ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

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

Редакционный совет
Садовничий В.А., акад. РАН (*председатель*)
Бетелин В.Б., акад. РАН
Васильев В.Н., чл.-корр. РАН
Жижченко А.Б., акад. РАН
Макаров В.Л., акад. РАН
Панченко В.Я., акад. РАН
Стемпковский А.Л., акад. РАН
Ухлинов Л.М., д.т.н.
Федоров И.Б., акад. РАН
Четверушкин Б.Н., акад. РАН

Главный редактор
Васенин В.А., д.ф.-м.н., проф.

Редколлегия
Антонов Б.И.
Афонин С.А., к.ф.-м.н.
Бурдонов И.Б., д.ф.-м.н., проф.
Борзовс Ю., проф. (Латвия)
Гаврилов А.В., к.т.н.
Галатенко А.В., к.ф.-м.н.
Корнеев В.В., д.т.н., проф.
Костюхин К.А., к.ф.-м.н.
Махортов С.Д., д.ф.-м.н., доц.
Манцивода А.В., д.ф.-м.н., доц.
Назиров Р.Р., д.т.н., проф.
Нечаев В.В., д.т.н., проф.
Новиков Б.А., д.ф.-м.н., проф.
Павлов В.Л. (США)
Пальчунов Д.Е., д.ф.-м.н., доц.
Петренко А.К., д.ф.-м.н., проф.
Позднеев Б.М., д.т.н., проф.
Позин Б.А., д.т.н., проф.
Серебряков В.А., д.ф.-м.н., проф.
Сорокин А.В., к.т.н., доц.
Терехов А.Н., д.ф.-м.н., проф.
Филимонов Н.Б., д.т.н., проф.
Шапченко К.А., к.ф.-м.н.
Шундеев А.С., к.ф.-м.н.
Щур Л.Н., д.ф.-м.н., проф.
Язов Ю.К., д.т.н., проф.
Якобсон И., проф. (Швейцария)

Редакция
Лысенко А.В., Чугунова А.В.

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.

© Издательство "Новые технологии", "Программная инженерия", 2019

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

Vol. 10
N 11-12
2019

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

Editorial Council:

SADOVNICHY V. A., Dr. Sci. (Phys.-Math.), Acad. RAS (*Head*)
BETELIN V. B., Dr. Sci. (Phys.-Math.), Acad. RAS
VASIL'EV V. N., Dr. Sci. (Tech.), Cor.-Mem. RAS
ZHIZHCHENKO A. B., Dr. Sci. (Phys.-Math.), Acad. RAS
MAKAROV V. L., Dr. Sci. (Phys.-Math.), Acad. RAS
PANCHENKO V. YA., Dr. Sci. (Phys.-Math.), Acad. RAS
STEMPKOVSKY A. L., Dr. Sci. (Tech.), Acad. RAS
UKHLINOV L. M., Dr. Sci. (Tech.)
FEDOROV I. B., Dr. Sci. (Tech.), Acad. RAS
CHETVERTUSHKIN B. N., Dr. Sci. (Phys.-Math.), Acad. RAS

Editor-in-Chief:

VASENIN V. A., Dr. Sci. (Phys.-Math.)

Editorial Board:

ANTONOV B.I.
AFONIN S.A., Cand. Sci. (Phys.-Math)
BURDONOV I.B., Dr. Sci. (Phys.-Math)
BORZOVS JURIS, Dr. Sci. (Comp. Sci), Latvia
GALATENKO A.V., Cand. Sci. (Phys.-Math)
GAVRILOV A.V., Cand. Sci. (Tech)
JACOBSON IVAR, Dr. Sci. (Philos., Comp. Sci.), Switzerland
KORNEEV V.V., Dr. Sci. (Tech)
KOSTYUKHIN K.A., Cand. Sci. (Phys.-Math)
MAKHORTOV S.D., Dr. Sci. (Phys.-Math)
MANCIVODA A.V., Dr. Sci. (Phys.-Math)
NAZIROV R.R. , Dr. Sci. (Tech)
NECHAEV V.V., Cand. Sci. (Tech)
NOVIKOV B.A., Dr. Sci. (Phys.-Math)
PAVLOV V.L., USA
PAL'CHUNOV D.E., Dr. Sci. (Phys.-Math)
PETRENKO A.K., Dr. Sci. (Phys.-Math)
POZDNEEV B.M., Dr. Sci. (Tech)
POZIN B.A., Dr. Sci. (Tech)
SEREBRJAKOV V.A., Dr. Sci. (Phys.-Math)
SOROKIN A.V., Cand. Sci. (Tech)
TEREKHOV A.N., Dr. Sci. (Phys.-Math)
FILIMONOV N.B., Dr. Sci. (Tech)
SHAPCHENKO K.A., Cand. Sci. (Phys.-Math)
SHUNDEEV A.S., Cand. Sci. (Phys.-Math)
SHCHUR L.N., Dr. Sci. (Phys.-Math)
YAZOV Yu. K., Dr. Sci. (Tech)

Editors: LYSENKO A.V., CHUGUNOVA A.V.

