потом m раз применим доказанную выше лемму.

Определение. C_i^f — событие "при передаче первых i блоков по стратегии f блок каждого вида будет успешно передан хотя бы один раз, т. е. прием блоков будет завершен". Таким образом, $C_i^f = B_i^{f,1} \cap ... \cap B_i^{f,m}$.

Вероятность данного события будем обозначать как p_i^f. Согласно следствию из леммы

$$p_i^f = P\left(C_i^f\right) = P\left(B_i^{f,1}\right)...P\left(B_i^{f,m}\right) =$$

$$= \left(1 - q^{h_1(i)}\right)...\left(1 - q^{h_m(i)}\right).$$

Определение. Будем считать, что стратегия f_1 не лучше стратегии f_2 (обозначение: $f_1 \leqslant f_2$), если для всех i верно $p_i^{f_1} \leqslant p_i^{f_2}$.

Перед доказательством главного результата изучим некоторые свойства вероятностей p_i^f. В представленной далее лемме требуемый результат по сути уже содержится.

Определение. Будем обозначать $S(m) = 1 - q^m$, $S(m_1, ..., m_t) = S(m_1)...S(m_t) = \left(1 - q^{m_1}\right)...\left(1 - q^{m_t}\right)$. Таким образом, $p_i^f = S\left(h_1^f(i), ..., h_m^f(i)\right)$.

Лемма. Пусть $a \leqslant b$, тогда $S(a, b + 1) \leqslant S(a + 1, b)$.

Доказательство.

1. $a \leqslant b \Rightarrow q^a \geqslant q^b$, так как $0 \leqslant q \leqslant 1$.

2. $q^a \geqslant q^b \Rightarrow (1 - q)q^a \geqslant (1 - q)q^b \Rightarrow q^a - q^{a+1} \geqslant q^b - q^{b+1} \Rightarrow q^a + q^{b+1} \geqslant q^b + q^{a+1}$.

3. $1 - q^a - q^{b+1} \leqslant 1 - q^b - q^{a+1} \Rightarrow 1 - q^a - q^{b+1} + q^{a+b+1} \leqslant 1 - q^b - q^{a+1} + q^{a+b+1} \Rightarrow (1 - q^a)(1 - q^{b+1}) \leqslant (1 - q^{a+1})(1 - q^b) \Rightarrow S(a, b + 1) \leqslant S(a + 1, b)$.

Лемма доказана.

Обозначение. Пропуск переменной в списке переменных функции обозначается знаком циркумфлекса (крышечки) над пропускаемой переменной. То есть, например, $S\left(x, \hat{a}, y, b, z\right) = S(x, y, z)$.

Следствие. При $a \leqslant b$ $S(..., a, ..., b + 1, ...) \leqslant S(..., a + 1, ..., b, ...)$. Аргументов может быть сколько угодно, и a, b могут стоять на любых местах. Отсюда следует $S(..., a, ..., b, ...) = S(a, b)S\left(..., \hat{a}, ..., \hat{b}, ...\right)$.

Из представленных выкладок следует: из каждого набора посредством "перекидывания единички между членами набора" можно добиться "наиболее равномерного распределения значений". Ранее доказано, что при "перекидывании единички в сторону большей равномерности" функция S не убывает. Очевидно также, что циклическая передача порождает только эти "наиболее равномерные распределения". Поэтому на порождаемых циклической передачей наборах значение S должно быть максимальным. Далее формализуем представленные рассуждения.

Прежде всего, введем еще несколько технических понятий.

Определение. Будем обозначать

$$H_i = \left\{(h_1, ..., h_m) \in \mathbb{N}^m \middle| h_1 + ... + h_m = i\right\}.$$

Утверждение. Для любой стратегии f при всех i $\left(h_1^f(i), ..., h_m^f(i)\right) \in H_i$, так как $h_1^f(i) + ... + h_m^f(i) = i$.

Определение.

$$M_i = \{(h_1, ..., h_m) \in H_i \mid \forall (h'_1, ..., h'_m) \in$$
$$\in H_i\, S(h'_1, ...h'_m) \leqslant S(h_1, ...h_m)\}.$$

То есть M_i — это те наборы из H_i, на которых S принимает максимальные значения. В силу конечности множества H_i, $M_i \neq \varnothing$.

Лемма. Пусть стратегия f такова, что для всех i $\left(h_1^f(i), ..., h_m^f(i)\right) \in M_i$. Тогда данная стратегия является неулучшаемой, т. е. для любой стратегии f' выполнено $f' \leqslant f$.

1. Предположим обратное. Тогда, согласно определению отношения "не лучше", существует такое i, что $p_i^{f'} > p_i^f$, т. е.

$$S\left(h_1^{f'}(i), ..., h_m^{f'}(i)\right) > S\left(h_1^f(i), ..., h_m^f(i)\right),$$

причем оба

$$\left(h_1^{f'}(i),\ ...,\ h_m^{f'}(i)\right),\ \left(h_1^{f}(i),.\ ...,\ h_m^{f}(i)\right) \in H_i.$$

2. Однако по условию $\left(h_1^{f}(i),\ ...,\ h_m^{f}(i)\right) \in M_i$ и согласно определению M_i должно быть

$$S\left(h_1^{f'}(i),\ ...,\ h_m^{f'}(i)\right) \leqslant S\left(h_1^{f}(i),\ ...,\ h_m^{f}(i)\right).$$

Противоречие. Лемма доказана.

Установим признак принадлежности набора $(h_1, ..., h_m) \in H_i$ множеству M_i.

Определение. Обозначим $\delta(h_1, \ldots, h_m) = \max_{i,\, j = 1, ..., m}|h_i - h_j| = \max_{i = 1, ..., m} h_i - \min_{i = 1, ..., m} h_i$ — "отклонение" набора $(h_1, ..., h_m)$.

Определение. Набор $(h_1, ..., h_m) \in H_i$ назовем равномерным, если $\delta(h_1, ..., h_m) < 2$. Множество равномерных наборов будем обозначать как E_i.

Утверждение. Если $(h_1, ..., h_m) \in E_i$, то $h_1, ..., h_m$ могут принимать только лишь два значения: $\min(h_1, ..., h_m)$ и, может быть, $\min(h_1, ..., h_m) + 1$.

Это очевидно из того, что эти числа не могут отличаться между собой больше чем на 1.

Утверждение. Если $(h_1, ..., h_m) \in E_i$, то среди m чисел $h_1, ..., h_m a$ принимают значение $\min(h_1, ..., h_m) + 1$, при этом $a = i \bmod m$, $\min(h_1, ..., h_m) = i/m$ (косой чертой обозначено целочисленное деление), $i = h_1 + ... + h_m = a(\min(h_1, ..., h_m) + 1) + (m - a)\min(h_1, ..., h_m) = a + m \min(h_1, ..., h_m)$. Далее искомое утверждение непосредственно следует из свойств деления с остатком.

Утверждение. Пусть $(h_1, ..., h_m)$, $(h'_1, ..., h'_m) \in E_i$. Тогда $S(h_1, ..., h_m) = S(h'_1, ..., h'_m)$.

Действительно, состав набора $(h_1, ..., h_m) \in E_i$ согласно предыдущему утверждению определен однозначно, а значение функции S зависит только от состава набора.

Определение. $mx(h_1, ..., h_m)$ — количество чисел, принимающих значение $\max(h_1, ..., h_m)$ среди $h_1, ..., h_m$.

Определение. $mn(h_1, ..., h_m)$ — количество чисел, принимающих значение $\min(h_1, ..., h_m)$ среди $h_1, ..., h_m$.

Лемма. Пусть $(h_1, ..., h_m) \in H_i$, $\delta(h_1, ..., h_m) \geqslant 2$, $mx(h_1, ..., h_m) > 1$, $mn(h_1, ..., h_m) > 1$.

Тогда найдется такой набор $(h'_1, ..., h'_m) \in H_i$, что

$$\delta(h'_1, ..., h'_m) = \delta(h_1, ..., h_m),$$

$$mx(h'_1, ..., h'_m) = mx(h_1, ..., h_m) - 1,$$

$$mn(h'_1, ..., h'_m) = mn(h_1, ..., h_m) - 1,$$

$$S\left(h'_1, ..., h'_m\right) \geqslant S\left(h_1, ..., h_m\right).$$

1. Выберем $h_{k_1} = \min(h_1, ..., h_m)$, $h_{k_2} = \max(h_1, ..., h_m)$. По условию $h_{k_2} - h_{k_1} \geqslant 2$.

2. Положим $h'_{k_1} = h_{k_1} + 1$, $h'_{k_2} = h_{k_2} - 1$, $h'_j = h_j$ при $j \neq k_1, k_2$. Тогда $(h'_1, ..., h'_m) \in E_i$, так как $h'_1 + ... + h'_m = h_1 + ... + h_m = i$.

3. h_{k_1}, h_{k_2} — не единственные минимальное и максимальное значения соответственно, поэтому $\delta(h'_1, ..., h'_m) = \delta(h_1, ..., h_m)$.

4. Очевидно также, что число минимальных и максимальных значений было уменьшено на 1:

$$mx(h'_1, ..., h'_m) = mx(h_1, ..., h_m) - 1,$$

$$mn(h'_1, ..., h'_m) = mn(h_1, ..., h_m) - 1.$$

5. $S(h'_1, ..., h'_m) = S(..., h'_{k_1}, ..., h'_{k_2}, ...) =$

$= S(..., h_{k_1} + 1, ..., h_{k_2} - 1, ...) \geqslant S(..., h_{k_1}, ..., h_{k_2}, ...) =$

$= S(h_1, ..., h_m)$,

так как $h_{k_1} \leqslant h_{k_2} - 1$. Лемма доказана.

Следствие. Пусть $(h_1, ..., h_m) \in H_i$, $\delta(h_1, ..., h_m) \geqslant 2$, тогда найдется такой набор $(h'_1, ..., h'_m) \in H_i$, что $\delta(h'_1, ..., h'_m) = \delta(h_1, ..., h_m)$, $S(h'_1, ..., h'_m) \geqslant S(h_1, ..., h_m)$, $mx(h_1, ..., h_m) = 1$ или $mn(h_1, ..., h_m) = 1$.

По индукции применим лемму нужное число раз.

Лемма. Пусть $(h_1, ..., h_m) \in H_i$, $\delta(h_1, ..., h_m) \geqslant 2$, тогда найдется такой набор $(h'_1, ..., h'_m) \in H_i$, что $\delta(h'_1, ..., h'_m) < \delta(h_1, ..., h_m)$, $S(h'_1, ..., h'_m) \geqslant S(h_1, ..., h_m)$.

1. По предыдущему утверждению достаточно только рассмотреть случай, когда $mx(h_1, ..., h_m) = 1$ или $mn(h_1, ..., h_m) = 1$.

2. Аналогично вычитаем 1 из максимума и прибавляем 1 к минимуму. Так как минимум или максимум только один, а также максимум больше минимума более, чем на 1, то это уменьшает значение δ. Лемма доказана.

Следствие. Пусть $(h_1, ..., h_m) \in H_i$, тогда найдется такой набор $(h'_1, ..., h'_m) \in E_i$, что $S(h'_1, ..., h'_m) \geqslant S(h_1, ..., h_m)$. Будем применять лемму по индукции. При каждом применении δ уменьшается. Следовательно, когда-нибудь будет достигнуто $\delta < 2$.

Заключительные шаги.

Лемма. $E_i \in M_i$.

1. Пусть $(h_1, ..., h_m) \in M_i \in H_i$. По предыдущему утверждению найдется такое $(h'_1, ..., h'_m) \in E_i$, что $S(h'_1, ..., h'_m) \geqslant S(h'_1, ..., h'_m)$.

2. Однако согласно определению множеств M_i как множеств, на которых S принимает максимальное значение, это означает $S(h'_1, ..., h'_m) = S(h_1, ..., h_m)$, $(h'_1, ..., h'_m) \in M_i$.

3. Пусть некое $(h''_1, ..., h''_m) \in E_i$. Тогда по ранее доказанному $S(h'_1, ..., h'_m) = S(h''_1, ..., h''_m)$, а значит, так как $(h'_1, ..., h'_m) \in M_i$, то и $(h''_1, ..., h''_m) \in M_i$. Лемма доказана.

Утверждение. Для стратегии циклической передачи для всех i $\left(h_1^c(i), ..., h_m^c(i)\right) \in E_i$.

1. Действительно,

$$h_k^c(i) = \left|\left\{j \mid j \leqslant i,\ (j-1) \bmod m + 1 = k\right\}\right| =$$

$$= \left|\{k,\ k + m,\ k + 2m ... \leqslant i\}\right|.$$

2. Пусть для каких-то k_1, k_2 $h^c_{k_2} - h^c_{k_1} \geqslant 2$. Тогда найдется такое r, что $k_1 + rm \leqslant i$, $k_1 + (r + 1)m > i$, но $k_2 + (r + 2)m \leqslant i$.

3. Откуда следует $k_2 + (r + 2)m < k_1 + (r + 1)m \Rightarrow$ $\Rightarrow k_2 + m < k_1$. Чего не может быть, так как k_1, $k_2 = 1, ..., m$. Утверждение доказано.

Сформулируем в рамках данного раздела результат.

Теорема. Стратегия циклической передачи c неулучшаема.

Действительно, для всех i $\left(h^c_1(i), \ ..., \ h^c_m(i)\right) \in E_i \subset M_i$, а этого, согласно доказанному ранее, достаточно для неулучшаемости.

Аналогично можно дать определение среднему числу передач блока (до момента, когда каждый из m блоков будет принят) при использовании стратегии f.

Определение. L_f — случайная величина, имеющая значение i, если произошло событие C^f_i (а следовательно, произошли события C^f_{i+1}, C^f_{i+2}, ...) и не произошло событие C^f_{i-1}. Таким образом, $P\left(L_f = i\right) = p^f_i - p^f_{i-1}$ при $i > 1$, $P(L_f = 1) = p^f_1$.

Утверждение. Для любой стратегии f верно $EL_f \geqslant EL_c$.

1. $P(L_f \geqslant s) = 1 - (P(L_f = 1) + P(L_f = 2) + ...$ $+ P(L_f = s - 1)) = 1 - p_{s-1}$.

2. Далее используем формулу $EL = \sum_{s=1}^{\infty} P(L \geqslant s)$

(справедливую для любой случайной величины L). Утверждение доказано.

6. Инструментальные средства реализации модели в операционных системах *nix

Как отмечалось ранее, согласно принятой модели в передаче информации по скрытому каналу участвуют процесс-передатчик и процесс-приемник. Процесс-передатчик принимает на вход имя передаваемого файла, после чего осуществляет его поблочную передачу. Передача информации происходит побайтно посредством изменения объема памяти, выделенного процессу-передатчику, который считывает процессом-приемником значения этого объема. Процесс-приемник также получает на вход имя файла, периодически считывает характеристики передатчика и записывает результат передачи.

В процессе-передатчике используется функция C++ *sbrk*, изменяющая объем выделенной процессу памяти. Отметим, что функции *brk/sbrk* в языке C++ — это имеющие определенную специфику устаревшие функции, реализующие системные вызовы управления памятью, определяющие адрес окончания ее сегмента. В настоящее время они, как правило, не используются, вместо них используются функции *malloc* и *free*. Однако поведение функции *malloc* неоднозначно, она выделяет память, которая необязательно отображается извне как действительно занимаемая. С этих позиций функция *sbrk* в отношении виртуального объема работает однозначно. Упомянем также, что реализация функций *malloc/free* использует именно вызовы *brk/sbrk*. Параметры процесса считываются с помощью псевдофайловой системы */proc*. Для запущенного процесса-передатчика с идентификатором *pid* в директории */proc* операционных систем *nix есть файл */proc/pid/status*, в котором имеется поле *VmData*. В этом поле записано значение объема виртуальной памяти, выделенной процессу.

В контексте обсуждаемой модели предполагается, что именно поле *VmData* процесс-приемник будет читать с соответствующей периодичностью. Использование манипуляции именно этим параметром имеет то преимущество, что значение *VmData* не видно (по умолчанию) как вывод утилит, предоставляющих информацию о запущенных в системе процессах. К их числу относятся диспетчер задач в Windows, системный монитор и др.

Исходный код программной реализации передатчика и приемника размещен в github-репозитории https://github.com/ibkazakov/article3. Сделаем необходимые комментарии к представленному в этом репозитории программному коду. Прежде всего, отметим, что в нашем случае имеются 256 основных состояний процесса-передатчика при нумерации от 1 до 256, что соответствует одному байту. При этом k-му состоянию соответствует $k \times 1$ МБайт (данные параметры могут быть выбраны и иными) выделенной памяти. Такая память выделяется дополнительно, сверх занятой самим процессом. Таким образом, переход от k-го состояния к l-му выполняется посредством вызова *sbrk*$(1024(l - k))$. Будем считать, что если процесс-передатчик находится в k-м состоянии, то занято k блоков памяти. Максимальное число занимаемых блоков памяти (capacity) далее будем обозначать как c.

После получения имени файла в процессе-передатчике создается объект класса *MemoryManipulato*r, внутри метода *increase* которого и происходит вызов *sbrk*. На основе *increase* последовательно написаны методы *set, transmit, transmit_int*.

Далее восходим к методам *send_block* и *blocks_sender*, реализованным уже не в самом *MemoryManipulator*, а принимающим как один из своих параметров указатель на объект данного класса. Эти методы реализованы в соответствии с протоколом, который будет описан далее.

В исходном коде процесса-приемника, прежде всего, есть метод *getVmData*, считывающий соответствующее поле из */proc/pid/status*. На его основе реализованы *get_one_value* и *accept_block*.

В течение передачи процесс-передатчик будет циклически печатать номер очередного отправленного блока. Процесс-приемник выводит на консоль номер полученного блока и его состояние, отражающее, успешно/неуспешно/уже был принят блок ранее. Успешно принятый блок сохраняется во временном файле. После того, как все блоки оказываются успешно принятыми, приемник собирает файл-результат из временных файлов.

7. Особенности модели скрытого канала передачи

Периодически считывая содержимое поля *VmData*, процесс-приемник получает последовательность объемов занятой процессом-передатчиком памяти, относящихся к моментам чтения содержимого поля. Каждый из считанных объемов можно считать самостоятельным сообщением, однако при данном предположении неизбежно возникают следующие вопросы и затруднения, требующие разрешения.

Во-первых, считываемое количество занятой памяти равно $1024k + x$, где x — это какое-то неизвестное для процесса-приемника количество, занимаемое самим процессом-передатчиком. Можно предполагать, что это количество меньше, чем 1 Мбайт, и использовать для декодирования отбрасывание дробной части от $(1024k + x)/1024$. Однако это ничем не обосновано, так как действительный остаток может быть любым.