CONTENTS

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

УДК 004.4'233 DOI: 10.17587/prin.10.419-429

В. А. Галатенко, д-р физ.-мат. наук, зав. сектором, galat@niisi.ras.ru,
К. А. Костюхин, канд. физ.-мат. наук, ст. науч. сотр., kost@niisi.ras.ru,
Научно-исследовательский институт системных исследований Российской академии наук (НИИСИ РАН), Москва

Автоматический ремонт программ: сравнительный анализ подходов*

Статья является второй частью работы, посвященной автоматическому ремонту программ (см. "Программная инженерия". 2019. Т. 10, № 7—8. С. 291—296). Работа состоит в проведении сравнительного анализа идей и подходов к автоматическому ремонту программ. К сожалению, программа — весьма неудобный объект для обработки, и это создает принципиальные трудности для реализации автоматического ремонта. Ряд предлагаемых подходов не имеет теоретического обоснования, а практические результаты их применения оказываются неудовлетворительными.

Авторы попытались рассмотреть основные существующие подходы, преимущества и недостатки каждого из них.

Ключевые слова: *отладка, ремонт программ, восстановление программ, воспроизведение выполнения, самолечение, восстановление данных*

Используемая терминология

Вначале напомним основные понятия предметной области "ремонт программ".

Неисправность (*failure*) — наблюдаемое неприемлемое поведение.

Отклонение (*error*) — нарастание некорректного состояния до неисправности (остающееся до времени незамеченным).

Дефект (*fault*) — первопричина ошибки (в частности, некорректный программный код).

Ошибка (*bug*) — собирательный термин для неисправности, отклонения и дефекта, расхождение между ожидаемым и наблюдаемым поведением программы. Отметим, что здесь в роли наблюдателя может выступать как человек, так и некая аппаратно-программная сущность.

Класс ошибок (*bug class*) — абстрактное понятие, обозначающее семейство ошибок, имеющих какие-либо общие характеристики (например, одинаковые проявления, одинаковые первопричины или одинаковые способы исправления).

Ремонт (*repair*) — преобразование неприемлемого поведения программы в приемлемое, соответствующее спецификациям, осуществляемое при помощи оператора вручную или автоматически с использованием специальных программ. В последнем случае говорят об **автоматическом ремонте** (*automatic repair*).

Ремонт поведения (*behavioral repair*) — изменение кода ремонтируемой программы и, тем самым, изменение ее последующего поведения. Может изменяться исходный или исполняемый код, эти изменения могут проводиться как в оперативном, так и в автономном режимах.

Программная коррекция, заплата (*patch*) — исправление, вносимое в программу.

Переподгонка (*overfitting*) — обеспечение положительного результата всех тестов из заданного тестового набора, возможно, в ущерб другим тестам или неявным спецификациям.

Ремонт состояния (*state repair*) — изменение состояния ремонтируемой программы и/или ее окружения. Ремонт состояния проводится в оперативном режиме, во время выполнения программы и может заключаться в перезагрузке, откатке, изменении конфигурации и т. п.

Оснащение программы (*program equipment*) — модификация исходного или бинарного кода программы для обеспечения возможности воздействия на нее внешних аппаратно-программных средств.

Ремонт использования прикладных программных интерфейсов

Около половины реальных ошибок требует исправлений, характерных для прикладных программных интерфейсов. В работе [1] предложен общий подход к решению этой задачи.

В качестве отправной точки используется пример с интерфейсом Calendar (листинг 1). Ошибка в первоначальном варианте программы состояла в том, что

* Результаты исследований, представленные в публикации, выполнены в рамках государственного задания по проведению фундаментальных научных исследований по теме (проекту) "38. Проблемы создания глобальных и интегрированных информационно-телекоммуникационных систем и сетей, развитие технологий и стандартов GRID. Исследование и реализация программной платформы для перспективных многоядерных процессоров (0065-2019-0002)."

форматировалось устаревшее время. Для обновления времени требовалось добавить вызов `getTime ()`. Система ремонта поведения может сделать это, только если она обладает знаниями о конкретном прикладном программном интерфейсе. Подобные знания в работе [1] предлагается представлять в виде схем программ, задающих состав и последовательность действий.

```
  public StringBuffer format (Calendar
vCalendar, StringBuffer buf) {
    if (mTimeZoneForced) {
      vCalendar.getTime (); // Исправление:
устанавливает в объекте текущее время
      vCalendar= (Calendar) vCalendar.clone ();
      vCalendar.setTimeZone (mTimeZone);
    }
    ...
  }
```

Листинг 1. Пример ремонта использования прикладного программного интерфейса

К сожалению, в документации к программным интерфейсам необходимые схемы отсутствуют. В работе [1] предложено обратиться к проектам с открытыми исходными текстами и попытаться найти в них правильные последовательности действий. Далее требуется подставить в вызовы нужные фактические параметры. Этот этап пока не удалось автоматизировать, а значит, генерация программной коррекции проводится в ручном режиме.