Во-вторых, возникают значительные трудности синхронизации времени записи и времени чтения. Для правильной передачи данных в промежутке между двумя записями должно произойти одно и только одно чтение. Если чтения не произойдет, то получится пропуск значения. Если произойдет два чтения, то ранее принятое значение повторится в последовательности. По этой причине скорость чтения должна строго соответствовать скорости записи, а не просто превышать ее, что неосуществимо на практике.

Для разрешения перечисленных выше вопросов будем отслеживать не непосредственно считанный объем памяти, а его изменение. Таким образом, считается, что информация была передана тогда и только тогда, когда текущий считанный объем отличается от предыдущего. Если объем увеличился на 1 Мбайт, считается, что было передано значение 0, если он увеличился на 2 Майт, то передано значение 1 и т. д. Далее опишем эту модель более строго.

Пусть $X = \{0, 1, 2, ..., c - 1\}$ — множество передаваемых значений, $Y = \{-c, ..., -1, 1, ..., c\}$ — множество возможных изменений числа занятых блоков. Пусть в данный момент занято A блоков и требуется передать значение $s \in X$.

Замечание. Здесь считается, что остаток от деления mod определен только на положительных значениях.

Схема кодирования/декодирования выглядит следующим образом.

Кодирование: переводим передатчик в состояние $(A + s + 1) \bmod(c + 1)$, т. е. передаем разность $(A + s + 1)\bmod(c + 1) - A$. Декодирование: пусть получена некая разность $d \in Y$, тогда считаем, что принято значение $(d + c + 1)\bmod(c + 1) - 1$ ($-c \leqslant d \leqslant c$, $1 \leqslant d + c + 1 \leqslant 2c + 1$, в соответствии с замечанием).

Замечание. Данным формальным определениям можно дать очень простую содержательную интерпретацию. Представим себе, что есть некий диск с $c + 1$ делениями от 0 до c, вращая который, можно по циклу изменять состояние передатчика. Соответственно, у него есть два направления вращения — положительное $0 \to 1 \to 2 \to ... \to c \to 0$ и отрицательное $0 \to c \to ... \to 1 \to 0$. Отправить значение s означает повернуть диск на $s + 1$ делений в положительном направлении. Приемник в момент изменения состояния передатчика получает на свой вход его новое состояние. Сравнивая новое со старым, приемник по формуле для декодирования рассчитывает, на сколько делений нужно повернуть диск в положительном направлении для преобразования старого состояния в новое, тем самым восстанавливая значение s.

Относительно представленной схемы кодирования рассмотрим следующую теорему.

Теорема. Схема кодирования является корректной, а именно: декодирование действительно обратно кодированию.

Доказательство. Предварительно докажем вспомогательные утверждения.

Утверждение. Полученная в результате кодирования разность принадлежит множеству Y. Предположим, что $(A + s + 1)\bmod(c + 1) \equiv A$. Тогда $A \equiv A + s + 1 \bmod(c + 1)$, $s + 1 \equiv 0 \bmod(c + 1)$. Однако $s \in X$, следовательно, $0 < s + 1 < c + 1$. Противоречие показывает, что эта разность не равна 0.

Вместе с тем, $0 \leqslant (A + s + 1)\bmod(c + 1) \leqslant c$, $-c \leqslant -A \leqslant 0$. Откуда следует, что $-c \leqslant (A + s + 1)\bmod(c + 1) - A \leqslant c$.

Представленные отношения показывают, что искомое утверждение справедливо.

Утверждение. Полученное в результате декодирования значение принадлежит множеству X. При этом $0 \leqslant (d + c + 1) \bmod(c + 1) \leqslant c$, $-1 \leqslant (d + c + 1)\bmod(c + 1) - 1 \leqslant c - 1$.

Доказательство.

Предположим, что $(d + c + 1)\bmod(c + 1) - 1 = -1$, $(d + c + 1)\bmod(c + 1) = 0$.

Такое возможно лишь при $d = t(c + 1)$, $t \in Z$, где Z — множество целых чисел, а t — произвольное целое число. Однако среди множества Y таких значений нет. Следовательно, $0 \leqslant (d + c + 1) \bmod(c + 1) - 1 \leqslant c - 1$. Утверждение доказано.

Окончание доказательства теоремы (последовательное применение сначала кодирования, а затем декодирования):

$(d + c + 1)\bmod(c + 1) - 1 = (((A + s + 1) \bmod(c + 1) - A + c + 1\bmod(c + 1) - 1 = (((A + s + 1) \bmod(c + 1) + (c + 1 - A))\bmod(c + 1) - 1 = (A + s + 1 + c + 1 - A) + \bmod(c + 1) - 1 = (s + 1) \bmod(c + 1) - 1 = s + 1 - 1 = s$, так как $(x\bmod z + y)\bmod z = (x\bmod z \bmod z + y \bmod z)\bmod z = (x\bmod z + y\bmod z)\bmod z = (x + y) \bmod z$.

А также $A \leqslant c$, $c + 1 - A > 0$ и $0 < s + 1 < c + 1$.

Здесь полагается $d(s) = ((A + s + 1)\bmod(c + 1)) - A$ соответственно вышеопределенной схеме кодирования, а затем согласно схеме декодирования вычисляется $(d + c + 1) \bmod(c + 1) - 1$. Тот факт, что данное выражение оказывается равно исходному s, и означает, что декодирование в точности восстанавливает закодированное значение.

Теорема доказана.

8. Ошибки при передаче информации

При использовании представленной выше схемы кодирования данных, если между двумя последовательными записями было два или более чтения, то ошибки дублирования не произойдет. Дело в том, что между двумя последовательными записями объем памяти, занимаемый процессом-передатчиком, остается по определению неизменным. Также, так как алгоритм декодирования вычисляет именно разность между объемами памяти, сокращается неизвестный процессу-приемнику объем памяти, который процесс-передатчик занимает сам. Таким образом, условием корректной передачи данных является превышение скорости чтения над скоростью записи. В частности, если промежуток времени между двумя чтениями меньше промежутка между двумя записями, то ошибка вообще не произойдет.

Однако ошибка может произойти, если в промежутке между двумя записями не произошло ни одного чтения. В данном случае ошибка не сводится к простому пропуску значения, а является "склеиванием" двух значений. Например, $(+1, +1) \rightarrow (+2)$, $(+1, -3) \rightarrow (-2)$. В очень редких случаях может быть вариант $(+1, -1) \rightarrow 0$, т. е. пропуск обоих значений. Если чтения не было на двух и более соседних интервалах между записями, то могут "склеиться" три и более значения, например $(+1, +1, +1) \rightarrow (+3)$.

При возникновении ошибки значения всегда только склеиваются. Это означает, что число значений, полученных процессом-приемником, становится меньше числа значений, отправленных процессом-передатчиком. Следовательно, если процесс-приемник знает заранее, сколько значений должно быть передано, он сможет распознать сам факт возникновения ошибки, а именно ошибка произошла, если и только если число полученных значений меньше числа переданных. Поэтому в представленной модели передачи данных нет необходимости в механизмах проверки корректности передачи наподобие контрольных сумм. Однако нужно как-то сообщить процессу-приемнику, сколько же значений должно быть передано.

Известны коды, исправляющие пропуски [7]. Однако так как в представленной схеме кодирования значения не "выпадают", а "склеиваются", то к непосредственному употреблению эти коды непригодны. В дальнейшем необходимо исследовать построение кодов, исправляющих склейку и построенных по аналогии с кодами Левенштейна.

9. Протокол поблочной передачи

Вероятность передать некий массив данных (например, файл) без ошибок убывает от размера этого массива экспоненциально, т. е. $p = P^{-h}$. Предполагаем, что события типа "произошла ошибка" для разных мест массива являются независимыми. Можно разделить массив на n частей. При этом если вероятность передать часть безошибочно равна p, то вероятность передать безошибочно их все согласно предположению о независимости равна p^n. Следовательно, при большом размере массива эта вероятность быстро стремится к 0.

В целях осуществления безошибочной передачи большого массива данных его можно разбить на небольшие массивы-блоки, которые можно передать с большой вероятностью безошибочно. Блоки следует передавать в циклическом порядке: сначала 1-й блок, потом 2-й блок и так до последнего, а потом снова 1-й блок и т. д. Передачу следует вести до тех пор, пока каждый из блоков не будет передан безошибочно.

Для реализации этой идеи на практике добавим к уже имеющимся значениям 0...255 еще два дополнительных значения 256 и 257, означающих начало и конец блока соответственно. Процесс-передатчик работает постоянно, и, следовательно, процессу-приемнику нужно отловить момент начала передачи очередного блока. Процессу-приемнику необходимо также определять момент конца передачи блока. После завершения приема проводится сверка заявленного числа передаваемых значений с числом фактически принятых.

Как было уже отмечено ранее, если промежуток времени между двумя записями достаточно велик для того, чтобы в нем гарантированно произошло хотя бы одно чтение, то ошибка не возникнет. Поэтому до и после передачи значений 256, 257 процесс-передатчик выжидает большой промежуток времени *BIG_INTERVAL*. Промежутки времени между записями обычных значений 0...255 будут далее называться малыми и обозначаться как *SMALL_INTERVAL*.

В начале каждого блока с промежутками *BIG_INTERVAL* между записями передаются: общее число блоков; номер данного блока; размер блока. На каждое из этих полей отводится 4 значения; всего 12 значений, разделяемых большим интервалом.

Процесс-приемник гарантированно считывает заголовок без ошибок, создает временный файл (если он еще не существует), который имеет имя, соответствующее номеру блока, и считывает в него тело блока. Если было считано менее заявленного размера блока, то блок считается переданным ошибочно и временный файл удаляется. Как только число сохраненных временных файлов становится равным числу блоков, считывание с процесса-передатчика прекращается, а записанные файлы по порядку их имен склеиваются в файл-результат.

10. Методика измерения вероятности ошибок

Как ранее было отмечено, если интервал времени между двумя записями больше, чем интервал времени между двумя чтениями, то ошибка невозможна. Очевидно, если интервал между записями более чем вдвое меньше, чем интервал между чтениями, то ошибка неизбежна на каждом чтении.

Пусть между некими двумя последовательными (далее называемыми 1-м и 2-м чтением) чтениями произошла всего одна запись. Эта запись делит интервал на две части: между 1-м чтением и записью и между записью и 2-м чтением. Так как внутри данных частей записей не происходило, то каждый из этих интервалов меньше, чем интервал между записями. Следовательно, исходный интервал между чтениями, будучи суммой этих двух частей, больше, чем удвоенный интервал между записями.

Как следствие изложенного выше, в данных пределах отношения между интервалами чтения и записи частота ошибок варьируется от их неизбежности до их невозможности. Точный вид зависимости установить теоретически затруднительно, и, следовательно, возникает задача экспериментального измерения частоты ошибок.

Измерение будет осуществляться следующим образом. Во-первых, со стороны процесса-передатчика отправляется значение 0, что в разностном коде означает переход от состояния k к состоянию $(k + 1)\mathrm{mod}c$. То есть состояния процесса-передатчика сменяются циклически $0 \to 1 \to 2 \to ... \ 255 \to 0$.

Во-вторых, процесс-приемник (специально упрощенный для целей измерения вероятности ошибок, см. исходный код) считывает передаваемые разности. Если ошибки не произошло, то процесс-приемник считывает, как и ожидается, значение 0. Если же между двумя последовательными чтениями произошло n ошибок (т. е. между чтениями произошло $n + 1$ записей), то в момент 1-го чтения было считано значение A, а в момент 2-го — значение $(A + n + 1)\mathrm{mod}c$. В итоге принята разность $d = (A + n + 1)\mathrm{mod}(c + 1) - A$. Следовательно, по формуле декодирования было принято значение $x = (d + c + 1)\mathrm{mod}(c + 1) - 1$. Если $A + n + 1 < c + 1$, то $x = (A + n + 1 - A)\mathrm{mod}(c + 1) - 1 = n$. В случае $A + n + 1 \geqslant c + 1$ также $x = (A + n + 1 - (c + 1) - A\mathrm{mod}(c + 1) - 1 = n$. Таким образом, в конечном итоге принимаем состояние с номером, равным числу произошедших ошибок.

Замечание. Принимается то упрощение, что не могут произойти 256 и более ошибок подряд — это оправданно в силу предельно малой вероятности подобного. Более точно, если скорость записи такова, что это все же может произойти, то реализация модели уже находится в зоне неизбежности ошибок, где измерение вероятности вообще не имеет смысла.

В процессе-приемнике создадим два значения-накопителя: число отправленных значений (*All*) и число произошедших ошибок (*Err*). Пусть было принято значение n. Тогда, согласно ранее принятой логике, произошло $n + 1$ записей ($All = All + n + 1$) и n ошибок ($Err = Err + n$).

В-третьих, каждые *length* отправленных значений будет фиксироваться изменение общего числа произошедших ошибок, тем самым определяя число ошибок в блоке длины *length*. Между блоками для удобства будем передавать дополнительное разделительное значение, с обеих сторон отграниченное большим временным интервалом *BIG_INTERVAL*.

Таблица 4

Число блоков с определенным числом ошибок (эксперимент)

SMALL_INTERVAL, мкс	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
50	2840	3572	2253	944	298	74	16	3
75	5767	3174	875	160	22	2	0	0
100	7767	1963	248	21	1	0	0	0
125	7604	2083	285	26	2	0	0	0
150	9492	495	13	0	0	0	0	0

В итоге процесс-приемник принимает определенное число блоков длины *length* и подсчитывает числа блоков с 0, 1, 2, 3... ошибками.

Далее будем обозначать через A общее число блоков, через C_i — число блоков с i ошибками. $A = C_0 + C_1 + C_2 + ...$, $\Delta = 0 \times C_0 + 1 \times C_1 + 2 \times C_2 + ...$ — общее число ошибок. $p_k = \dfrac{C_k}{A}$ — эмпирическая вероятность допустить k ошибок за блок.

В заключение приведем экспериментальные данные (табл. 4). Варьируется *SMALL_INTERVAL* между записями, чтение происходит настолько быстро, насколько это возможно. Число принятых блоков — 10 000. Размер блока — 1024.

Данные табл. 4 свидетельствую о том, что подавляющее большинство блоков, принятых с ошибкой, содержит одну или две ошибки. Вероятность получить в блоке три ошибки быстро стремится к нулю при увеличении промежутка *SMALL_INTERVAL*. Еще быстрее вероятности стремятся к нулю при дальнейшем увеличении числа ошибок. Сравнение с теоретическими значениями будет представлено далее.

11. Распределение случайной величины числа ошибок в блоке

Экспериментальные данные, представленные в табл. 4, нуждаются в интерпретации. Необходимо исходя из априорных соображений определить вероятность совершения n ошибок в блоке. Нужно сравнить эти априорные вероятности с эмпирическими, т. е. проверить, соответствуют ли экспериментальные данные теоретически полученному вероятностному распределению.

В первую очередь заметим, что блок обладает свойством делимости на меньшие блоки, а также что общее число ошибок в нем равно сумме числа ошибок в подблоках, на которые его можно разделить. При этом случайные величины, выражающие число ошибок в подблоках, являются независимыми. Однако эта делимость имеет предел, и блок можно представить разделенным на N самых малых частей

(элементарных блоков), в каждой из которых с малой вероятностью f есть одна ошибка, а с вероятностью $1 - f$ ошибки нет. Этот самый малый подблок естественно соответствует двум последовательным чтениям, поэтому можно принять $N = 1024$.

Сформулируем это формально. Пусть S — случайная величина, равная числу ошибок в блоке, $\xi_1, ..., \xi_N$ — независимые случайные величины, принимающие два значения: $\xi = 0$, если в элементарном блоке не произошло ошибки, и $\xi = 1$, если в нем ошибка произошла. Причем $P(\xi = 1) = f$, $P(\xi = 0) = 1 - f$. Тогда $S = \xi_1 + \xi_2 + ... + \xi_N$.

Определение. Выражение

$$p_k = P(S = k) = \binom{k}{N-k} f^k (1-f)^{N-k} —$$

теоретическая вероятность допустить k ошибок за блок.

Однако в таком виде данная приближенная формула еще не пригодна к употреблению, так как в ее состав входит эмпирически ненаблюдаемый параметр f. Таким образом, нужно свести его к эмпирически наблюдаемым, т. е. тем, которые мы можем высчитать непосредственно из экспериментальной таблицы распределения вероятностей, например к математическому ожиданию.

Проведем вычисление: $ES = E\xi_1 + E\xi_2 + ... + E\xi_N = NE\xi_1 = Nf$, т. е. $f = \frac{ES}{N}$.

Приведем табл. 5 результатов расчета, по которой можно судить, насколько удачны представленные выше теоретические рассуждения. Ее столбцы: *SMALL_INTERVAL*, матожидание *ES*, а также теоретические вероятности, умноженные на общее число блоков $C^* = p_i^* A$, соотвествующие четырем столбцам C_0, C_1, C_2, C_3 из табл. 4.

Из табл. 5 следует, что экспериментальные данные хорошо сходятся с результатом теоретического расчета.

Таблица 5

Число блоков с определенным числом ошибок (теория)

SMALL_ INTERVAL, мкс	*ES*	C_0^*	C_1^*	C_2^*	C_3^*
50	1,2589	2837,46	3572,08	2251,79	944,24
100	0,5502	5767,49	3173,27	873,05	159,89
150	0,2526	7767,54	1962,08	247,69	20,82
200	0,2739	7603,80	2082,68	285,10	25,99
250	0,0521	9492,37	494,55	12,87	0,22

Заключение

В работе описаны модель и программная реализация скрытого канала, основанного на модуляции объема занимаемой памяти. Процесс-передатчик меняет свой объем в зависимости от сообщения, закодированного с помощью специально введенного разностного кода. Согласно принятой модели процесс-приемник периодически считывает этот объем и определяет очередной символ сообщения. В том случае, если между двумя записями не происходит чтения, пара значений склеивается. Исправление таких ошибок обеспечивается многократной передачей каждого символа. В рамках дальнейших исследований планируется построение самокорректирующихся кодов, соответствующих введенной модели ошибки.

Рассматривается задача передачи сообщения из m, $m \in \mathbb{N}$, блоков, с вероятностью успешной передачи p, причем успешности передачи различных блоков независимы. В результате возникает случайная величина, задающая число переданных блоков, обеспечивающее успешную передачу всего сообщения. Оказывается, что оптимальной стратегией является циклическая передача. Для этого случая изучены математическое ожидание и дисперсия, а также объем, необходимый и достаточный для передачи сообщения с вероятностью не ниже $1 - \varepsilon$ для $0 < \varepsilon < 1$.