Таким образом, в работе [1] заявлен подход, но отсутствует решение, так как подход носит слишком общий характер. Посредством схем программ невозможно специфицировать все корректные последовательности вызовов в рамках прикладного программного интерфейса, не говоря уже об их параметрах. Анализ фрагмента кода на листинге 1 показывает, что вызов `getTime ()` нужен только тогда, когда требуется отформатировать текущее время, а не время, переданное в качестве параметра. Однако системы ремонта поведения не могут различить эти ситуации, а тесты могут оказаться бесполезными, поскольку разница во времени может быть минимальной. Иными словами, никаких преимуществ по сравнению с универсальными системами ремонта поведения в данном случае не наблюдается.

Более конкретная постановка задачи рассмотрена в работе [2] — ремонт использования строковых (*String*) прикладных программных интерфейсов в программах на языке Java.

Известно, при каких условиях и какие исключительные неблагоприятные ситуации могут инициировать строковые объекты. Основная идея системы CLOTHO, которая описана в работе [2], состоит в том, чтобы посредством статического анализа выявлять потенциально опасные использования строковых интерфейсов и накладывать в этих местах заплаты, оформленные как обработчики исключительных ситуаций.

На листинге 2 приведен пример небезопасного оператора. Подстроку из имени файла нельзя выделить, если `prefix > index + separatorAdd`.

```
  return filename.substring (prefix, index +
separatorAdd);
```

Листинг 2. Пример небезопасного вызова строкового метода

Система CLOTHO вместо этого оператора сгенерирует заплату, показанную на листинге 3.

```
  String temp = null;
  try {
    temp = filename.substring (prefix, index +
separatorAdd);
  } catch (IndexOutOfBoundsException ex) {
    int length = filename.length;
    int t = index + separatorAdd;
    temp = filename.substring (getStart
(prefix, t, length), getEnd (prefix, t, length));
  }
  return temp;
```

Листинг 3. Пример программной коррекции, сгенерированной системой CLOTHO

Программные коррекции, генерируемые системой CLOTHO, активируются только в исключительных ситуациях, т. е. не влияют на нормальную работу и минимизируют отклонения от ожидаемого поведения.

Как показали эксперименты, система CLOTHO сумела сгенерировать заплаты для 25 из 30 рассматриваемых дефектов в реальных проектах с открытыми исходными текстами. Эти заплаты семантически близки исправлениям, сделанным разработчиками позднее. Данный результат следует оценить как высокий, а подход, реализованный в CLOTHO, как перспективный. Ориентация на конкретный прикладной программный интерфейс позволила построить формальную модель его поведения в виде конечного автомата, разделяющего переходы между состояниями на безопасные и опасные, и тем самым локализовать подозрительные места в коде. Для уточнения результатов статического анализа в CLOTHO применяется динамический анализ. Все это способствует высокому качеству программных коррекций.

Общей частью ремонта использования (почти) всех прикладных программных интерфейсов является устранение обращений по пустому указателю. Эта задача рассмотрена в работе [3]. Предложено два способа ремонта: замена на обращение по другому указателю или пропуск опасного обращения. Это еще один пример реализации подхода "порождай и проверяй", когда корректность заплаты оценивается только с помощью тестов со всеми присущими данному подходу недостатками.

Ремонт обработки ошибочных ситуаций

Некорректная обработка ошибочных ситуаций является одной из главных причин уязвимостей в программах. Типичный пример — отсутствие проверки

результата после вызова malloc (), следствием чего может стать обращение к памяти по пустому указателю.

Отсутствие в языке программирования С стандартных средств подобной обработки приводит к тому, что в каждом проекте реализуется своя дисциплина, а от разработчиков требуется следовать ей с безупречной точностью, не упуская ни одной детали, что крайне сложно и утомительно.

В работе [4] поставлена и решена задача выявления и исправления дефектов обработки ошибочных ситуаций. Подобный дефект имеет место, если функция *F*1 вызывает функцию *F*2, которая по какой-либо причине терпит неудачу, возвращая соответствующий код ошибки, а *F*1 не может этот код надлежащим образом обработать.

В первой части работы [4] построена классификация дефектов обработки. Выделены четыре класса:

- некорректная/отсутствующая проверка на ошибочность ситуации (*error checks*, EC);
- некорректное/отсутствующее распространение информации об ошибочной ситуации (*error propagation*, EP);
- некорректное/отсутствующее протоколирование ошибочной ситуации (*error outputs*, EO);
- некорректное/отсутствующее освобождение ресурсов (*resource release*, RR).

Анализ истории шести проектов с открытыми исходными текстами показал, что самыми частыми являются дефекты EC и RR.

Во второй части работы [4] описана система ErrDoc (*error doctor*).

Введено понятие маршрута с ошибкой. Это путь в графе управления программы, содержащий ветвь, на которой какая-либо вызываемая функция фиксирует ошибочную ситуацию. Обработчик ошибочной ситуации определяется как фрагмент маршрута с ошибкой, осуществляющий проверку и обработку кода возврата.