Адекватность принятой математической модели проверена с помощью эксперимента, проведенного с использованием программной реализации скрытого канала.

Список литературы

1. **Lampson B. W.** A note on the confinement problem // Communications of ACM. 1973. Vol. 16, N. 10. P. 613—615.

2. **Zander S., Armitage G., Branch P.** A Survey of Covert Channels and Countermeasures in Computer Network Protocols // IEEE Communications Surveys & Tutorials. 2007. Vol. 9, N 3. P. 44—57.

3. **Cox I.** Digital Watermarking and Steganography 2nd edition. San Francisco: Morgan Kaufmann Publishers Inc., 2008. 295 p.

4. **Казаков И. Б.** Кодирование в скрытом канале перестановки пакетов // Программная инженерия. 2018. Т. 9, № 4. С. 163—173.

5. **Казаков И. Б.** Структура графа на множестве перестановок S_n, задаваемая моделью ошибки в скрытом канале перестановки пакетов // Интеллектуальные системы. Теория и приложения. 2018. Т. 22, № 2. С. 53—79.

6. **Conway J. H., Guy R. K.** The Euler-Mascheroni Number // The Book of Numbers. New York: Springer-Verlag, 1996. P. 260—261.

7. **Левенштейн В. И.** Двоичные коды с исправлением выпадений, вставок и замещений символов // Докл. АН СССР. 1965, Т. 163, № 4. С. 845—848.

Difference Code and a Protocol for Cyclic Blockwise Transmission in a Memory-Based Covert Channel

I. B. Kazakov, i_b_kazakov@mail.ru, Lomonosov Moscow State University, Moscow, 119234, Russian Federation

Corresponding author:

Kazakov Ilia B., Postgraduate Student, Lomonosov Moscow State University, Moscow, 119234, Russian Federation, E-mail: i_b_kazakov@mail.ru

We study a covert channel based on manipulation of the volume of memory occupied by a transmitting process. The channel includes errors that lead to gluing several values into one. Channel reliability is provided by multiple transmission. We propose a strategy that minimizes the number of blocks transmitted under the given level of reliability and present reliability estimations. Model adequacy is verified through an experiment performed using a software implementation of the channel.

Keywords: *covert channels, throughput, difference code, multiple transmission, probability of error, virtual memory*

For citation:

Kazakov I. B. Difference Code and a Protocol for Cyclic Blockwise Transmission in a Memory-Based Covert Channel, *Programmnaya Ingeneria,* 2019, vol. 10, no. 5, pp. 204—218.

DOI: 10.17587/prin.10.204-218

References

1. **Lampson B. W.** A note on the confinement problem, *Communications of ACM,* 1973, vol. 16, no. 10, pp. 613—615.

2. **Zander S., Armitage G., Branch P.** A Survey of Covert Channels and Countermeasures in Computer Network Protocols, *IEEE Communications Surveys & Tutorials*, 2007, vol. 9, no. 3, pp. 44—57.

3. **Cox I.** *Digital Watermarking and Steganography*, 2nd edition. San Francisco, Morgan Kaufmann Publishers Inc., 2008, 295 p.

4. **Kazakov I. B.** Kodirovanie v skrytom kanale perestanovki paketov (Coding in a Covert Channel of Data Packages' Permutations), *Programmnaya Ingeneria*, 2018, vol. 9, no. 4, pp. 163—173 (in Russian).

5. **Kazakov I. B.** Struktura grafa na mnojestve perestanovok S_n, zadavaemaya modelju oshibki v skrytom kanale perestanovki paketov (The structure of the graph induced on the set of permutations S_n by an error model in a covert channel based on permutation of packets), *Intellektualnye sistemy. Teoria i prilojeniya*, 2018, vol. 22, no. 2, pp. 53—79 (in Russian).

6. **Conway J. H., Guy R. K.** The Euler-Mascheroni Number, *The Book of Numbers,* New York, Springer-Verlag, 1996, pp. 260—261.

7. **Levenshtein V. I.** Dvoichnye kody s ispravleniem vypadenij, vstavok i zameshchenij simvolov (Binary codes capable of correcting deletions, insertions, and reversals), *Dokl. Akad. Nauk SSSR*, 1965, vol. 163, no. 4, pp. 845—848 (in Russian).

ИНФОРМАЦИЯ

УДК 004.891.2 DOI: 10.17587/prin.10.219-225

М. Ю. Щелыкалин, и. о. начальника отдела, e-mail**:** Schelikal@gmail.com,
М. А. Шатский, канд. техн. наук, доц., начальник направления, e-mail: 246@mars-mokb.ru,
М. Ю. Косинский, канд. техн. наук, начальник отдела, e-mail: 246@mars-mokb.ru,
Федеральное государственное унитарное предприятие "Московское опытно-конструкторское бюро "Марс" (МОКБ "Марс")

Анализ результатов испытаний бортового программного обеспечения космического аппарата на основе базы решающих правил

Предложена методика ускорения отработки бортового программного обеспечения. Методика основана на создании баз решающих правил для проверки правильности работы бортового программного обеспечения. Такой подход позволяет уменьшить трудозатраты на анализ результатов моделирования полета космического аппарата.

Методика реализована и апробирована на примере процесса отработки бортового программного обеспечения МОКБ "Марс".

Ключевые слова: *бортовое программное обеспечение, решающие правила, испытания, тестирование, математическое моделирование, система поддержки разработки, система автоматизации тестирования*

Введение

В статье рассматриваются актуальные вопросы повышения качества разработки бортового программного обеспечения (БПО) космических аппаратов (КА). В последнее время к КА предъявляется все большее число требований в части длительности автономной работы, резервирования и взаимодействия с высокоточной научной аппаратурой, при этом постоянно сокращаются сроки, отводимые на разработку. Это существенно влияет на процесс отработки БПО, реализующего логику работы КА. Отработка БПО — это итеративный процесс, включающий ряд этапов автономной и комплексной отладки и испытаний БПО на различных стендах [1, 2].

Для создания БПО, отвечающего требованиям заказчиков, предлагается использовать автоматизированные базы знаний, способные автоматически испытывать создаваемое БПО. Такой подход приводит к изменениям в процессе наземной отработки БПО.

Традиционно БПО обладает иерархической структурой (рис. 1) и включает в себя ряд программных подсистем [3], число которых может достигать десятков. В разработке программных модулей (ПМ) для подсистем участвуют десятки разработчиков. Каждый из модулей проходит этап автономной отладки, после успешного завершения которого модуль признается годным для использования в составе БПО. На основе проверенных модулей специалист, ответственный за комплексирование, создает очередную версию БПО.

Версия БПО проверяется на испытательном стенде. Проверка на стенде осуществляется путем моделирования различных режимов работы КА в соответствии с "Программой и методикой испытаний БПО" (ПиМ). Такая программа составляется на основе технических требований к системе управления КА, записанных в техническом задании (ТЗ). Состоит ПиМ из десятков комплексных вариантов, моделирующих различные режимы работы систем управления (БПО СУ).

Результаты моделирования разработчики ПМ проверяют на соблюдение требований ТЗ. В случае выявления замечаний разработчики корректируют ПМ, отлаживают и передают их для создания новой версии БПО. Новая версия БПО повторно проверяется в объеме всех режимов работы, описанных в ПиМ. Среднее время моделирования одного режима работы КА на испытательном стенде — 8 ч.

Таким образом, осуществляется многократно повторяющийся цикл отработки БПО до полного и безошибочного прохождения всех испытаний в соответствии с ПиМ.

Цели и задачи исследования

Целью исследования, результаты которого представлены в настоящей работе, является уменьшение трудоемкости процесса анализа результатов моделирования полета КА для ускорения отработки БПО.

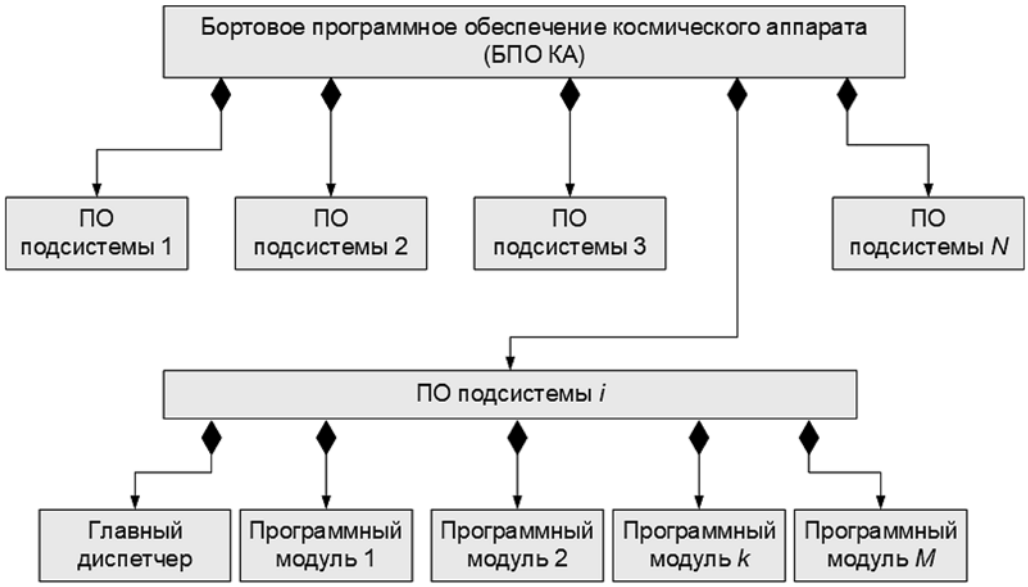

Рис. 1. Пример иерархической структуры БПО [4]

К задачам исследования относятся следующие:

1) анализ процесса отработки БПО для выявления возможных мест, позволяющих уменьшить трудоемкость;

2) создание модернизированной методики отработки БПО;

3) программная реализация баз знаний и их интеграция с существующими информационными системами;

4) внедрение модернизированной методики в рабочий процесс.

Анализ процесса отработки БПО

Рассмотрим типовую схему отработки БПО более подробно (рис. 2) [1]. На основании ТЗ на БПО СУ выделяются программные подсистемы, для которых разрабатываются ПМ. После отладки ПМ на рабочих местах разработчиков из них компонуется версия БПО. Далее версия передается на испытательный стенд. При отработке БПО используется ряд стендов [1], позволяющих проводить математическое и полунатурное моделирование работы БПО. Рассмотрим процесс отработки БПО на аттестационном стенде — автоматизированном цифровом комплексе (АЦК). Особенностями данного стенда являются использование реальной бортовой аппаратуры и работа в реальном масштабе времени.

Организацией отработки на стенде занимается специалист, ответственный за комплексирование и отработку БПО. При этом он руководствуется ПиМ, содержащей перечень моделируемых режимов.

При проведении испытаний ответственный специалист в соответствии с планом испытаний должен организовать моделирование режимов полета КА на стенде. Исходные данные для моделируемых режимов

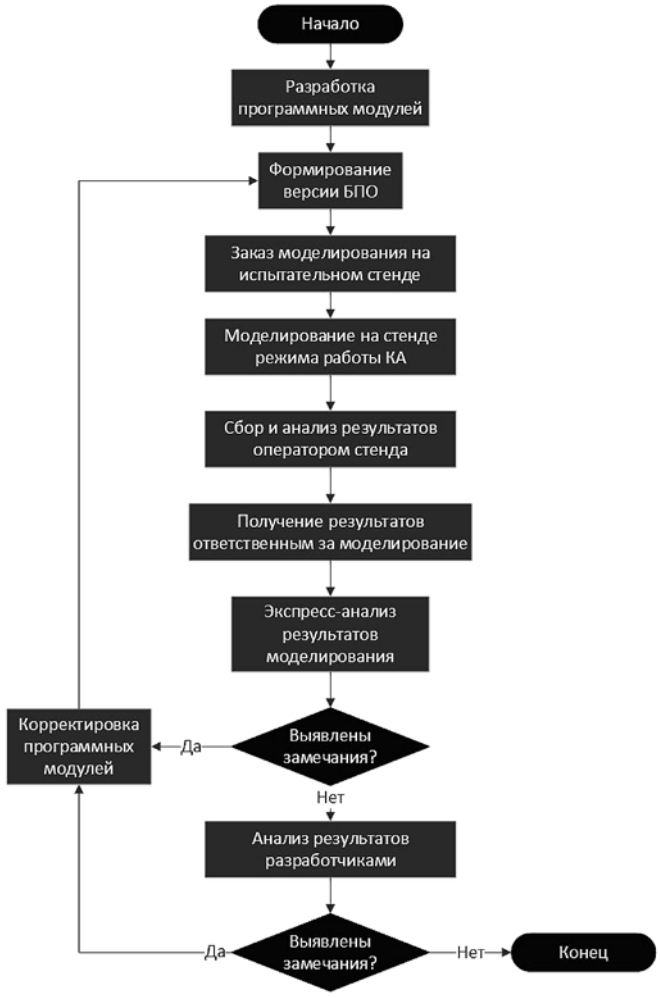

Рис. 2. Методика отработки БПО на испытательном стенде

составляются при помощи системы автоматизации испытаний (САИ) [5, 6]. Эта система представляет собой клиент-серверное приложение, предназначенное для управления процессом отработки БПО. Она хранит в базе данных (PostgreSQL) информацию о существующих версиях БПО, информацию для составления циклограмм (наборы актуальных кодовых команд), составленные циклограммы моделируемых режимов.

После внесения данных о моделируемом режиме в САИ ответственный специалист указывает дату проведения испытания БПО. В свою очередь, САИ автоматически передает данные на стенд.

Для моделирования режима работы КА оператор стенда выгружает из САИ необходимые данные: закодированную в формат стенда циклограмму выдачи команд, файлы начальных условий, информацию о версии БПО. Оператор настраивает стенд, запускает его в работу и контролирует состояние. В результате моделирования формируется следующий набор данных:

- телеметрическая информация КА (ТМИ);
- данные имитационного ПО (параметры движения КА по данным математических моделей);
- протоколы работы имитаторов бортовых устройств.

Оператор стенда загружает указанные данные в САИ, дополняет их результатом анализа работы стенда. Данные с результатами испытаний БПО сжимаются и записываются САИ на ftp-сервер.

Ответственного за отработку специалиста САИ информирует о получении результатов со стендов. Ответственный специалист проводит экспресс-анализ полученных результатов на предмет нормативности отработки циклограммы режима и наличия параметров, сигнализирующих о сбоях.

При выявлении замечаний на этапе экспресс-анализа специалист, ответственный за испытания, определяет подсистему, вызвавшую замечания. Далее для детального анализа результатов задействуют разработчиков ПМ указанной подсистемы.

Если на этапе экспресс-анализа замечаний не выявлено, то специалист, ответственный за испытания, определяет перечень проверяемых в данном испытании подсистем и через САИ передает разработчикам результаты моделирования для более детального анализа. В большинстве случаев для анализа результатов моделирования привлекают большую часть коллектива разработчиков.

Разработчики получают запрос на анализ результатов моделирования через САИ. Система предоставляет разработчикам следующую информацию: описание моделируемого режима; идентификатор версии БПО; название стенда, на котором проводилось моделирование; файлы с результатами моделирования; результаты анализа. Разработчики анализируют результаты работы своих подсистем и записывают в САИ суждение о нормативности их функционирования. В случае обнаружения замечаний разработчики корректируют ПМ, формируют новую версию БПО и повторно проводят отработку всей ПиМ.

Анализ ТМИ разработчиками БПО происходит следующим образом.

1. Проводится первичная обработка ТМИ, содержащейся в двоичном файле, полученном со стенда. В результате получается протокол изменения во времени набора параметров для отдельной подсистемы.

2. Проводится вторичная обработка избранных параметров, при которой рассчитываются дополнительные характеристики, на основе которых становится возможным принять решение о нормативности работы ПМ отдельных подсистем.

Особенностью вторичной обработки ТМИ является использование специализированных программных комплексов для принятия решения о нормативности работы бортовых программ. Данные комплексы называют обработчиками телеметрической информации (ОТМИ). В зависимости от особенностей работы подсистем выбирают различные методы и средства анализа результатов моделирования. В МОКБ "Марс" в настоящее время используют перечисленные далее варианты ОТМИ.

- Автономные ОТМИ, созданные разработчиками ПМ для своих нужд на языках Scilab, Matlab, perl, python, C++, Ruby, C#;
- Автономные средства на базе стандартных офисных пакетов с дополнением макросами.

Обработчики телеметрической информации анализируют количественные и качественные параметры, подразделяющиеся на ряд групп. К их числу относятся:

- реализация циклограммы комплексных режимов;
- параметры ориентации КА;
- параметры ориентации панелей солнечных батарей;
- значения углов приводов антенн;
- точность определения используемых параметров.

Для упрощения анализа ОТМИ содержат в своем составе средства первичной обработки ТМИ. Таким образом, ОТМИ состоят из следующего набора модулей: модули первичной обработки ТМИ, модули графического интерфейса, модули обработки пользовательских шаблонов для выделения параметров из ТМИ, модули вторичной специализированной обработки результатов, модули формирования отчетов. Модули пользовательского интерфейса отвечают за взаимодействие с пользователем, подготовку исходной информации для обработки, настройку вариантов обработки. Данные, которые разработчики получали от САИ, обычно передавались в ОТМИ вручную. Разработчики ПМ проводили настройку режима работы ОТМИ для обработки каждого моделируемого режима. В зависимости от ОТМИ и его настроек формировались следующие данные:

- текстовые протоколы основных признаковых и программных параметров;
- данные для графического отображения;
- отчеты о выполнении программы полета (по этапам, по подсистемам, в целом) и подтверждении требований ТЗ.

Обработчики телеметрической информации представляли собой отдельные приложения, функционирующие независимо. От разработчика требовалось вручную проводить выгрузку ТМИ и параметров моделируемого режима из САИ и внесение их в ОТМИ. При этом получаемые результаты не сохранялись систематизированно, что впоследствии затрудняло анализ группы режимов или сравнение текущих результатов с полученными ранее.

Таким образом, в рамках существующего процесса отработки БПО анализом результатов занимаются несколько групп разработчиков БПО, каждая из которых самостоятельно решает задачу автоматического анализа результатов моделирования, основываясь на собственном опыте. Разработчики выбирают наиболее подходящее представление данных для анализа с учетом доступности методов обработки данных в используемых ими средствах обработки и наличия специалистов, владеющих этими средствами обработки. В результате разрабатывается большая номенклатура ОТМИ с использованием различных средств программирования с разным уровнем автоматизации обработки данных. Получаемые результаты работы ОТМИ имели разнородный характер и зачастую могли быть интерпретированы только разработчиком конкретной подсистемы БПО.