Работу системы ErrDoc можно подразделить на три этапа. На первом этапе, который можно назвать подготовительным, для проверяемой пары вызывающей и вызываемой функций выявляются маршруты с ошибкой и выделяются обработчики ошибочных ситуаций. ErrDoc использует каркас статического анализа Clang, точнее, его средства символьного выполнения, анализа зависимостей по данным и обработки абстрактных синтаксических деревьев. Эти средства дополняются эвристикой: обработчик ошибочной ситуации — это более короткая ветвь условного оператора, проверяющего код завершения функции.

Второй этап — определение места и категорирование дефектов обработки ошибочных ситуаций. Чтобы выявить некорректное/отсутствующее освобождение ресурсов, в ErrDoc применяется эвристика, направленная на определение пары захват/освобождение. Предполагается, что захват должен происходить до обращения к вызываемой функции, а освобождение — после. Отметим, что это непростое действие, поскольку кроме стандартных пар malloc/free и lock/unlock могут существовать пары, определенные разработчиком, а одной функции захвата (например, CRYPTO_malloc ()) может соответствовать несколько функций освобождения (например, CRYPTO_free () и CRYPTO_clear_free ()), и наоборот. Возможно, предпочтительнее определять перечень потенциальных пар вручную.

Исправления дефектов обработки ошибочных ситуаций, осуществляемые в рамках третьего этапа работы системы ErrDoc, должны следовать дисциплине, принятой в ремонтируемом проекте, поэтому ErrDoc не только выявляет место и характер дефекта, но и отыскивает существующие обработчики и использует их код для программных коррекций.

Система ErrDoc не свободна от пропусков ошибок и ложных срабатываний. Однако как показали эксперименты, результаты которых приведены в работе [4], число их невелико, а генерируемые программные коррекции, как правило, принимаются разработчиками.

Можно констатировать, что реалистичная постановка задачи позволила дополнить существующие средства статического анализа удачными эвристиками и обеспечить высокое качество ремонта, проводимого системой ErrDoc.

Ремонт поведения целочисленных арифметических вычислений

Законы машинной арифметики отличаются от привычных математических. Это требует от разработчиков соответствующих знаний и аккуратности при программировании вычислений. Дополнительную сложность создают следующие обстоятельства:

- представление чисел и дисциплина обработки исключительных ситуаций зависят от аппаратной платформы;
- правила приведения типов, стандартизованные для языков программирования C/C++, довольно сложны, особенно применительно к учету знаковости;
- целочисленное переполнение может не возбуждать исключительную ситуацию и не влиять на передачу управления, что затрудняет спектральную локализацию дефектов;
- для исправления дефектов могут потребоваться коррекции не только выполняемых операторов, но и описаний, а это нетипично для систем автоматического ремонта поведения.

В то же время любой дефект в реализации целочисленной арифметики чреват не только неверным результатом, но и появлением уязвимостей, зачастую критических. Например, потенциальная полная потеря мощности в Boeing 787 Dreamliners стала следствием знакового переполнения 32-битного счетчика. Значит, исправление подобных дефектов желательно осуществлять аккуратно и быстро.

Важная и сложная проблема автоматического ремонта поведения целочисленной арифметики стала предметом работы [5], в которой описана система IntPTI, ориентированная на ремонт программ на языке С.

Ключевой элемент системы IntPTI — вывод надлежащих типов для целочисленных переменных и выражений, чтобы новый тип обеспечивал представление всех возможных значений переменной или выражения.

Диапазон возможных значений определяется средствами статического анализа. Используются интервальный анализ с учетом семантики библиотечных функций, анализ указателей и анализ потоков данных.

Задача вывода надлежащих типов формулируется как задача частичной взвешенной максимальной выполнимости формул в теориях (*partial weighted MaxSMT*). Ограничения на надлежащие типы подразделяются на жесткие и мягкие. Пример мягкого ограничения — желательность сохранения прежних типов. За нарушение мягкого ограничения предусматривается штраф. Отыскивается решение, при котором выполняются все жесткие ограничения и минимизируется общий штраф за нарушение мягких ограничений.

Для генерации программных коррекций, осуществляемой SMT-решателем, используются шаблоны преобразований, выделенные путем анализа 668 известных ошибок целочисленной арифметики и их исправлений. Самым употребительным шаблоном является санитарная проверка, направленная на отбраковку заведомо неверных результатов (например, проверка того, что сумма положительных чисел осталась положительной). Другие шаблоны — явное преобразование типа выражения, изменение декларированного типа переменной (эти два преобразования расширяют множество представимых значений), а также преобразование выражения (в попытке избежать переполнения без расширения множества представимых значений).

Система IntPTI была проверена на семи проектах с открытыми исходными текстами. Из 25 известных ошибок целочисленной арифметики удалось исправить 23.