После проведения анализа каждый разработчик БПО должен был передать свое заключение о нормативности (в виде акта) ответственному за испытания, который формировал суммарный акт по результатам испытания.

Модернизация процесса анализа результатов отработки БПО

Для повышения эффективности процесса анализа результатов отработки предлагается его модернизировать в соответствии с методикой, включающей в себя следующие этапы:

1) внедрение в ОТМИ элементов систем принятия решений, в том числе механизмов баз знаний с решающими правилами, способных выдавать заключения типа "норма" или "замечание";

2) создание полного перечня данных, связанных с испытаниями, которые могут быть доступны ОТМИ при автоматическом анализе;

3) создание информационной системы, хранящей в себе необходимые для анализа данные и собственно исполняемые файлы ОТМИ и способной автоматически их запускать и интегрировать результаты их работы;

4) создание единого протокола взаимодействия информационной системы и ОТМИ;

5) внедрение информационной системы в процесс отработки БПО.

Рассмотрим некоторые особенности этой методики подробнее.

Для проведения автоматического анализа результатов отработки БПО необходимо создать набор баз знаний, содержащих опыт анализа результатов испытаний в продукционном представлении в виде решающих правил. Каждая такая база знаний будет позволять формировать заключение по какой-либо подсистеме БПО или частной задаче анализа.

Информационная система должна хранить в себе ОТМИ с базами знаний как подключаемые модули и обращаться к ним при необходимости. После чего информационная система должна интегрировать результаты анализа нескольких задач и формировать оценку по конкретному испытанию. Именно для целей автоматического получения итогового суждения по испытанию следует модернизировать существующие ОТМИ с внедрением механизма баз знаний и реализации функции выдачи типовых заключений "норма" или "замечание". Одним из наиболее распространенных подходов [7] по формализации критериев нормативности прошедшего испытания является использование базы решающих правил типа "если... то... иначе...", что позволяет в понятном виде получить как набор формализованных знаний, так и причину получения суждения о нормативности работы и сформировать отчетную документацию.

При автоматическом вызове модуля необходимо передавать ему определенный набор данных с использованием по возможности универсального механизма. Для разработки такого механизма необходимо создать максимально полный перечень данных, описывающих конкретное испытание. ОТМИ требуют различные наборы данных, которые могут включать: ТМИ, циклограмму испытания, включающую перечни команд и полетных заданий со временами ввода, файлы кодовых команд, файлы полетного задания, начальные условия моделирования, списки имитируемых отказов и т. д. В целях обеспечения возможностей по расширению номенклатуры ОТМИ необходимо как можно раньше выявить максимально возможный перечень данных и проработать методы их хранения. Например, средний размер файлов, относящихся к одному испытанию, на стенде составляет 400 Мбайт. Число испытаний на стендах для одного КА может составлять тысячи.

На текущий момент все эти данные хранятся в САИ и предоставляются по запросу для анализа. Именно в САИ разработчики в ручном режиме вносят результаты анализа испытаний. Поэтому целесообразно использовать именно САИ в качестве информационной системы для работы с ОТМИ.

Для обеспечения взаимодействия ОТМИ и САИ необходимо создать протокол информационного взаимодействия. Создание и соблюдение такого протокола позволяет упростить подключение новых ОТМИ: любой созданный согласно протоколу обработчик может быть подключен к диспетчерской системе с минимальными усилиями. Благодаря этому появилась возможность автоматического переноса информации из САИ в ОТМИ, что избавило пользователей от рутинных операций по настройке ОТМИ под конкретный моделируемый режим. Информационный обмен проводится при помощи файлов, перечень и формат которых и определяется про-

токолом. В качестве наиболее подходящего и широко используемого формата файлов можно назвать XML. Рис. 3 (см. вторую сторону обложки) содержит примерный вид XML-файла с результатами обработки.

Получаемые XML-файлы должны обрабатываться САИ в том числе и для серии режимов, с формированием отчетной документации и при необходимости оповещением разработчиков. Файлы, содержащие более подробную расшифровку результатов обработки, должны передаваться на ftp-сервер и быть доступны для просмотра пользователям системы.

Таким образом, после внедрения предлагаемой методики в процесс отработки БПО работа САИ будет выглядеть следующим образом (рис. 4, см. вторую сторону обложки).

1. Пользователь САИ (специалист, ответственный за испытание, или разработчик ПМ) запускает анализ результатов выбранными ОТМИ.

2. САИ получает результаты моделирования с испытательных стендов, дополняет их необходимыми данными о промоделированном режиме, после чего формирует и передает массив данных в ОТМИ и запускает обработку.

3. ОТМИ обрабатывают результаты и формируют наборы файлов с результатами обработки в соответствии с протоколом обмена. Набор файлов представлен файлом с заключением, файлами с подробными результатами обработки. Файл с заключением в формате XML, определяющимся протоколом, содержит суждение о нормативности результата анализа с кратким описанием ошибки (если она есть) и предназначен для автоматизированной обработки средствами САИ. Остальные файлы с результатами обработки, содержание которых определяется разработчиком ОТМИ, предназначены для углубленного анализа при его необходимости.

4. САИ, получив результаты работы ОТМИ, проводит их интеграцию, вырабатывая итоговое заключение, обеспечивает его хранение и предоставление пользователю.

Как показывает мировой опыт создания сложных технических систем, выявление ошибок на ранних этапах разработки значительно сокращает расходы на их исправление [8]. В связи с этим разумно не только ускорить запуск обработчиков путем автоматической передачи в них данных из САИ, но и автоматически анализировать результаты моделирования с помощью ОТМИ на более раннем этапе отработки БПО — экспресс-анализе. Это позволяет выявлять большее число ошибок до привлечения к анализу разработчиков ПМ и сокращать трудозатраты разработчиков на анализ ТМИ.

Еще одним вопросом, на который стоит обратить внимание, является вопрос организации разработки ОТМИ с учетом предлагаемых изменений процесса. Если ранее ОТМИ могли корректироваться разработчиками уже в процессе анализа результатов моделирования со стендов, то теперь необходимо иметь готовый ОТМИ на момент начала анализа, т. е. проводить корректировку ОТМИ совместно с корректировкой ПМ с соответствующей их отработкой.

В итоге предлагается процесс отработки БПО модернизировать следующим образом:

- разработку ОТМИ проводить одновременно с разработкой ПМ в соответствии с унифицированным протоколом;
- расширить этап экспресс-анализа результатов моделирования за счет автоматического вызова ОТМИ.

Рис. 5 описывает модернизированную методику отладки БПО с учетом предложенных изменений. Изменения в методике выделены штриховыми линиями.

Реализация модернизированной методики отработки БПО

Для апробации предложенной методики были выбраны ОТМИ из уже существующих:

1) ОТМИ таймирования, контролирующий время выполнения бортовых программ;

2) ОТМИ анализа длительности выдачи корректирующих импульсов при управлении орбитальным движением КА (ВКИ).

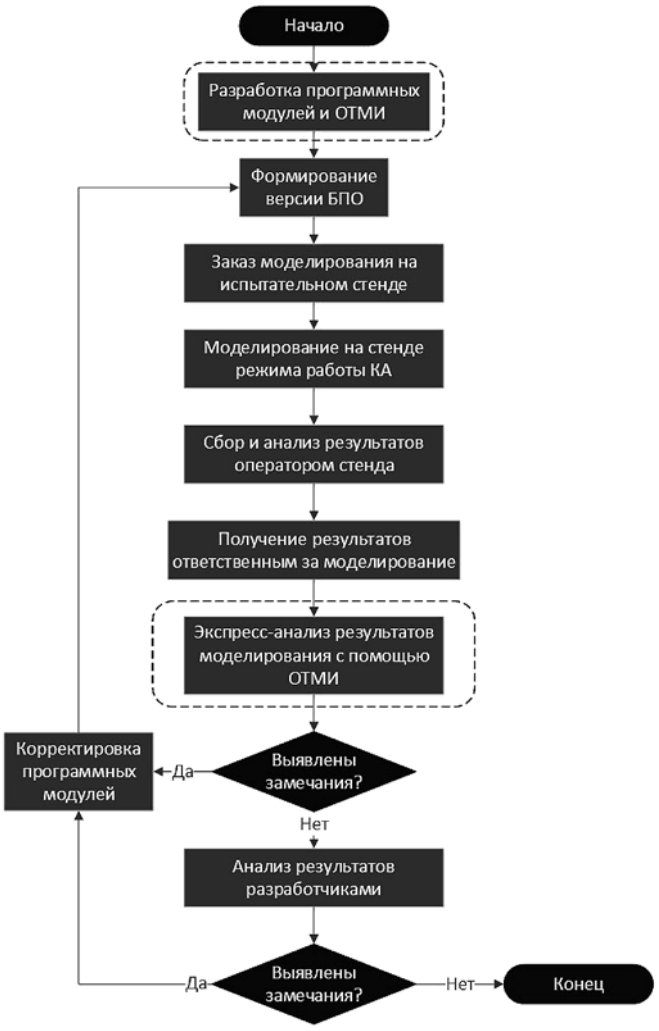

Рис. 5. Методика отладки БПО с использованием ОТМИ

Первый ОТМИ предназначен для выдачи суждения о нормативности функционирования программных подсистем в циклограмме бортового вычислителя на основании зафиксированного значения запаса времени функционирования ΔT, допустимого значения запаса $\Delta T_{доп}$ и наличия прерываний вычислений I. Суждения R формируются на основе решающих правил DR следующего вида:

$$DR_i\text{: Если } F_1 \wedge F_2 \ldots \wedge F_n,$$
$$\text{то } R = \text{Норма иначе } R = \text{Замечание},$$

где $F_1 \wedge F_2 \ldots \wedge F_n$ — предпосылка решающего правила.

В данном случае $n = 2$, $F_1 = [\Delta T > \Delta T_{доп}]$, $F_2 = [I = 0]$, и правило примет следующий вид:

$$\text{Если } [I = 0] \wedge [\Delta T > \Delta T_{доп}],$$
$$\text{то } R = \text{Норма иначе } R = \text{Замечание}.$$

Для более наглядного представления решающие правила сведены в таблицу.

Обработчик телеметрической информации ВКИ предназначен для вынесения суждения о нормативности отработки корректирующего импульса на основе его зафиксированной длительности $T_{факт}$ с учетом номинальной длительности $T_{номин}$ и допуска по времени отклонения $T_{допуск}$. Суждения R формируются на основе решающих правил DR следующего вида:

- DR_1: Если $[T_{факт} \pm T_{допуск} = T_{номин}]$, то R = Норма;
- DR_2: Если $[T_{факт} \pm T_{допуск} < T_{номин}]$, то R = Замечание;
- DR_3: Если $[T_{факт} \pm T_{допуск} > T_{номин}]$, то R = Замечание.

Наборы шаблонов для решающих правил были внесены в ОТМИ. После чего для каждого ОТМИ была заполнена база решающих правил.

Для реализации взаимодействия с САИ в каждый из рассмотренных ОТМИ добавлены следующие модули: модуль консольного запуска, модуль считывания решающих правил из базы знаний, модуль вывода результатов обработки в файл.

Для реализации взаимодействия с ОТМИ в САИ созданы следующие модули: модуль выбора ОТМИ; модуль считывания результатов обработки ТМИ; модуль добавления результатов обработки в СУБД; модуль отображения результатов обработки (рис. 6, см. вторую сторону обложки). Также модернизирована схема данных САИ в СУБД.

Решающие правила *DR*

$I = 0$	$\Delta T > \Delta T_{доп}$	
	Истина	Ложь
Истина	Норма	Замечание
Ложь	Замечание	Замечание

Заключение

Предложена методика отработки БПО с использованием баз знаний для автоматического анализа результатов моделирования полета КА. Методика заключается в создании баз знаний, выдающих заключения "норма" или "замечание" по результатам анализа данных моделирования полета КА, как для различных подсистем БПО СУ, так и в целом для БПО СУ. Каждая из баз знаний рассматривается как отдельный модуль, связанный с конкретной подсистемой. Модули объединятся информационной системой по единому протоколу взаимодействия. Система проводит автоматический запуск модулей и интегрирование результатов анализа от различных баз знаний.

Предлагаемый подход апробирован в МОКБ "Марс" путем создания двух баз знаний. Первая база содержит правила проверки нормативности времени выполнения программ. Вторая база содержит правила проверки нормативности выдачи корректирующего импульса для коррекции орбиты КА. Независимость баз знаний позволяет не только доверить их поддержку различным специалистам, но и реализовывать их на различных языках программирования. Могут быть использованы несколько способов реализации базы знаний: использование существующей оболочки для работы с базами знаний или же самостоятельное создание базы знаний на любом из языков программирования. Исходя из стоящих перед разработчиками задач и имеющихся ресурсов может быть выбран наиболее подходящий способ создания базы знаний.

Представленный подход показал свою эффективность и позволил сократить скорость выявления ошибок в указанных подсистемах, для которых реализованы базы знаний, с нескольких дней до нескольких минут.

Проведена апробация предложенного подхода при разработке БПО для КА Арктика-М, Спектр-Р, Спектр-РГ, Электро-Л № 1, Электро-Л № 2, Электро-Л № 3.

Список литературы

1. **Проектирование** и испытание бортовых систем управления: Уч. пособие / Под редакцией А. С. Сырова. М.: Изд-во МАИ-ПРИНТ, 2011. 344 с.
2. **Иванов Д. В., Казанкин Е. А.** Технологии разработки критического по безопасности программного обеспечения // Сайт Корпорации "Русские системы". URL: http://www.rusys.ru/docs/AAFSS/Statya.pdf
3. **Бровкин А. Г., Бурдыгов Б. Г., Гордийко С. В.** и др. Бортовые системы управления космическими аппаратами: Уч. пособие / Под редакцией А. С. Сырова. М.: Изд-во МАИ-ПРИНТ, 2010. 304 с.
4. **Косинский М. Ю., Шатский М. А.** Применение современных графических средств системного анализа для повышения эффективности процесса разработки архитектуры бортового программного обеспечения // Сборник статей VII Научно-технической конференции молодых ученых и специалистов Центра управления полетами г. Королев, М. О., ЦНИИмаш, 2017, С. 330—337.
5. **Щелыкалин М. Ю., Шатский М. А.** Использование информационных технологий при интеграции разнородных компонентов поддержки процесса отработки бортового программного обеспечения // Сборник статей V Научно-технической конференции молодых ученых и специалистов Центра управления полетами г. Королев, М. О., ЦНИИмаш, 2015. С. 365—372.

6. **Щелыкалин М. Ю.** Использование информационных технологий для поддержки разработки бортового программного обеспечения // Труды МАИ. 2016. № 88. Интернет-журнал. URL: http://mai.ru//upload/iblock/5ab/shchelykalin_rus.pdf

7. **Кравец В. Г.** Автоматизированные системы управления космическим полетом. М.: Машиностроение, 1995. 256 с.

8. **Вигерс К.** Разработка требований к программному обеспечению. М.: Издательско-торговый дом "Русская Редакция", 2004. 576 с.

Knowledge Base Application for Onboard Software Testing Results Analysis

M. Yu. Schelykalin, e-mail: Schelikal@gmail.com, **M. A. Shatsky**, e-mail: 246@mars-mokb.ru, **M. Yu. Kosinsky**, e-mail: 246@mars-mokb.ru, FGUP MOKB "Mars", Moscow, 127473, Russian Federation

Corresponding author:

Schelykalin Maxim Yu., Acting Head of the Laboratory, FGUP MOKB "Mars", Moscow, 127473, Russian Federation, E-mail: Schelikal@gmail.com

Received on December 11, 2018
Accepted on February 18, 2019

A new method to enhance spacecraft onboard software testing process based on decision rules application is proposed in the paper. Consider spacecraft onboard software consist of several interconnected subsystems, each of them must be thoroughly tested to meet requirements as well as the whole software complex.

Existing onboard software testing process and its significant milestones are analyzed. As a result, weak points of the whole process are determined. Especially, subsystems test result analysis stage takes relatively a long time and a lot of effort because of its complexity and current low automation level. Also results aren't saved systematically which makes it difficult for future analysis.

The proposed method uses knowledge bases to produce conclusions such as "Ok" or "Failed" as a result of software testing both for its subsystems separately and for the whole software.

Knowledge bases software is integrated into The Tests Automation System (TAS) software which is already developed. Every knowledge base comprises a software module that corresponds to particular software subsystem. Modules interact with TAS according to specially developed protocol based on xml standard. TAS executes the modules automatically after completion of software test to combine the results and save them into the database.

The proposed method is evaluated within "Mars" design bureau during the development of onboard software for several spacecraft (Spektr-R, Elektro-L, Arktika-M, Spektr-RG) and has proven its effectiveness.

Keywords: *onboard software, decision rules, testing, math modeling, development support system, test automation system*

For citation:

Schelykalin M. Yu., Shatsky M. A., **Kosinsky M. Yu**. Knowledge Base Application for Onboard Software Testing Results Analysis, *Programmnaya Ingeneria,* 2019, vol. 10, no. 5, pp. 219—225.

DOI: 10.17587/prin.10.219-225

References

1. ***Proektirovanie*** *i ispytanie bortovyh sistem upravleniya* (Design and testing of on-board control systems): Uchebnoe posobie / Eds A. S. Syrov. Moscow, Izd-vo MAI-PRINT, 2011, 344 p. (in Russian).

2. **Ivanov D. V., Kazankin E. A.** Tekhnologii razrabotki kriticheskogo po bezopasnosti programmnogo obespecheniya (Technology development critical safety software), *Korporaciya "Russkie sistemy"*, available at: http://www.rusys.ru/docs/AAFSS/Statya.pdf (in Russian).

3. **Brovkin A. G., Burdygov B. G., Gordijko S. V.** et al. *Bortovye sistemy upravleniya kosmicheskimi apparatami* (Onboard spacecraft control systems): Uchebnoe posobie / Eds A. S. Syrov, Moscow, Izd-vo MAI-PRINT, 2010, 304 p. (in Russian).

4. **Kosinskij M. YU., Shatskij M. A.** Primenenie sovremennyh grafiche-skih sredstv sistemnogo analiza dlya povysheniya ehffektivnosti processa razrabotki arhitektury bortovogo programmnogo obespecheniya (The use of modern graphic-based system analysis tools to improve the efficiency of the development process of the onboard software architecture), *Sbornik statej VII Nauchno-tekhnicheskoj konferencii molodyh uchyonyh i specialistov Centra upravleniya polyotami g. Korolyov*, M. O., CNIImash, 2017, pp. 330—337 (in Russian).