Ремонт поведения динамического управления памятью

В программах на языке C выделение и освобождение памяти возлагается на программиста и проводится низкоуровневыми средствами — вызовами библиотечных функций malloc () и free (). Это чревато ошибками и появлением уязвимостей, нередко критических. Так, утечки памяти, т. е. отсутствие освобождения ставших ненужными блоков, позволяет организовать атаку на доступность. Эффект от использования ранее освобожденного блока может быть непредсказуемым. Следовательно, ошибки динамического управления памятью весьма вероятны, находить и исправлять их желательно оперативно, делать это нужно аккуратно, не внося новых ошибок.

В то же время поиск и устранение подобных ошибок — сложная задача даже для опытных программистов, а у универсальных систем автоматического ремонта поведения вообще нет шансов это сделать. Причина в том, что практически невозможно построить тестовый набор, выявляющий все такие ошибки, равно как сформулировать пред- и постусловия корректности. Отсутствие ошибок динамического управления памятью — это глобальное свойство программы, здесь нет определенного места дефекта, который достаточно исправить. Требуется проанализировать всю программу, все возможные пути выполнения и все возможные значения указателей на блоки динамической памяти. Какие-то надежды можно связывать только со специализированными системами ремонта.

Возможные ошибки динамического управления памятью можно подразделить на три категории:

- утечки памяти;
- повторное освобождение памяти;
- использование ранее освобожденного блока памяти (использование после освобождения).

LeakFix [6] стала, вероятно, первой специализированной системой автоматического ремонта поведения динамического управления памятью. Более точно — она решает задачу выявления и безопасного (т. е. без внесения новых ошибок) устранения утечек памяти. С формальной точки зрения этого достаточно, поскольку, если убрать из программы все вызовы free (), повторных освобождений и использований после освобождения заведомо не будет, так что останется только безопасным образом устранить утечки.

Чтобы устранить утечки памяти, система LeakFix вставляет вызовы free (), удовлетворяющие следующим условиям:

- при любом маршруте выполнения блок памяти резервируется до освобождения;
- блок памяти освобождается не более одного раза;
- блок памяти не используется после освобождения.

Выполнение этих условий обеспечивает безопасность исправлений. Проблема в том, чтобы найти подходящие места и подходящие указатели, передаваемые как аргументы при вызовах free ().

Одной из основных составляющих процесса ремонта поведения динамического управления памятью является анализ указателей. В системе LeakFix используются готовые средства статического анализа указателей, присутствующие в инфраструктуре LLVM, которые при необходимости можно заменить. Отметим, что важным продвижением в статическом анализе указателей явилось использование логики разделения (называемой также сепарационной логикой — Separation Logic), положенной в основу еще одной системы ремонта поведения динамического управления памятью FootPatch [7].

Помимо обеспечения безопасности исправлений, желательно осуществлять освобождение памяти как можно раньше, как только блок памяти перестал быть нужным. Это условие также учитывается в системе LeakFix. Освобождать блок памяти нужно в функции, в которой он (прямо или косвенно) резервируется, но указатель на него не передается вовне. Если этого не сделать, будет иметь место утечка памяти. В этой функции нужно найти место, где можно построить выражение, результатом вычисления которого всегда является указатель на рассматриваемый блок памяти, и где освобождение было бы безопасным.

Результаты применения системы LeakFix к реальным программам можно оценить как неровные. В некоторых случаях устраняются все известные

утечки. В то же время применительно к gcc были устранены лишь 2 утечки из 44. Типичным для gcc является фрагмент кода, показанный на листинге 4, где присутствует условное резервирование блока памяти. Подобному резервированию должно соответствовать условное освобождение, сгенерировать которое LeakFix не может.

```
char p = ...
if (...) {
   p = malloc (...);
   ...
}
// Использование p
```

Листинг 4. Пример условного резервирования блока динамической памяти

Направление автоматического ремонта поведения динамического управления памятью получило дальнейшее развитие в системе MemFix [8], которая способна исправлять не только утечки памяти, но также повторное освобождение и использование после освобождения.

Нахождение множества операторов освобождения памяти сводится в MemFix к известной математической задаче точного покрытия, для решения которой используется SAT — решатель. Здесь "покрытие" означает устранение утечек, а "точное" — отсутствие повторных освобождений. Использование проверенных средств статического анализа гарантирует правильность генерируемых программных коррекций.