5. **Shchelykalin M. Yu., Shatskij M. A.** Ispol'zovanie informacionnyh tekhnologij pri integracii raznorodnyh komponentov podderzhki processa otrabotki bortovogo programmnogo obespecheniya (The use of information technology in the integration of heterogeneous components support the process of testing on-board software), *Sbornik statej V Nauchno-tekhnicheskoj konferencii molodyh uchyonyh i specialistov Centra upravleniya polyotami g. Korolyov*, M. O., CNIImash, 2015, pp. 365—372 (in Russian).

6. **Shchelykalin M. Yu.** Ispol'zovanie informacionnyh tekhnologij dlya podderzhki razrabotki bortovogo programmnogo obespecheniya (Use of information technology to support the development of onboard software), *Trudy MAI,* 2016, no. 88, internet-zhurnal, available at: http://mai.ru//upload/iblock/5ab/shchelykalin_rus.pdf (in Russian).

7. **Kravec V. G.** *Avtomatizirovannye sistemy upravleniya kosmicheskim poletom* (Automated space flight control systems), 1995, 256 p. (in Russian).

8. **Vigers K.** *Razrabotka trebovanij k programmnomu obespecheniyu* (Developing software requirements), Moscow, Izdatel'skotorgovyj dom "Russkaya Redakciya", 2004, 576 p. (in Russian).

УДК 004.925.5, 621.311.25:621.039.620.192 DOI: 10.17587/prin.10.226-233

О. М. Гулина, д-р техн. наук, проф., e-mail: olga@iate.obninsk.ru,
М. Н. Типикина, магистрант, e-mail: tipikinamariya@mail.ru,
Н. Г. Типикин, канд. техн. наук, доц., e-mail: tipikin@iate.obninsk.ru,
Обнинский институт атомной энергетики (ИАТЭ НИЯУ МИФИ)

Математическая модель визуализации данных толщинометрии трубопроводов АЭС и ее программная реализация

Описаны разработка и реализация алгоритмов трехмерной визуализации результатов эксплуатационного контроля толщинометрии трубопроводов в рамках проекта "Прогнозирование остаточного ресурса трубопроводов АЭС[1]". Анализ известных средств визуализации применительно к этой задаче позволил выявить их основные недостатки: высокая стоимость готовых продуктов, отсутствие привязки к форме и положению элемента, отсутствие возможности представления состояния элемента в случае 3D-визуализации и др. В связи с этим предлагается способ визуализации, при котором толщина трубопровода описывается цветом. Выполнена разработка алгоритма двумерной интерполяции цвета по заданному списку узлов, как базы для построения трехмерных моделей. Разработаны алгоритмы построения трехмерных моделей типовых элементов — прямых участков и гибов, окраска которых выполняется на основе созданных интерполяционных алгоритмов.

На основе предложенных в настоящей работе алгоритмов выполнена разработка программы для отображения результатов контроля.

Ключевые слова: *трубопровод, эрозионно-коррозионный износ, визуализация, графика, трехмерная модель, двумерная модель, интерполяция цвета, цветовая семантика, программа*

Введение

Эрозионно-коррозионный износ (ЭКИ) является распространенным механизмом деградации трубопроводов, которому подвержены многие элементы конденсатно-питательного и парового трактов АЭС. Эрозионно-коррозионный износ — процесс коррозионного повреждения металла за счет формирования и последующего удаления окисных пленок под воздействием потока среды. Многократное повторение этого процесса приводит к значительным утонениям стенки или разрушению трубопровода [1].

Эксплуатационный контроль состояния трубопроводов проводят с использованием различных методов. В большей части при эксплуатационном контроле проводится единичный замер толщин стенок трубопроводов. На рис. 1 представлен пример технологической карты замеров.

Проверки проводят один раз в 1...1,5 года, при этом обслуживают примерно 1600 элементов трубопроводов на одном блоке. После проведения замеров обрабатывают и тщательно анализируют результаты [2], однако только около 10 % данных могут быть оперативно обработаны. Таким образом, существует необходимость в использовании специальных программ для визуализации и обработки этих данных.

В разработанных ранее программах анализа результатов толщинометрии было реализовано несколько подходов для визуализации данных. Чаще всего это двумерные картограммы результатов измерений, в которых остаточная толщина трубопровода передается с помощью цвета. Эти программы имеют слабые возможности по управлению диапазонами представимых значений в силу дискретности отображения толщин в фиксированное множество цветов (рис. 2—4, см. третью сторону обложки). Вместе с тем возможности современной вычислительной техники позволяют получать трехмерные изображения, которые имеют большие выразительность и наглядность. Такие модели могут быть использованы при отображении как результатов измерений, так и результатов анализа.

С учетом изложенных выше соображений в настоящей публикации предложены математические модели трехмерной визуализации результатов эксплуатационного контроля толщин трубопроводов, которые позволяют получить реалистичные трехмерные изображения внутренней поверхности и передать особенности ЭКИ-повреждений цветом.

[1] АЭС — атомная электростанция.

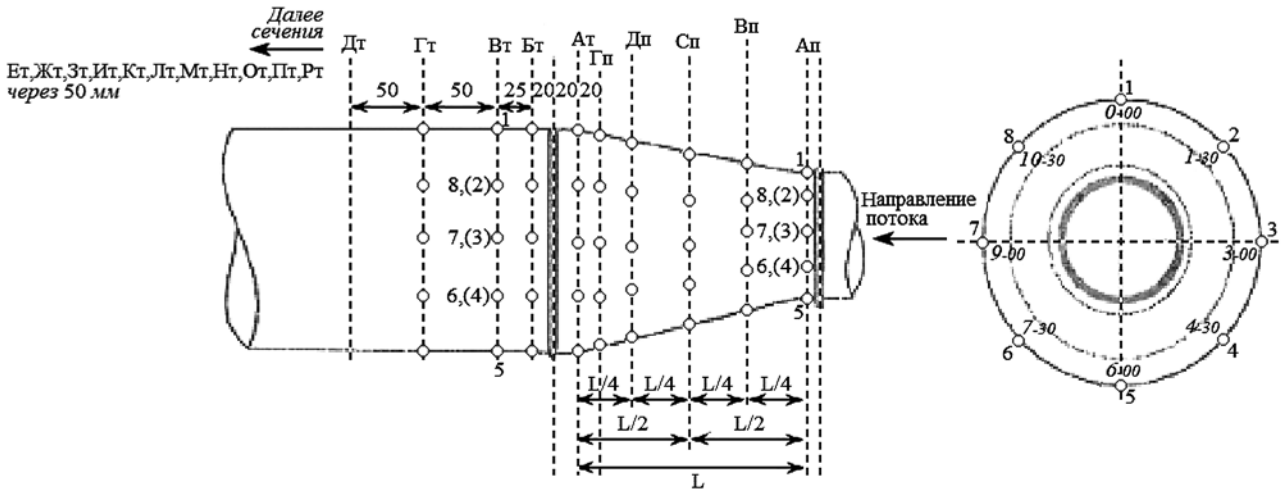

Рис. 1. Технологическая карта замеров

Обзор способов визуализации, реализованных в существующих программах

Результаты замеров толщин стенок трубопроводов представляются в протоколах замеров и имеют вид матрицы (прямоугольной таблицы) с разным числом строк и столбцов в зависимости от типа элемента трубопровода (прямой участок, гиб[2], околошовная зона и т. д.). Для оценки остаточного ресурса[3] элемента необходимы определенные преобразования и расчеты, однако общее представление о характере износа можно получить с помощью картограмм.

На рис. 2—4 (см. третью сторону обложки) представлены примеры двумерных и трехмерных разверток, построенных разными авторами по результатам измерений толщин стенок трубопроводов [3]. Недостатком представленных на рис. 2 и 3 картограмм является сложное для восприятия представление пространственного расположения элемента трубопровода и дефектов внутренней поверхности. Модель на рис. 3 отображает точечные значения, полученные из измерений, по ним сложно судить о состоянии всей поврежденной области.

На рис. 4 (см. третью сторону обложки) представлены примеры изображений результатов толщинометрии, построенные в системе Mathcad. Данные изображения можно считать адекватными, однако использование для построения моделей системы Mathcad является дорогостоящим. Кроме того, эта система достаточно требовательна к вычислительным ресурсам, а полученные изображения визуально не очень гладкие. При этом отсутствует возможность повернуть полученные изображения для всестороннего исследования.

В результате обзора существующих способов визуализации результатов толщинометрии можно сделать вывод о необходимости разработки новых программных средств, позволяющих строить как *двумерные*, так и *трехмерные модели* исследуемых участков трубопровода.

Исходные данные

Исходными данными для визуализации в рассматриваемой предметной области являются:

- геометрия элемента трубопровода;
- число замеров в осевом и в окружном направлениях;
- расстояния между замерами в осевом направлении;
- результаты замеров (в виде электронных таблиц);
- номинальная толщина стенки.

Такие данные, как правило, задаются в виде файлов MS Excel. Этот формат представляется достаточно удобным, так как MS Excel имеет широкий набор возможностей по форматированию файлов, а также по их экспорту из других форматов.

Подход к построению изображения

Для построения рассматриваемого в настоящей работе изображения предлагается следующий подход.

1. Строится трехмерное изображение одного из отмеченных ранее элементов.

2. Проводится *двумерная интерполяция* толщины стенки трубопровода по результатам эксплуатационного контроля. На основании полученных значений выполняется окрашивание изображения, чтобы выразить изменение толщины с помощью цвета.

3. Дополнительное сглаживание получается путем *билинейной интерполяции* цвета *методом Гуро* [4].

Интерполяция

Как было показано на рис. 2—4 (см. третью сторону обложки), в существующих программных ком-

[2] Гиб — колено, изготовленное в трубогибочном станке.

[3] Остаточный ресурс — продолжительность безопасной эксплуатации трубопровода на допустимых параметрах от данного момента времени до момента достижения прогнозируемого предельного состояния.

плексах результаты измерений отображаются в виде дискретных наборов точек. Однако в действительности дефект представляет собой гладкое углубление, являющееся результатом износа. Получить о нем адекватное представление можно путем построения гладкой поверхности с помощью интерполяции имеющихся измерений толщин.

Пусть на отрезке $a \leqslant \xi \leqslant b$ задана сетка

$$\omega = \{x_i | a = x_0 < x_1 < ... < x_i < ... < x_n = b\},$$

в узлах которой заданы значения функции $y(x)$:

$$y(x_0) = y_0, ..., y(x_i) = y_i, ..., y(x_n) = y_n,$$

общее число узлов сетки равно $(n + 1)$.

Требуется построить *интерполянту* — функцию $f(x)$, совпадающую с функцией $y(x)$ во всех узлах сетки [5]:

$$f(x_i) = y_i,\ i = 0, 1, ..., n. \quad (1)$$

Интерполирующие функции $f(x)$, как правило, строят в виде линейных комбинаций некоторых элементарных функций:

$$f(x) = \sum_{k=0}^{N} c_k \phi_k(x),$$

где $\{\phi_k(x)\}$ — фиксированные линейно независимые функции; N — число этих функций; $c_0, c_1, ..., c_N$ — не определенные коэффициенты.

Из условия (1) получаем систему из $(n + 1)$ уравнений относительно коэффициентов $\{c_k\}$:

$$\sum_{k=0}^{N} c_k \phi_k(x_i) = y_i,$$

где $i = 0, 1, ..., n$.

Кубический сплайн

Сплайном называется функция, которая вместе с несколькими производными непрерывна на всем заданном отрезке $[a, b]$, а на каждом частичном отрезке $[x_{i-1}, x_i]$ в отдельности является некоторым алгебраическим многочленом.

Максимальная по всем частичным отрезкам степень многочленов называется *степенью сплайна*, а разность между степенью сплайна и порядком наивысшей непрерывной на $[a, b]$ производной — *дефектом сплайна*.

На практике наиболее широкое распространение получили сплайны третьей степени, имеющие на $[a, b]$ непрерывную хотя бы первую производную, называемые *кубическими*. Кубический сплайн, принимающий в узлах x_i те же значения f_i, что и некоторая функция f, называется *интерполяционным* [6].

Пусть на отрезке $[a, b]$ задана некоторая функция $f(x)$. *Кубическим сплайном* дефекта 1 называется функция $S(x)$, которая:

— является некоторым алгебраическим многочленом степени не выше третьей на каждом частичном отрезке $[x_{i-1}, x_i]$;

— имеет непрерывные первую и вторую производные на всем отрезке $[a, b]$;

— интерполирует функцию f в точках x_i, т. е. $S(x_i) = f(x_i)$.

Для однозначного задания необходимо наложить дополнительные требования. Естественным кубическим сплайном называется кубический сплайн, удовлетворяющий также граничным условиям вида $S''(a) = S''(b) = 0$.

Интерполяция кубическими сплайнами

Интерполяция кубическими сплайнами является частным случаем кусочно-полиномиальной интерполяции. В этом специальном случае между любыми двумя соседними узлами функция интерполируется кубическим полиномом. Его коэффициенты на каждом интервале определяются из условий сопряжения в узлах:

$$f_i = y_i,\ f'(x_i - 0) = f'(x_i + 0),\ f''(x_i - 0) = f''(x_i + 0),$$

где $i = 0, 1, ..., n - 1$.

Кроме того, на границе при $x = x_0$ и $x = x_n$ ставятся условия:

$$f''(x_0) = 0,\ f''(x_n) = 0. \quad (2)$$

Будем искать кубический полином в следующем виде:

$$f(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3,$$

где $x_{i-1} \leqslant x \leqslant x_i$.

Из условия $f_i = y_i$ имеем:

$$\begin{gathered} f(x_{i-1}) = a_i = y_{i-1}, \\ f(x_i) = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 = y_i,\ h_i = x_i - x_{i-1}, \end{gathered} \quad (3)$$

где $i = 0, 1, ..., n - 1$.

Вычислим производные:

$$f'(x) = b_i + 2c_i(x - x_{i-1}) + 3d_i(x - x_{i-1})^2,$$

$$f''(x) = 2c_i + 6d_i(x - x_{i-1}),$$

где $x_{i-1} \leqslant x \leqslant x_i$.

Потребуем их непрерывности при $x = x_i$:

$$b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2,\ \ c_{i+1} = c_i + 3d_i h_i, \quad (4)$$

где $i = 1, 2, ..., n - 1$.

Общее число неизвестных коэффициентов, очевидно, равно $4n$, число уравнений (3) и (4) равно $(4n - 2)$. Недостающие два уравнения получаем из условия (2) при $x = x_0$ и $x = x_n$:

$$c_1 = 0,\ c_n + 3d_n h_n = 0.$$

Из выражения (4):

$$d_i = \frac{c_{i+1} - c_i}{3h_i},$$

подставляя это выражение в (3) и исключая $a_i = y_{i-1}$, получим:

$$b_i = \left[\frac{y_i - y_{i-1}}{h_i}\right] - \frac{1}{3}h_i\left(c_{i+1} + 2c_i\right), \quad i = 1,2,\ldots,n-1,$$

$$b_n = \left[\frac{y_n - y_{n-1}}{h_n}\right] - \frac{2}{3}h_n c_n.$$

Подставив теперь выражения для b_i, b_{i+1} и d_i в первую формулу (4), после несложных преобразований получим для определения c_i разностное уравнение второго порядка:

$$h_i c_i + 2\left(h_i + h_{i+1}\right)c_{i+1} + h_i + c_{i+2} = \\ = 3\left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}\right), \qquad (5)$$

где $i = 1, 2, ..., n - 1$, с краевыми условиями

$$c_1 = 0,\ c_{n+1} = 0. \qquad (6)$$

Условие $c_{n+1} = 0$ эквивалентно условию $c_n + 3d_n h_n = 0$ и уравнению $c_{i+1} = c_i + d_i h_i$. Разностное уравнение (5) с условиями (6) можно решить методом прогонки, представив в виде системы линейных алгебраических уравнений вида $\mathbf{Ax} = \mathbf{F}$, где вектор $\mathbf{x}$ соответствует вектору $\{c_i\}$, вектор $\mathbf{F}$ поэлементно равен правой части уравнения (6), а матрица $\mathbf{A}$ имеет следующий вид:

$$\mathbf{A} = \begin{Bmatrix} C_1 & B_1 & 0 & 0 & \ldots & 0 & 0 \\ A_2 & C_2 & B_2 & 0 & \ldots & 0 & 0 \\ 0 & A_3 & C_3 & B_3 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & B_{n-1} \\ 0 & 0 & 0 & 0 & \ldots & A_n & C_n \end{Bmatrix},$$

где $A_i = h_i$, $i = 2, ..., n$; $B_i = h_{i+1}$, $i = 1, ..., n - 1$; $C_i = 2(h_i + h_{i+1})$, $i = 1, ..., n$.

Метод прогонки

Метод прогонки основан на предположении, что искомые неизвестные связаны рекуррентным соотношением:

$$x_i = \alpha_i x_{i+1} + \beta_{i+1},\ i = 1, ..., n - 1.$$

Используя это соотношение, выразим x_{i-1} и x_i через x_{i+1} и подставим в i-е уравнение:

$$x_{i-1} = \alpha_i(\alpha_{i+1}x_{i+1} + \beta_{i+1}) + \beta_i = \\ = \alpha_i\alpha_{i+1}x_{i+1} + \alpha_i\beta_{i+1} + \beta_i,$$

$$(A_i\alpha_i\alpha_{i+1} + C_i\alpha_{i+1} + B_i)x_{i+1} + \\ + A_i\alpha_i\beta_{i+1} + A_i\beta_i + C_i\beta_{i+1} - F_i = 0,$$

где F_i — правая часть i-го уравнения. Это соотношение будет выполняться независимо от решения, если потребовать:

$$A_i\alpha_i\alpha_{i+1} + C_i\alpha_{i+1} + B_i = 0,$$
$$A_i\alpha_i\beta_{i+1} + A_i\beta_i + C_iB_{i+1} - F_i = 0.$$

Отсюда следует:

$$\alpha_{i+1} = \frac{-B_i}{A_i\alpha_i + C_i},\ \beta_{i+1} = \frac{F_i - A_i\beta_i}{A_i\alpha_i + C_i}.$$

Из уравнения для $i = 1$ получим:

$$\alpha_2 = \frac{-B_1}{C_1},\ \beta_2 = \frac{F_1}{C_1}.$$

После нахождения прогоночных коэффициентов α и β, используя уравнение (1), получим решение системы. При этом

$$x_n = \frac{F_n - A_n\beta_n}{C_n + A_n\alpha_n}.$$

Реализация трехмерной интерполяции

По имеющимся исходным данным строится интерполяционная кривая вдоль одной из образующих исследуемого тела, например, в осевом направлении вдоль образующей "1-30 часа" (рис. 5, *а*).