Приведем пример работы системы MemFix. На листинге 5 показан фрагмент программы с ошибкой — повторным освобождением.

```
p = malloc(1);
if (...) {
   q = malloc(1);
} else
   q = p;
// Использование q
free(q);
free(p); // Повторное освобождение, если выполнялась else-часть
```

Листинг 5. Фрагмент С-кода с ошибкой — повторным освобождением памяти

Система MemFix исправила ошибку так, как показано на листинге 6.

```
p = malloc(1);
if (...) {
   q = malloc(1);
   free(p);
} else
   q = p;
// Использование q
free(q);
```

Листинг 6. Результат работы системы MemFix

Рассмотрим еще один пример из работы [8] (листинги 7 и 8), в котором, помимо динамического управления памятью, присутствует обработка ошибочных ситуаций. Если какой-либо из вызовов malloc(2) на листинге 7 завершится неудачей, то после перехода на метку `err` произойдет повторное освобождение блока памяти. Отметим, что разработчики-люди смогли отремонтировать программу только с третьей попытки и ушло на это десять месяцев.

```
in = malloc(1);
out = malloc(1);
// Использование in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {
   goto err;
}
out = malloc(2);
if (out == NULL) {
   free(in);
   goto err;
}
// Использование in, out
err:
free(in);
free(out);
```

Листинг 7. Фрагмент С-кода с двумя ошибками — повторным освобождением

```
in = malloc(1);
out = malloc(1);
// Использование in, out
free(in);

in = malloc(2);
if (in == NULL) {
   goto err;
}
free(out);
out = malloc(2);
if (out == NULL) {
   goto err;
}
// Использование in, out
err:
free(in);
free(out);
```

Листинг 8. Результат работы системы MemFix

Система MemFix была испытана на трех тестовых наборах. Из Juliet Test Suite (небольшие, но разнообразные программы с известными уязвимостями) было выбрано подмножество с утечками памяти, повторными освобождениями и использованием после освобождения. В качестве второго тестового набора выступили утилиты GNU Coreutils*, в качестве третьего набора — небольшие модельные программы, построенные авторами

* https://www.gnu.org/software/coreutils/

работы [8] на основе проектов с открытыми исходными текстами. Чтобы сравнить MemFix и LeakFix, для LeakFix убирали все обращения к free () и подсчитывали число устраненных утечек памяти.

На наборе Juliet Test Suite система MemFix показала абсолютный результат, исправив все 210 ошибок (LeakFix — 137). В GNU Coreutils результативность MemFix составила 50 % (исправлены 12 из 24 программ). Система LeakFix сгенерировала заплаты для трех программ, две из которых были исправлены лишь частично. Из 100 модельных программ, каждая из которых содержала по одной ошибке, MemFix удалось исправить 37. Для LeakFix результаты получить не удалось в силу нестабильной работы системы, в некоторых случаях генерировавшей очевидно некорректные заплаты.

Таким образом, результаты MemFix лучше, чем у LeakFix, но и их трудно назвать удовлетворительными. Система MemFix имеет ряд принципиальных ограничений, делающих проблематичным ее практическое использование.

Первое из ограничений состоит в том, что для исправления дефектов используются только вставки и удаления вызовов free (). В частности, MemFix не может сгенерировать условное освобождение памяти. Не справляется MemFix и с неявным освобождением, осуществляемым функцией realloc (). Далее, во многих программах используются зависящие от приложения обертки для функций управления динамической памятью, например, CRYPTO_malloc (), CRYPTO_free (), CRYPTO_clear_free () и т. п., распознать и использовать которые MemFix не может.

Второе ограничение проистекает из ограничений используемых средств статического анализа, которые не различают отдельные элементы массива указателей. Соответственно, не могут быть исправлены дефекты при работе с такими указателями. Сложности вызывают и связанные структуры данных, такие как списки и деревья.

Третье ограничение касается масштабируемости. Плохо масштабируется как статический анализ, так и решение задачи точного покрытия, которая, как известно, является NP-полной.

Не ясно, как перечисленные ограничения можно преодолеть.

Ремонт поведения параллельных программ

Параллельное программирование сопряжено с целым рядом новых сложных проблем. Одна из них — взаимная блокировка потоков управления (тупики), ставшая темой исследования [9]. Рассматриваются многопотоковые программы на языке C, следующие стандарту POSIX и использующие для блокировок примитивы вида `lock ()` и/или `trylock ()`. На листинге 9 приведен пример программы, приводящей к тупику.

```
// Поток управления t1:
pthread _ mutex _ lock (&mtx1);
pthread _ mutex _ lock (&mtx2);
// Критический интервал
...

// Поток управления t2:
pthread _ mutex _ lock (&mtx2);
pthread _ mutex _ lock (&mtx1);
// Критический интервал
...
```

Листинг 9. Пример программы, приводящей к тупику

Тупик возможен, если по очереди устанавливается несколько блокировок и между ними существует циклическая зависимость. В приведенном примере поток `t1` сначала пытается установить блокировку `l1`, потом `l2`, а потом `t2` действует в обратном порядке — сначала `l2`, поток `l1`. Если поток `t1` сумеет установить блокировку `l1`, а `t2` — блокировку `l2`, возникнет тупиковая ситуация.