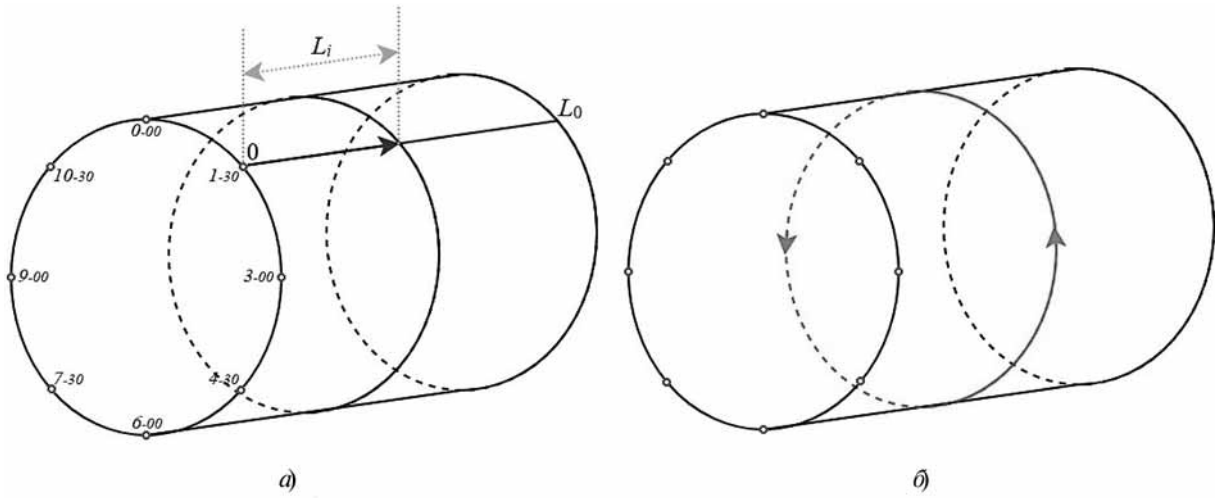

Рис. 5. Выбор направляющей

В ходе построения расстояние L_i от начальной окружности до окружности, обрабатываемой на текущем шаге построения, изменяется в пределах от 0 до L_0 (длина исследуемого участка или длина круговой образующей) с шагом δ. На каждом шаге выполняется вычисление аппроксимированных значений толщин для текущего сечения.

По полученным значениям строится интерполяционный полином по круговой образующей участка (рис. 5, *б*). Проводится интерполяция вдоль круговой образующей, и полученные значения используются для задания цвета поверхности.

Матричные преобразования

Матричные преобразования являются распространенным приемом в компьютерной графике при построении трехмерных изображений [4]. При этом любое преобразование описывается в матричном виде как комбинация переносов, поворотов и масштабирований (матрицы).

Алгоритм построения изображения

В OpenGL используется левосторонняя система координат. В этой системе ось *x* направлена горизонтально вправо от наблюдателя, ось *y* вертикально вверх, ось *z* в глубину экрана. В данной системе координат на расстоянии *r* от начала координат выбирается точка на оси *y*, *r* — радиус начальной окружности. Для построения множества точек начальной окружности выполняется вращение выбранной точки в плоскости *xy* вокруг оси *z* на угол φ (рис. 6, *а*), значение которого определяется по формуле: $\frac{2\pi}{n}$, где *n* — число промежуточных точек. Число *n* выбирается произвольно так, чтобы полученное изображение было визуально гладким. В результате имеем множество точек $\mathbf{P} = \{P_i\}$, где каждая следующая точка находится по формуле: $\mathbf{P}_{i+1} = P_i \cdot \mathbf{R}_Z(\varphi)$, где $\mathbf{R}_Z(\varphi)$ — матрица вращения вокруг оси *z*.

Путем смещения точки на угол φ получается новая координата, затем выполняется интерполяция для вычисления цвета для окрашивания каждой точки. Результатом является ломаная в плоскости *xy*, приближенная к окружности (рис. 6, *б*).

Для построения цилиндра (прямого участка) каждую из точек необходимо сдвинуть вдоль оси *z* на некоторое расстояние *dz*. В результате имеем новое множество точек — вторую окружность (рис. 7, *а*). Множество точек следующей окружности рассчитывается по формуле:

$$\mathbf{P}' = \mathbf{P}\mathbf{T}_Z(dz), \qquad (7)$$

где $\mathbf{T}_Z(dz)$ — матрица переноса вдоль оси *z*.

Следующим шагом является построение множества треугольников, путем соединения пары точек множеств соседних окружностей **P** и **P′** (рис. 7, *б*). Данное множество треугольников будет приближать поверхность цилиндра.

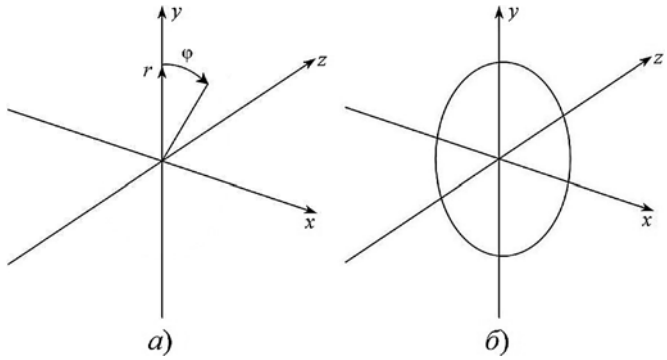

Рис. 6. Построение множества точек начальной окружности

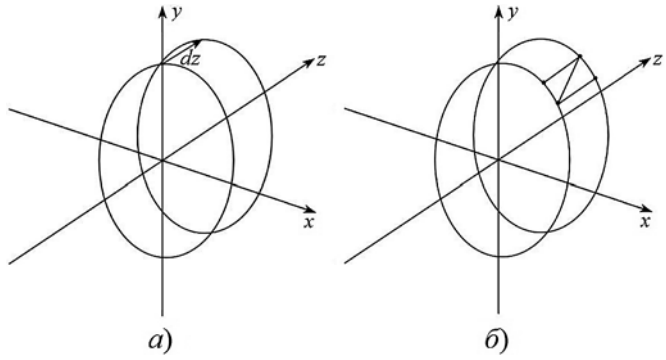

Рис. 7. Построение треугольников на множествах точек соседних окружностей

Для построения гиба аналогичным образом строится множество точек $\mathbf{P} = \{P_i\}$ начальной окружности радиуса *r*. Пусть *R* — радиус оси гиба (рис. 8). Будем считать, что центр начальной окружности располо-

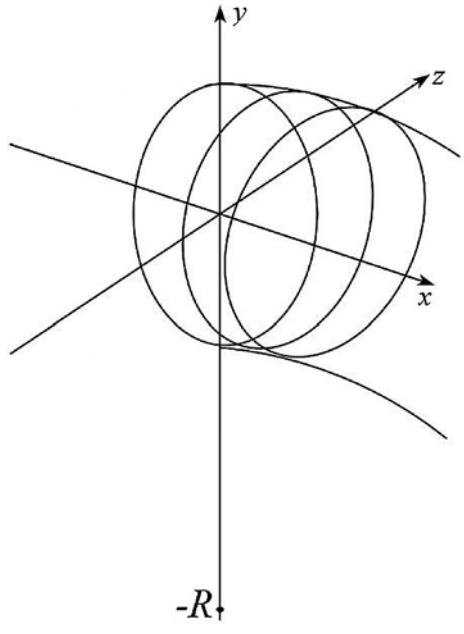

Рис. 8. Построение гиба

жен в начале координат, а центр образующей гиба находится в точке $-R$ на оси y.

Множество точек окружности на очередном шаге алгоритма строится следующим образом: выполняется перенос множества точек первой окружности вдоль оси y, поворот точки вокруг оси x на некоторый угол θ, а затем обратный перенос. Матрица преобразования координат

$$\mathbf{M} = \mathbf{T}_y(R)\mathbf{R}_x(\theta)\mathbf{T}_y(R).$$

Множество точек следующей окружности рассчитывается по формуле:

$$\mathbf{P}' = \mathbf{PM}. \qquad (8)$$

Далее строится множество треугольников, соединяющих пары точек множеств $\mathbf{P}$ и $\mathbf{P}'$.

Очевидно, что формулы (7) и (8) одинаковы, различны лишь матрицы преобразования координат. Пусть есть некоторая генерируемая матрица

$$\mathbf{M}_{\mathbf{gen}} = \begin{cases} \mathbf{T}_z(dz), & \text{для цилиндра} \\ \mathbf{T}_y(R)\mathbf{R}_x(\theta)\mathbf{T}_y(R) & \text{для гиба.} \end{cases}$$

Тогда можно построить общий алгоритм построения.

Шаг 1. Выбор начальной точки построения на расстоянии r на оси y.

Шаг 2. Поворот начальной точки вокруг оси z для получения множества точек $\mathbf{P} = \{P_i\}$ начальной окружности.

Шаг 3. Формирование матрицы $\mathbf{M}_{\mathbf{gen}}$ для построения нужного типа элемента.

Шаг 4. Циклическое выполнение построения множества точек $\mathbf{P}'$ по формуле $\mathbf{P}' = \mathbf{PM}_{\mathbf{gen}}$.

Шаг 5. Построение треугольников на множествах $\mathbf{P}$ и $\mathbf{P}'$.

Шаг 6. Далее $\mathbf{P} = \mathbf{P}'$.

Шаг 7. Повторение шагов 4 и 5 до тех пор, пока не будет выполнено построение всей фигуры.

Цветовая семантика значения отклонения

Данные толщинометрии представляют собой массив значений, которые могут иметь некоторое отклонение от номинального значения толщины. Ранее было отмечено, что данные толщинометрии хранятся в виде электронных таблиц. Для обеспечения высокой степени наглядности при отображении массива данных в таблицу можно выполнить окраску ячеек этой таблицы в зависимости от значения отклонения.

В технических приложениях смысл цветовых маркеров достаточно жестко определяет применение каждого цвета: красный цвет — "опасность", зеленый — "безопасность". Утонение стенки трубопровода — опасное явление, поэтому участки, где толщина меньше номинальной, помечаются красным цветом, причем интенсивность цвета выбирается пропорционально значению утонения. Толщина больше номинальной является более безопасной, поэтому соответствующие участки окрашиваются в зеленый цвет.

Для вычисления компонент цвета необходимо выполнить перечисленные далее действия.

1. Определить максимальное и минимальное значения отклонения, в том числе:

1.1 определить максимальное и минимальное значения в массиве результатов (max, min);

1.2 определить максимальное отклонение от номинального значения (nominal)

$$\Delta_{\max} = \max - \text{nominal};$$

определить минимальное отклонение

$$\Delta_{\max} = \text{nominal} - \max.$$

2. Цвет ячейки определяется по следующему алгоритму:

2.1 пусть $\boldsymbol{R}$, $\boldsymbol{G}$, $\boldsymbol{B}$ — красная, зеленая и синяя составляющие цвета, value — текущее значение массива результатов, где

$$\boldsymbol{R} = 255;\ \boldsymbol{G} = 255;\ \boldsymbol{B} = 255;$$

2.2 если значение массива больше номинального значения (value > nominal), то

$$\Delta = \frac{\text{value} - \text{nominal}}{\Delta_{\max}} \cdot 255,\ \boldsymbol{R} = 255 - \Delta,\ \boldsymbol{B} = 255 - \Delta;$$

если значение массива больше номинального значения (value < nominal), то

$$\Delta = \frac{\text{nominal} - \text{value}}{\Delta_{\min}} \cdot 255,\ \boldsymbol{G} = 255 - \Delta,\ \boldsymbol{B} = 255 - \Delta\,.$$

Таким образом, ячейка, содержащая значение, много меньшее номинального значения, — утонение, окрашивается в ярко красный цвет. Ячейка, содержащая значение, много большее номинального значения, — отложение, окрашивается в ярко зеленый цвет. Ячейка, содержащая значение, равное номинальному значению, окрашена в серый цвет.

Программная реализация

В рамках исследований на рассматриваемом направлении была создана программная реализация описанных в настоящей статье алгоритмов. С ее помощью данные, полученные в результате эксплуатационного контроля, могут быть визуализированы в виде таблицы, интенсивность цвета в ячейках которой пропорциональна значению износа. Кроме того, программа позволяет выводить двумерные картограммы и трехмерные модели поверхности исследуемого участка трубопровода.

Разработка программы осуществлялась на языке программирования *C#* с использованием графической библиотеки *OpenGL — Open Graphics Library.*

На рис. 9—11 (см. четвертую сторону обложки) приведены примеры снимков экрана разработанной программы. На рис. 9 показан результат отображения считанных данных в виде двумерной раскрашенной

таблицы. На экране также показаны предельные значения, полученные из исходных данных. Красным выделены области утонения, зеленым — области утолщения.

Рис. 10 иллюстрирует отображение двумерной картограммы с интерполяцией дефектов в виде непрерывных изменений цвета. Две приведенные картограммы построены с использованием реальных результатов толщинометрии для околошовной зоны трубопровода одного из энергоблоков АЭС с энергоблоками РБМК-1000 (реактор большой мощности канальный) в 2011 (*а*) и 2016 (*б*) годах [9].

Примеры трехмерных изображений приведены на рис. 11, это модели прямого участка (*а*) и гиба трубопровода (*б*). Для всестороннего исследования построенные модели могут быть повернуты в любом направлении путем нажатия левой клавиши мыши и перемещения курсора мыши в нужном направлении.

Заключение

В рамках настоящей работы предложены модель и алгоритмы трехмерной визуализации на основе интерполяции результатов измерений толщин трубопроводов. Действие разработанных процедур проверено на основе протоколов толщинометрии с действующих АЭС.

Практическая значимость разработанных моделей заключается в возможности применения разработанных методов при отображении результатов исследований состояния трубопроводов АЭС в условиях эрозионно-коррозионного износа. Визуальное представление результатов с помощью цвета обладает наглядностью и, как следствие, способно повысить эффективность деятельности специалистов.

Список литературы

1. **РД ЭО 0571—2010.** Нормы допускаемых толщин стенок элементов трубопроводов из углеродистых сталей при эрозионно-коррозионном износе. 2010.
2. **ПНАЭ Г-7-002—86.** Нормы расчета на прочность оборудования и трубопроводов атомных энергетических установок. 1986.
3. **Бараненко В. И., Гулина О. М., Тарасова О. С.** Эксплуатационный контроль трубопроводов, подверженных эрозионно-коррозионному износу. М.: Теплоэнергетика, 2009. 69 с.
4. **Гайдуков С.** Профессиональное программирование трехмерной графики. СПб.: БХВ-Петербург, 2004. 736 с.
5. **Корн Г., Корн Т.** Справочник по математике для научных работников и инженеров. М: Наука. Главная редакция физико-математической литературы, 1984. 832 с.
6. **Волков Е. А.** Численные методы. 2-е изд., исправленное. М.: Наука, 1987. 248 с.
7. **Фоли Дж., вэн Дэм А.** Основы интерактивной машинной графики: В 2-х кн., Кн. 1. М.: Мир, 1985. 368 с.
8. **Райт-мл. Р. С., Липчак Б.** OpenGL. Суперкнига. 3-е изд. М.: Вильямс, 2006. 1040 с.
9. **Бараненко В. И., Кузьмичевский А. Ю.** Особенности коррозионного износа трубопроводов питательной воды энергоблока АЭС с РБМК-1000. Энергоблок РД ЭО 0571—2010. Нормы допускаемых толщин стенок элементов трубопроводов из углеродистых сталей при эрозионно-коррозионном износе. М.: Теплоэнергетика, 2017. 27 с.

Mathematical Model of Data Visualization of the NPP Pipelines Thickness and Software Implement

M. N. Tipikina, tipikinamariya@mail.ru, **O. M. Gulina,** olga@iate.obninsk.ru,
N. G. Tipikin, tipikin@iate.obninsk.ru, Obninsk Institute for Nuclear Power Engineering (IATE MEPHI), Obninsk, 249040, Russian Federation,

Corresponding author:

Tipikina Mariya N., Student, Obninsk Institute for Nuclear Power Engineering (IATE MEPHI), Obninsk, 249040, Russian Federation,
E-mail: tipikinamariya@mail.ru

Received on November 19, 2018
Accepted on February 18, 2019

The article is devoted to the development and implementation of algorithms for three-dimensional visualization of the results of operational monitoring of pipe thickness gauging within the framework of the project "Prediction of the residual lifetime of NPP pipelines". Analysis of the known visualization tools revealed their main drawbacks: the high cost of the finished products, the lack of reference to the form and position of the element, the impossibility of representing the state of the element in the case of 3D visualization, etc.

In this regard, a visualization method is proposed, in which the pipeline thickness is described by color. A two-dimensional color interpolation algorithm has been developed for a given list of nodes as a base for building three-dimensional models. Algorithms are developed for constructing three-dimensional models of typical elements — straight sections and bends, which are painted on the basis of the interpolation algorithms created, including cubic spline interpolation, sweep method and matrix transformation.

A program has also been developed for displaying data from wall thickness measurements in the form of a table, the intensity of color in the cells of which is proportional to the amount of wear. The developed algorithms are imple-

mented as a C# program using the Open GL graphic library.

Keywords: *pipeline, erosion-corrosion wear, visualization, graphics, three-dimensional model, two-dimensional model, interpolation of color, color semantics, program*

For citation:

Tipikina M. N., Gulina O. M., Tipikin N. G. Mathematical Model of Data Visualization of the NPP Pipelines Thickness and Software Implement, *Programmnaya Ingeneria,* 2019, vol. 10, no. 5, pp. 226—233

DOI: 10.17587/prin.10.226-233

References

1. **RD EO 0571—2010.** Normy dopuskaemyh tolshhin stenok jelementov truboprovodov iz uglerodistyh stalej pri jerozionno-korrozionnom iznose (Norms of permissible wall thicknesses of carbon steel pipeline elements during erosion-corrosive wear). 2010 (in Russian).

2. **PNAE G-7-002—86.** Normy rascheta na prochnost' oborudovanija i truboprovodov atomnyh jenergeticheskih ustanovok (Standards for strength calculation of equipment and pipelines of nuclear power plants). 1986 (in Russian).

3. **Baranenko V. I., Gulina O. M., Tarasova O. S.** *Jekspluatacionnyj kontrol' truboprovodov, podverzhennyh jerozionno-korrozionnomu iznosu* (Operational control of pipelines subject to erosive wear), Moscow, Teplojenergetika, 2009, 69 p. (in Russian)

4. **Gajdukov S.** *Professional'noe programmirovanie trehmernoj grafiki* (Professional three-dimensional graphics programming), Saint Petersburg, BHV-Petersburg, 2004, 736 p. (in Russian).

5. **Korn G., Korn T.** *Spravochnik po matematike dlja nauchnyh rabotnikov i inzhenerov* (Mathematics Handbook for Scientists and Engineers), Moscow, Nauka Glavnaja redakcija fiziko-matematicheskoj literatury, 1984, 832 p. (in Russian).

6. **Volkov E. A.** *Chislennye metody* (Numerical methods), Moscow, Nauka, 1987, pub. 2, 248 p. (in Russian).

7. **Foley J. D., Van Dam A.** *Osnovy interaktivnoj mashinnoj grafiki* (Fundamentals of Interactive Computer Graphics), Moscow, Mir, 1985, vol. 1, 368 p. (in Russian).

8. **Rajt-ml. R. S., Lipchak B.** *OpenGL. Superkniga* (OpenGL. Superbook), Moscow, Vil'jams, 2006, pub. 3, 1040 p. (in Russian).

9. **Baranenko V. I., Kuz'michevskij A. Ju.** *Osobennosti korrozionnogo iznosa truboprovodov pitatel'noj vody jenergobloka AJeS s RBMK-1000 Jenergoblok № 3 Leningradskoj AJeS* (Features of corrosive wear of feedwater pipelines of the NPP unit with RBMK-1000, Unit 3 of the Leningrad NPP), Moscow, Teplojenergetika, 2017, 27 p. (in Russian).

ИНФОРМАЦИЯ

УДК 004.9 DOI: 10.17587/prin.10.234-240

А. А. Рогов, д-р техн. наук, проф., зав. кафедрой, e-mail: rogov@petrsu.ru,
К. А. Кулаков, канд. физ.-мат. наук, доц. кафедры, e-mail: kulakov@cs.karelia.ru,
Н. Д. Москин, канд. техн. наук, доц., доц. кафедры, e-mail: moskin@petrsu.ru,
Петрозаводский государственный университет

Программная поддержка в решении задачи атрибуции текстов

Рассмотрены вопросы оцифровки, разметки и программной поддержки в решении задачи атрибуции текстов. Эффективное решение актуальных задач анализа текстовых произведений, включая поиск заимствований и определение авторства, требует совмещения работы эксперта-лингвиста и применения различных математических методов над большими объемами данных. Представлено описание модифицированного программного комплекса "СМАЛТ" для реализации инструментария задачи атрибуции текстов и модульной структуры информационной системы "Фольклор" для автоматизированного построения и анализа теоретико-графовых моделей текстов. Приведена также математическая модель для разрешения актуальной задачи определения грамматических признаков в случае омонимии.

Ключевые слова: *атрибуция текстов, разметка текстов, интеллектуальный анализ данных, программный комплекс "СМАЛТ", информационная система "Фольклор", Ф. М. Достоевский*

Введение

В статье рассмотрена математическая и программная поддержка задачи атрибуции (установления авторства) анонимных текстов [1]. Данная задача возникает в разных областях науки, начиная с определения авторства известных литературных произведений (например, М. А. Шолохова, А. С. Пушкина, В. Шекспира, Ф. М. Достоевского) или исторических документов [2] до идентификации искусственно сгенерированного текста [3]. Актуальной проблемой является поиск заимствований и плагиата, который стал распространенным явлением в сфере науки и образования, в том числе и благодаря бурному развитию интернета [4]. Не менее важным представляется проведение авторoведческой экспертизы в криминалистике [5].

В настоящее время одной из актуальных задач является анализ публицистических статей XIX в., а именно около 500 неатрибутированных текстов из журналов "Время" (1861—1863 гг.), "Эпоха" (1864—1865 гг.) и еженедельника "Гражданин" (1873—1874 гг.) [6]. Известно, что Ф. М. Достоевский (вместе со своим братом М. М. Достоевским) редактировал и возглавлял эти журналы, поэтому уже давно ведутся исследования на предмет принадлежности его перу данных произведений. Большое число этих статей опубликовано анонимно, т. е. либо без подписи, либо под псевдонимами. Впрочем, это относится и к статьям, которые исследователи давно приписывали Достоевскому, более или менее основываясь на документальных данных.

Текст изучают на разных уровнях [7], а именно пунктуационном, орфографическом, синтаксическом, лексико-фразеологическом и стилистическом (отметим, что под "авторским стилем" обычно понимают последние три уровня). Существует два подхода к проведению таких исследований, включая экспертный (выполняемый профессиональным экспертом-лингвистом) и формальный (с помощью математических методов). При использовании математических методов требуется провести морфологическую разметку исследуемых и контрольных текстов. Среди существующих морфологических анализаторов на русском языке можно отметить системы LINGVO-MASTER (Институт проблем информатики РАН), TREETON (факультет вычислительной математики и кибернетики МГУ), ЭТАП-3 (Институт проблем передачи информации РАН), SemSin и др. [8].

В качестве примера размеченного корпуса текстов можно выделить "Национальный корпус русского языка". Однако он плохо приспособлен для решения задачи атрибуции, прежде всего вследствие ограниченности в выборе параметров [9]. Наиболее приспособлены для решения этой задачи размеченные корпуса, полученные с помощью информационных систем "СМАЛТ" (Статистические методы анализа литературных текстов) и "Фольклор", которые разработаны в Петрозаводском государственном университете [9—12].

Для установления авторства произведений используют: методы, основанные на нейронных сетях; метод QSUM; деревья решений; машину опорных векторов (SVM); метод k-средних; байесовский клас-

сификатор; марковские цепи; метод главных компонент; дискриминантный анализ; генетические алгоритмы; статистические критерии (хи-квадрат Пирсона, критерий Стьюдента, Колмогорова—Смирнова) и др. (подробнее обзор методов, а также результатов их применения представлен в работах [13—15]). Отметим существующие отечественные программные средства атрибуции текстов [7]:

- "Стилеанализатор" (в основе — марковские цепи, нейронные сети, деревья решений);
- "Авторовед" (нейронные сети, метод опорных векторов, QSUM);
- "Атрибутор" (марковские цепи);
- "Лингвоанализатор" (марковские цепи).

Разработчики некоторых из этих программ сами указывают на их ограниченность при проведении серьезных научных исследований [7, 13]. Наиболее представительно с этих позиций выглядит комплекс "Стилеанализатор". Однако авторам данной статьи не удалось найти информацию о типе разметки, применяемой в этом комплексе. При проведении исследований в ряде случаев можно воспользоваться общепринятыми статистическими пакетами, такими как Statistica, SPSS, языком программирования *R* и т. д.

Одним из самых полных и авторитетных исследований по атрибуции анонимных и псевдонимных публицистических статей из журналов "Время" и "Эпоха" является работа, которая была выполнена специалистом по использованию компьютерных технологий в атрибуции литературных произведений Г. Хетсо в 1979 г. [16]. Публикация его результатов вызвала поток критики как со стороны филологов, так и со стороны специалистов по математической статистике.

Несмотря на большое количество исследований в этой области, существует известный скептицизм к их результатам. Во-первых, это связано с отсутствием возможности повторить проведенное исследование по причине отсутствия полной информации о размеченных текстах. Во-вторых, отсутствуют исследования, связанные с пригодностью используемых математических методов для решения задачи атрибуции. Модернизируемый программный комплекс "СМАЛТ", который описан далее, призван исправить эту ситуацию.

Программный комплекс "СМАЛТ"

Работы по созданию программного комплекса (ПК) "СМАЛТ" в Петрозаводском государственном университете начались в 1995 г. [9]. С тех пор прошло несколько модификаций, и в настоящее время он имеет в своей основе базу данных текстов, состоящую из публицистических статей XIX в. разной тематической направленности. К их числу относятся публикации в петербургских журналах "Время", "Эпоха", "Современник", "Гражданин", "Светоч", "Молва", "Библиотека для чтения", "Заря" [10] в оригинальной орфографии. Заметим, что многие тексты имеют двойную разбивку с разным набором грамматических параметров [11, 16].

Атрибуцию текста проводили с помощью отдельных программных модулей, которые работали на основе информации, выгруженной из ПК "СМАЛТ". При этом использовали различный набор правил, построенный как с применением методик Г. Хетсо [16], так и с использованием других статистических методов. Несмотря на полученные положительные результаты атрибуции текстов, осталось много вопросов и нерешенных задач. Далее отметим две из них.

1. Зависимость результатов атрибуции от типа разметки. Минимальные исследования в этой области были проведены только Ю. В. Сидоровым [17]. Современные размеченные корпуса текстов содержат единственную разметку, что не позволяет проводить подобные исследования.

2. Возможность автоматизации выбора корпуса текстов для проведения исследования и зависимость результатов атрибуции от этого.

Решение этих задач позволит повысить доверие к процедуре атрибуции. Однако существующий ресурс не позволил их решить, поэтому потребовалась его глобальная модернизация. Обозначим возникшие при этом наиболее существенные проблемные вопросы и способы их решения.

Автоматизация грамматической разметки

Заметим, что существующие программы для предварительной разметки и обработки текстов ориентированы на современные тексты и современную графику, что делает их использование невозможным для произведений в графике XIX в. В ПК при проведении грамматического разбора текста используется алгоритм автоматической разметки [11, 18]. Он заключается в том, что ищет (по написанию) разбираемое слово в словаре, после чего выдает разбор этого слова. Отметим основные проблемные вопросы автоматической разметки.

1. Некоторые слова отсутствуют в словаре и поэтому остаются неразмеченными. С расширением словаря таких слов становится меньше.

2. Существует проблема омонимии. Она заключается в том, что для некоторого слова в словаре может содержаться несколько вариантов его грамматической разметки, и встает вопрос, какой из них выбрать.

Решение этой проблемы основывается на размеченном корпусе текстов и представлено в работе [11] в следующем виде. Рассматриваются, учитывая порядок слов, тройки подряд идущих слов $v_1v_2v_3$ с соответствующими им разборами a_1, a_2 и a_3. В качестве оценки вероятности разбора берут $P(a_i) = \frac{n(a_i, v_i)}{N(v_i)}$, где $n(a_i, v_i)$ — число встречаемых в корпусе разборов a_i слова v_i, а $N(v_i)$ — общее число разборов слова, встречаемых в корпусе. Условные вероятности оценивают аналогично. Для оценки грамматического разбора слова v_3 можно воспользоваться равенством

$$P(a_1a_2a_3) = \alpha P(a_3)P(a_2|a_3)P(a_1|a_2a_3) + \\ +(1-\alpha)P(a_1)P(a_2|a_1)P(a_3|a_1a_2),$$

где α — настроечный коэффициент.

С использованием размеченного корпуса проводится статистическая оценка вероятностей из правой части равенства. Разбор $a_1a_2a_3$, на котором достигается максимум оценки функции $P(a_1a_2a_3)$, принимается как разбор по умолчанию в случае омонимии. Настроечный коэффициент α подбирается экспериментально. Для увеличения точности снятия омонимии этот метод был модифицирован [18]. Рассмотрим пять последовательных слов $v_1v_2v_3v_4v_5$. Для разбора слова v_3 предлагается использовать две тройки разборов $a_1a_2a_3$ и $a_3a_4a_5$. Оптимальный разбор будет соответствовать максимуму оценки функции

$$P(a_1a_2a_3a_4a_5) = \alpha P(a_3)P(a_2|a_3)P(a_1|a_2a_3) + \\ +(1-\alpha)P(a_3)P(a_4|a_3)P(a_5|a_3a_4).$$

Проведенные исследования показали перспективность предложенного подхода.

Точность грамматической разметки можно увеличить, используя дополнительную информацию об авторе текстов. Проведенное А. В. Скабиным исследование [19] на 28 произведениях Ф. М. Достоевского из корпуса "СМАЛТ" [9, 11] с общим числом слов более 80 тыс. показало, что всего он использовал около 69 тыс. уникальных пятерок последовательно идущих слов. Чаще всего встречаются словосочетания: "несмотря на то что", "и так далее и тому" и похожие. В этом случае используется грамматический разбор всего устойчивого словосочетания.

Архитектура программного комплекса "СМАЛТ"

Несмотря на достаточно большое число различных корпусов текстов русского языка и средств создания своих корпусов, актуальной является задача обеспечения комфортной работы по атрибуции текстов, построению русскоязычного корпуса и проведению лингвистических исследований в современных информационных средах. Использование клиент-серверной архитектуры позволяет выполнять совместные работы с синхронизацией данных на сервере. Такой подход предоставляет возможность избежать вопросов, связанных с потерей данных частью сообщества исследователей, а также позволяет другим участникам оперативно получать новую информацию. Ключевыми особенностями предметной области являются значительный объем разнородной информации (например, грамматические, синтаксические, семантические разборы) и большое число операций с текстом (например, ручная разметка частей текста, грамматический разбор слова в тексте, разбор всего текста, анализ разборов текстов).

В качестве хранилища выбрана документо-ориентированная база данных MongoDB. В ней можно хранить данные в виде структурированных документов (объектов), что дает возможность работать с произвольным содержимым, позволяя при этом создавать объекты, удобные для использования как специалистами-филологами, так и техническим персоналом. Серверная часть (Backend) реализована на базе микрофреймворка Slim3 (PHP). Микрофреймворк позволяет быстро и легко организовать REST — интерфейс для загрузки и сохранения данных, а также для выдачи необходимых результатов по запросу. Решение трудоемких задач, например, автоматический разбор текстов или математические методы анализа текстов, вынесены за пределы серверной части и реализованы в виде обработчиков для платформы Gearman. Платформа Gearman позволяет выполнять распределенные вычисления с использованием очереди задач и может быть масштабирована в зависимости от потребностей. Клиентская часть реализована на базе библиотеки построения интерфейсов пользователя React JS. Библиотека позволяет реализовать функционально-ориентированный подход, когда интерфейс пользователя подстраивается под выполняемые задачи и текущие состояния. Для хранения и управления данными на клиентской стороне используется библиотека Redux. Она позволяет организовать единое хранилище данных для всех компонентов интерфейса пользователя, выполнять обновление состояний компонентов интерфейса и последующую перерисовку.

Программный комплекс имеет модульную структуру, изображенную на рис. 1. Каждый модуль представлен в виде наборов компонентов клиентской и серверной частей. Модульная структура добавляет гибкости в использовании ПК и позволяет конфигурировать систему под необходимые задачи [20].

Модуль "Пользователь" предоставляет механизмы идентификации и аутентификации пользователя, а также взаимодействия с пользователем. Модуль "Группа" реализует возможности организации коллаборации между пользователями и управления правами доступа.

Деятельность по анализу текстовых произведений и статистических исследований реализована в виде проектов. Модуль "Проект" предоставляет возможности по конфигурированию задач и управлению

Рис. 1. Структурная схема ПК "СМАЛТ"

механизмами публикации результатов деятельности в сети Интернет.

В ПК реализуются два типа проектов: выполнение разметки текстового произведения (модуль "Текст") и статистический анализ текстов (модуль "Исследования"). Разметка текстового произведения выполняется с помощью модулей "Грамматика" (грамматический анализ), "Синтаксис" (синтаксический анализ) и "Семантика" (семантический анализ). Каждый модуль содержит как ручной режим работы, так и автоматическую разметку текста. Для обеспечения удобства ручного режима разметки текстов используется модуль "Предикативный ввод" — для реализации механизмов шаблонов и подсказок. Анализ результатов деятельности и определение статуса проекта реализуются с помощью модуля "Статистика".

Модуль "Исследования" реализует интерфейс взаимодействия с одним из используемых методов анализа текстов [6, 12]. Модуль предоставляет доступ к разметке одного или нескольких текстовых произведений и формирует отчет о выполненном исследовании.

В настоящее время подготовлен перечень из более чем 400 текстов различных авторов XIX в., требующих разбора и атрибуции. Они будут анализироваться в том числе и с помощью ПК "СМАЛТ".

Применение информационной системы "Фольклор" в задаче атрибуции текстов

Параллельно с программным комплексом "СМАЛТ" в Петрозаводском государственном университете велась разработка информационной системы (ИС) "Фольклор" в среде визуального программирования Delphi 7.0. Свое название система получила потому, что изначально предназначалась для исследования коллекций фольклорных текстов (впоследствии к ним также добавились коллекции исторических и литературных текстов) [21]. Особенностью программы является предоставленная ею возможность автоматизированного построения теоретико-графовых моделей на основе текстов и их последующий анализ (в том числе для решения задачи атрибуции текстов).

Программа включает в себя перечисленные далее функциональные модули (общая структура представлена на рис. 2).

1. Основной модуль предназначен для ознакомления с коллекциями и является связующим звеном между остальными частями программы.

2. Модуль ввода и редактирования текстов.

3. Модуль анализа текстов. Содержит процедуры графематического, морфологического и контент-анализа текстов, встроенный морфологический словарь (рис. 3).

4. Модуль построения теоретико-графовых моделей текстов, программным кодом которого реализована описанная далее пошаговая процедура.

Рис. 2. Структура информационной системы "Фольклор"

Шаг 1. Выбор параметров построения теоретико-графовых моделей.

Шаг 2. Определение объектов в тексте.

Шаг 3. Разбиение объектов на группы.

Шаг 4. Определение отношений в тексте.

Шаг 5. Разбиение отношений на группы.

5. Модуль визуализации теоретико-графовых моделей. Включает методы двухмерной и трехмерной визуализации, основанные на физических аналогиях и поуровневом рисования деревьев.

6. Модуль анализа теоретико-графовых моделей, который включает в свой состав следующие компоненты:

- агрегации теоретико-графовых моделей;
- вычисления числовых характеристик теоретико-графовых моделей (например, коэффициенты связности, распределение объектов и связей на группы, распределение степеней и функциональных весов вершин и др.);
- вычисления матриц расстояний между графами, основанных на подграфовых метриках;
- классификации текстов с помощью процедуры кластерного анализа.