Один из возможных способов (разумеется, не единственный и, вероятно, не лучший) ремонта подобного поведения — замена вызова `lock ()` на `trylock ()`, который не блокируется, а в случае занятости ресурса возвращает код ошибки, после чего следует освободить ранее занятые ресурсы и дать шанс захватить их другим потокам. Соответствующее исправление показано на листинге 10.

```
// Поток управления t1:
pthread _ mutex _ lock (&mtx1);
pthread _ mutex _ lock (&mtx2);
// Критический интервал
...

// Поток управления t2:
while (true) {
   pthread _ mutex _ lock (&mtx2);
   if (pthread _ mutex _ trylock (&mtx1) != 0)
     pthread _ mutex _ unlock (&mtx2);
}
// Критический интервал
...
```

Листинг 10. Исправленный вариант программы (без тупика)

Для поиска циклических зависимостей блокировок в многопотоковых программах, написанных на языке C и удовлетворяющих стандарту POSIX, рассматриваемая система автоматического ремонта поведения использует средства статического анализа. Затем для синтеза программных коррекций формулируется задача взвешенной частичной максимальной выполнимости (*weighted partial maximum satisfiability*, wpMAX-SAT). Кроме жестких ограничений на отсутствие циклических зависимостей в ней фигурируют мягкие ограничения, способствующие уменьшению сложности коррекций, желательности замены `trylock ()` на `lock ()`, а не наоборот (чтобы не надо было программировать освобождение ресурсов) и т. п.

Рассматриваемой системе присущи многочисленные ограничения, как фундаментальные, так и технические.

Фундаментальное ограничение состоит в том, что задача определения того, свободна ли программа от тупиков, алгоритмически неразрешима (в том числе потому, что сводится к проблеме завершимости).

Далее, изначально установленное ограничение на способы ремонта поведения исключительно путем замены `lock ()` на `trylock ()` или наоборот, `trylock ()` на `lock ()` не позволяет исправить даже фрагмент (однопотоковой) программы, показанный на листинге 11, где делается попытка дважды заблокировать один и тот же ресурс без промежуточного освобождения.

```
lock (l1)
... // l1 не освобождается
lock (l1) // тупик
...
```

Листинг 11. Пример программы с самоблокировкой

Также фундаментальными следует признать ограничения на точность анализа по выявлению объектов для блокировки, если к ним обращаются по указателям.

Основным техническим ограничением является сложность программирования освобождения ресурсов после неудачного завершения `trylock ()`. Если предыдущие блокировки делались далеко от места `trylock ()` и/или в них участвовали разделяемые переменные, то "откатиться" к состоянию без блокировок может и не получиться. Кроме того, предполагается, что каждый поток, если ему не мешают, в конце концов снимает все свои блокировки. "Утечки блокировок" не ремонтируются.

Первопричина тупиков — неатомарность захвата нескольких ресурсов. Некоторые системы ремонта поведения пытаются справиться с этим, оформляя последовательность захватов как критический интервал, однако для этого нужно вводить новые блокировки, которые могут привести к новым тупикам. Система DFixer [10] действует по-другому. Она выбирает один из потоков управления, участвующих в тупике, а также пару блокировок, таких, что ожидание установки второй из них при установленной первой вызывает тупик. После этого вторая блокировка "поднимается" до первой и программируется одновременная установка двух блокировок. Тем самым нарушается необходимое условие возникновения тупика "захват одного и ожидание другого". К сожалению, у этого подхода имеется ряд недостатков:

- требуется поддержка одновременной установки нескольких блокировок;
- в организации тупика могут участвовать несколько потоков и несколько блокировок, так что не всегда достаточно исправить один поток управления и "поднять" одну блокировку;
- если потоки управления, участвующие в тупике, одинаковы, изменение ровно одного из них может оказаться сложной задачей;
- две объединяемые блокировки устанавливаются в разных контекстах, поэтому программирование их одновременной установки также может оказаться сложной задачей.

Нарушение предполагаемого порядка действий — распространенный дефект параллельных программ, суть которого состоит в том, что оператор *A* некоторого потока управления должен выполниться раньше оператора *B* другого потока, но на деле происходит наоборот. На листинге 12 приведен пример подобного дефекта (предполагается, что `fifo` — разделяемая переменная потоков).

```
// Порожденный поток
...
unlock (fifo->mut); // A
...

// Родительский поток
...
fifo = NULL; // B
...
```

Листинг 12. Пример дефекта "нарушение порядка"

Система HFix [11] предназначена для ремонта поведения параллельных программ и, в частности, для устранения дефектов "нарушение порядка". Исходными данными для нее, помимо программы, является доклад об ошибке, в котором указаны операторы *A* и *B*. Иными словами, определять место дефектов не нужно (для параллельных программ это слишком сложная задача), нужно только их исправлять.

Для восстановления предполагаемого порядка действия следует воспользоваться средствами синхронизации. В системе HFix этой цели служит функция `pthread _ join ()`, вызов которой нужно вставить перед оператором *B*. В таком случае родительский поток дождется завершения порожденного потока (и, в частности, выполнения оператора *A*) и только потом обнулит переменную `fifo`.