В настоящее время с помощью ИС "Фольклор" был проведен ряд исследований по атрибуции текстов. В частности, решалась задача распознавания фольклорных текстов (не имеющих автора) и стилизованных под фольклор литературных текстов (имеющих автора, например, Н. А. Клюева, А. К. Толстого, С. А. Есенина) [22]. Экспертом-филологом были обработаны 250 текстов и построены соответствующие теоретико-графовые модели, отражающие семантическую структуру текстов. Была проведена серия экспериментов с применением пяти методов интеллектуального анализа данных (дерево решений, дискриминантный анализ, метод опорных векторов, нейронная сеть, случайный лес) с использованием восьми признаков. Все методы показали достаточно высокую среднюю точность распознавания (более 80 %).

Модульная структура ИС позволяет проводить исследования на других размеченных корпусах. С ее помощью планируется осуществить углубленный ана-

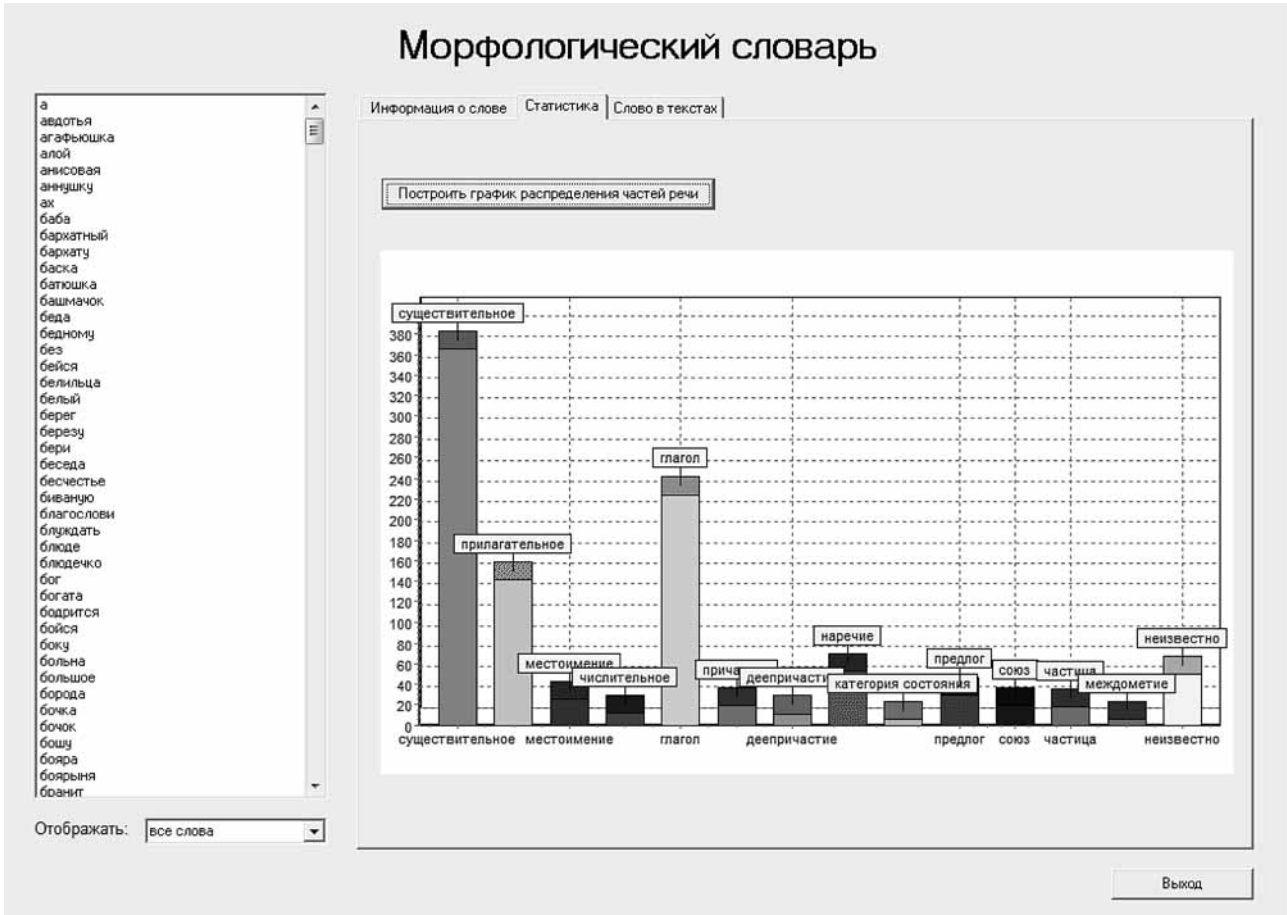

Рис. 3. График распределения частей речи в морфологическом словаре

лиз "графов сильных связей", построенных на корпусе текстов Ф. М. Достоевского (графы отражают закономерности распределения частей речи в рамках предложения). Ранее данная методика применялась при анализе древнерусских литературных источников [2].

Заключение

Представлены результаты исследований, которые направлены на разработку эффективных методов и средств для решения задачи атрибуции текстов. Одной из таких задач является анализ публицистических статей XIX в., которые приписывают Ф. М. Достоевскому (около 500 неатрибутированных текстов из журналов "Время", "Эпоха" и еженедельника "Гражданин"). Существующие программные системы не подходят для выполнения подобных исследований в силу ограниченности их инструментов и специфики задачи.

При поддержке модифицированного программного комплекса "СМАЛТ" и информационной системы "Фольклор", описание которых приведено в настоящей работе, планируется осуществить ввод, грамматическую разметку и последующий анализ указанных статей.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 18-012-90026.

Список литературы

1. **Малютов М. Б.** Обзор методов и примеров атрибуции текстов // Обозрение прикладной и промышленной математики. 2005. Т. 12. № 1. С. 41—78.

2. **Милов Л. В., Бородкин Л. И., Иванова Т. В.** и др. От Нестора до Фонвизина: новые методы определения авторства / Под ред. Л. В. Милова. М.: Прогресс, 1994. 445 с.

3. **Исхакова А. О.** Метод и программное средство определения искусственно созданных текстов: дис.... канд. техн. наук. Томск, 2016. 123 с.

4. **Ramnial H., Panchoo S., Pudaruth S.** Authorship Attribution Using Stylometry and Machine Learning Techniques // Intelligent Systems Technologies and Applications. Advances in Intelligent Systems and Computing. 2016. Vol. 384. P. 113—125.

5. **Романченко Т. Н.** Методы атрибуции в автороведческой экспертизе // Вестник Саратовской государственной юридической академии. 2013. № 2 (91). С. 228—233.

6. **Рогов А. А., Седов А. В., Сидоров Ю. В., Суровцова Т. Г.** Математические методы атрибуции текстов. Петрозаводск: Изд-во ПетрГУ, 2014. 96 с.

7. **Батура Т. В.** Формальные методы определения авторства текстов // Вестник Новосибирского государственного университета. Серия "Информационные технологии". 2012. Т. 10, № 4. С. 81—94.

8. **Артемов М. А., Владимиров А. Н., Селезнев К. Е.** Обзор систем анализа естественного текста на русском языке //

Вестник Воронежского государственного университета. Серия "Системный анализ и информационные технологии". 2013. № 2. С. 189—194.

9. **Котов А. А., Минеева З. И., Рогов А. А., Седов А. В., Сидоров Ю. В.** Лингвистические корпусы. Петрозаводск: Изд-во ПетрГУ, 2014. 140 с.

10. **Захаров В. Н., Леонтьев А. А., Рогов А. А., Сидоров Ю. В.** Программная система поддержки атрибуции текстов статей Ф. М. Достоевского // Труды Петрозаводского государственного университета. Серия "Прикладная математика и информатика". 2000. № 9. С. 113—122.

11. **Котов А. А., Некрасов М. Ю., Седов А. В., Рогов А. А.** Информационная система для создания размеченных корпусов малой размерности // Ученые записки Петрозаводского государственного университета. 2012. № 8—1. С. 108—112.

12. **Рогов А. А., Сидоров Ю. В., Суровцова Т. Г.** Математические методы атрибуции литературных текстов небольшого объема // Материалы XIII Всероссийской конференции "Математические методы в распознавании образов". М.: МАКС Пресс, 2007. С. 525—528.

13. **Романов А. С.** Методика и программный комплекс для идентификации автора неизвестного текста: дис.... канд. техн. наук. Томск, 2010. 149 с.

14. **Farringdon J. M.** Analyzing for Authorship. Cardiff: University of Wales Press, 1996. 324 p.

15. **Stamatatos E.** A Survey of Modern Authorship Attribution Methods // Journal of the American Society for Information Science and Technology, 2009. Vol. 60 (3). P. 538—556.

16. **Хетсо Г.** Принадлежность Достоевскому: К вопросу об атрибуции Ф. М. Достоевскому анонимных статей в журналах Время и Эпоха. Oslo: Solum Forlag A. S., 1986. 82 с.

17. **Сидоров Ю. В.** Математическая и информационная поддержка методов обработки литературных текстов на основе формально-грамматических параметров: дис.... канд. техн. наук. Петрозаводск, 2002. 127 с.

18. **Рогов А. А., Рогова О. Б.** Математическая модель для определения грамматических признаков в случае омонимии // Цифровые технологии в образовании, науке, обществе: материалы XII(2) Всероссийской научно-практической конференции (4—6 декабря 2018 г.). Петрозаводск, 2018. С. 196—198.

19. **Скабин А. В.** Математические модели, методы и алгоритмы дешифровки исторических стенограмм: дис.... канд. техн. наук. Петрозаводск, 2013. 101 с.

20. **Кулаков К. А., Седов А. В.** Разработка информационной системы для построения и анализа корпуса русских публицистических текстов XIX века СМАЛТ // Цифровые технологии в образовании, науке, обществе: материалы XII(2) Всероссийской научно-практической конференции (4—6 декабря 2018 года). Петрозаводск, 2018. С. 137—140.

21. **Москин Н. Д.** Теоретико-графовые модели фольклорных текстов и методы их анализа. Петрозаводск: Изд-во ПетрГУ, 2013. 148 с.

22. **Shchegoleva L., Lebedev A., Moskin N.** Recognition of Folklore Texts and Author's Poems Using Classification Trees and Neural Networks // Proceedings of the 22st Conference of Open Innovations Association FRUCT. Jyvaskyla, Finland, 2018. P. 418—420.

Software Support in Solving the Problem of Text Attribution

A. A. Rogov, rogov@petrsu.ru, **K. A. Kulakov**, kulakov@cs.karelia.ru,
N. D. Moskin, moskin@petrsu.ru, Petrozavodsk State University, Petrozavodsk, 185910, Russian Federation

Corresponding author:

Moskin Nikolai D., Assistant Professor, Petrozavodsk State University, Petrozavodsk, 185910, Russian Federation, E-mail: moskin@petrsu.ru

Received on February 24, 2019
Accepted on March 05, 2019

The article deals with the questions of digitization, markup and software support in solving the problem of attribution of texts. Effective solution of the actual problems of analyzing textual works, including searching for borrowings and determining authorship, requires combining both the work of a linguistic expert and the use of various mathematical methods on large amounts of data. Currently, one of the actual tasks is the analysis of 500 non-attributive publicistic articles from the magazines "Time", "Epoch" and the weekly "Citizen" (1861—1874). It is known that F. M. Dostoevsky edited and headed these magazines; therefore, research on the belonging of these works to his pen has been conducted for a long time. The work shows that modern information systems for attribution support do not allow to carry out such studies, so the staff of Petrozavodsk State University are working on the modernization of the "SMALT" software complex for implementing the text attribution task tool and the "Folklore" information system for automated construction and analysis of graph-theoretic models of texts, which will bring the research to a new level. The paper describing the architecture of the "SMALT", the modular structure of "Folklore" and a mathematical model for defining grammatical features in case of homonymy.

Keywords: *text attribution, text markup, data mining, software complex "SMALT", information system "Folklore", F. M. Dostoevsky*

Acknowledgements*: This work was supported by the Russian Foundation for Basic Research, project nos. 18-012-90026.*

For citation:

Rogov A. A., **Kulakov K. A.**, **Moskin N. D.** Software Support in Solving the Problem of Text Attribution, *Programmnaya Ingeneria,* 2019, vol. 10, no. 5, pp. 234—240.

DOI: 10.17587/prin.10.234-240

References

1. **Malyutov M. B.** Obzor metodov i primerov atribucii tekstov (Overview of methods and examples of text attribution), *Obozrenie prikladnoj i promyshlennoj matematiki,* 2005, vol. 12, no. 1, pp. 41—78 (in Russian).

2. **Milov L. V., Borodkin L. I., Ivanova T. V. et al.** *Ot Nestora do Fonvizina: novye metody opredeleniya avtorstva* (From Nestor to Fonvizin: new methods for determining authorship) / Edited by L. V. Milov. Moscow, Progress, 1994, 445 p. (in Russian).

3. **Iskhakova A. O.** *Metod i programmnoe sredstvo opredeleniya iskusstvenno sozdannyh tekstov* (Method and software for determining artificially created texts), Tomsk, 2016, 123 p. (in Russian).

4. **Ramnial H., Panchoo S., Pudaruth S.** Authorship Attribution Using Stylometry and Machine Learning Techniques, *Intelligent Systems Technologies and Applications. Advances in Intelligent Systems and Computing*, 2016, vol. 384, pp. 113—125.

5. **Romanchenko T. N.** Metody atribucii v avtorovedcheskoj ehkspertize (Attribution methods in the author's expertise), *Vestnik Saratovskoj gosudarstvennoj yuridicheskoj akademii*, 2013, no. 2 (91), pp. 228—233 (in Russian).

6. **Rogov A. A., Sedov A. V., Sidorov Y. V., Surovcova T. G.** *Matematicheskie metody atribucii tekstov* (Mathematical methods of text attribution). Petrozavodsk, PetrSU Publ., 2014, 96 p. (in Russian).

7. **Batura T. V.** Formalnye metody opredeleniya avtorstva tekstov (Formal methods for determining the authorship of texts), *Vestnik Novosibirskogo gosudarstvennogo universiteta. Series "Information Technology",* 2012, vol. 10, no. 4, pp. 81—94 (in Russian).

8. **Artemov M. A., Vladimirov A. N., Seleznev K. E.** Obzor sistem analiza estestvennogo teksta na russkom yazyke (Overview of natural text analysis systems in Russian), *Vestnik Voronezhskogo gosudarstvennogo universiteta. Series "System analysis and information technology"*, 2013, no. 2, pp. 189—194 (in Russian).

9. **Kotov A. A., Mineeva Z. I., Rogov A. A., Sedov A. V., Sidorov Y. V.** *Lingvisticheskie korpusy* (Linguistic Corpuses), Petrozavodsk, PetrSU Publ., 2014, 140 p. (in Russian).

10. **Zaharov V. N., Leontev A. A., Rogov A. A., Sidorov Y. V.** Programmnaya sistema podderzhki atribucii tekstov statej F. M. Dostoevskogo (Software support system for the attribution of texts of articles by F. M. Dostoevsky), *Trudy Petrozavodskogo gosudarstvennogo universiteta. Series "Applied Mathematics and Computer Science"*, 2000. no. 9. pp. 113—122. (in Russian).

11. **Kotov A. A., Nekrasov M. Y., Sedov A. V., Rogov A. A.** Informacionnaya sistema dlya sozdaniya razmechennyh korpusov maloj razmernosti (Information system for creating small-sized marked corpuses), *Uchenye zapiski Petrozavodskogo gosudarstvennogo universiteta,* 2012, no. 8—1, pp. 108—112 (in Russian).

12. **Rogov A. A., Sidorov Y. V., Surovcova T. G.** Matematicheskie metody atribucii literaturnyh tekstov nebolshogo obema (Mathematical methods for the attribution of small literary texts), *Proceedings of the XIII Conference "Matematicheskie metody v raspoznavanii obrazov*", Moscow, MAKS Press, 2007, pp. 525—528 (in Russian).

13. **Romanov A. S.** *Metodika i programmnyj kompleks dlya identifikacii avtora neizvestnogo teksta* (Methods and software for identifying the author of an unknown text), Tomsk, 2010, 149 p. (in Russian).

14. **Farringdon J. M.** *Analyzing for Authorship*, Cardiff, University of Wales Press, 1996, 324 p.

15. **Stamatatos E.** A Survey of Modern Authorship Attribution Methods, *Journal of the American Society for Information Science and Technology*, 2009, vol. 60 (3), pp. 538—556.

16. **Kjetsaa G.** *Attributed to Dostoevsky: The Problem of attributing to Dostoevsky anonymous articles in Time and Epoch*, Oslo, Solum Forlag A. S., 1986, 82 p.

17. **Sidorov Y. V.** *Matematicheskaya i informacionnaya podderzhka metodov obrabotki literaturnyh tekstov na osnove formalno-grammaticheskih parametrov* (Mathematical and informational support of literary text processing methods based on formal grammatical parameters), Petrozavodsk, 2002, 127 p. (in Russian).

18. **Rogov A. A., Rogova O. B.** Matematicheskaya model dlya opredeleniya grammaticheskih priznakov v sluchae omonimii (Mathematical model for determining grammatical attributes in the case of homonymy), *Proceedings of the XII(2) Conference "Cifrovye tekhnologii v obrazovanii, nauke, obshchestve"* (December 4—6, 2018), Petrozavodsk, 2018, pp. 196—198 (in Russian).

19. **Skabin A. V.** *Matematicheskie modeli, metody i algoritmy deshifrovki istoricheskih stenogramm* (Mathematical models, methods and algorithms for decoding historical transcripts), Petrozavodsk, 2013. 101 p. (in Russian).

20. **Kulakov K. A., Sedov A. V.** Razrabotka informacionnoj sistemy dlya postroeniya i analiza korpusa russkih publicisticheskih tekstov XIX veka SMALT (Development of an information system for the construction and analysis of the corpus of Russian publicistic texts of the XIX century SMALT), *Proceedings of the XII(2) Conference "Cifrovye tekhnologii v obrazovanii, nauke, obshchestve"* (December 4—6, 2018). Petrozavodsk, 2018, pp. 137—140 (in Russian).

21. **Moskin N. D.** *Teoretiko-grafovye modeli folklornyh tekstov i metody ih analiza* (Graph-theoretic models of folklore texts and methods of their analysis), Petrozavodsk, PetrSU Publ., 2013, 148 p. (in Russian).

22. **Shchegoleva L., Lebedev A., Moskin N.** Recognition of Folklore Texts and Author's Poems Using Classification Trees and Neural Networks, *Proceedings of the 22st Conference of Open Innovations Association FRUCT*, Jyvaskyla, Finland, 2018, pp. 418—420.

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4
Технический редактор *Е. М. Патрушева.* Корректор *З. В. Наумова*

Сдано в набор 06.03.2019 г. Подписано в печать 25.04.2019 г. Формат 60×88 1/8. Заказ PI519
Цена свободная.

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