Отметим, что вставка вызова `pthread _ join ()` — сильнодействующее средство, поскольку до оператора *B* должен выполниться не только оператор *A*, но и все следующие за ним операторы порожденного потока. Если среди них есть действия, способные остановить порожденный поток, может возникнуть тупиковая ситуация. По этой причине HFix в таком случае не будет генерировать вставку `pthread _ join ()`. Он не будет делать этого и тогда, когда оператор *B* находится внутри критического интервала, т. е. HFix проявляет осторожность и старается избежать возникновения тупиков. Если оператор *B* находится в теле цикла, HFix дополнит вставку `pthread _ join ()` манипуляциями с флагом однократного выполнения этого вызова.

Вторая стратегия генерации программных коррекций, реализованная в системе HFix, — это использование имеющихся операторов обработки данных и синхронизации. Например, если в родительском потоке уже присутствует вызов `pthread _ join ()`, его можно "поднять" и поставить перед оператором *B* или, наоборот, "опустить" *B*. Аналогично, если оператор *A* находится в родительском потоке управления, его можно поднять и поставить перед вызовом `pthread _ create ()` для потока, в котором содержится оператор *B*, и т. п.

Система HFix умеет ремонтировать не только нарушения порядка, но и нарушения атомарности. Последние имеют место, если последовательность действий, которая мыслится как неделимая, таковой не является. На листинге 13 приведен фрагмент кода, который в многопотоковой среде приводит к нару-

шению атомарности. Если между двумя приведенными операторами другой поток управления выполнит ту же пару действий, вторая копируемая строка наложится на первую. Предполагается, что `buf` и `buf _ free _ pos` — разделяемые, а `str` и `str _ len` — собственные переменные потоков.

```
memcpy (buf + buf _ free _ pos, str, str _ len);
buf _ free _ pos += str _ len;
```

Листинг 13. Фрагмент С-кода с потенциальным нарушением атомарности

Отправной точкой для устранения нарушений атомарности системой HFix является доклад об ошибке. В нем должны быть указаны два оператора из одного потока управления, которые принято обозначать *P* и *C*, а также оператор *R* из другого потока, такой, что выполнение *R* между *P* и *C* является некорректным действием. Нарушения атомарности система HFix устраняет только путем перемещения имеющихся операторов. В отличие от нее, система AlphaFixer [12] способна, кроме перемещений, генерировать новые блокировки, оформляя критические интервалы. Да и перемещения AlphaFixer выполняет аккуратнее, избегая самоблокировки потоков (когда повторная блокировка оказывается поднятой выше освобождения первой). Согласно приведенным в работе [12] результатам система AlphaFixer смогла исправить 15 из 15 нарушений атомарности, а HFix — только два; еще в четырех случаях исправления HFix привели к самоблокировке.

Ремонт поведения параллельных программ имеет много особенностей, одна из которых — архитектурная природа ошибок. Если при ремонте последовательных программ можно считать, что программа "почти правильная" и достаточно нескольких точечных исправлений, то в параллельном случае может потребоваться глобальная перестройка: упорядочение захватываемых ресурсов, "приватизация" данных потоками управления и т. д. Это ближе к синтезу, чем к ремонту программ. Параллельные программы должны быть корректными по построению, проверка набором тестов неубедительна.

Ремонт поведения сценариев сборки

Сценарии сборки — важное звено в технологической цепочке программирования, особенно при ориентации на непрерывную интеграцию. Как и другие элементы программного обеспечения, сценарии сборки требуют сопровождения, обновления версий и исправления ошибок. В то же время языки сценариев отличаются от привычных для большинства специалистов языков программирования, так что их написание и сопровождение требуют специальных знаний. Оперативное ручное исправление ошибок в таких условиях не всегда возможно, что делает актуальной задачу автоматического ремонта.

Система HireBuild [13] стала, вероятно, первой системой автоматического ремонта поведения сценариев сборки. Она ориентирована на Gradle — инструмент для сборки Java-проектов. Идейная основа HireBuild близка к универсальным системам автоматического ремонта поведения из категории "порождай и проверяй", но есть и особенности, следующие из специфики предметной области.

Протокол процесса сборки весьма подробный. В частности, он содержит детальную информацию о месте неисправности и ее характере. На листинге 14 приведен пример фрагмента протокола из работы [13].

```
   Could not resolve all dependencies for
configuration
   ':quasar-galaxy:compile'.
     > A conflict was found between the following
modules:
   - org.slf4j:slf4j-api:1.7.10
   - org.slf4j:slf4j-api:1.7.7.
```

Листинг 14. Пример фрагмента протокола процесса сборки с сообщением об ошибке

В системе HireBuild протокол используется для локализации дефектов. Кроме того, из протокола (а не из исходного текста сценария) извлекаются имена объектов, причастных к неисправности, которые, возможно, пригодятся для ремонта. На листинге 15 приведен пример программной коррекции, сгенерированной системой HireBuild (имеется в виду оператор `exclude group:`).

```
compile ("co.paralleluniverse:galaxy:1.4") {
   ...
   exclude group: org.slf4j, module: *
}
```

Листинг 15. Пример программной коррекции, сгенерированной системой HireBuild

В сценариях сборки много универсальных элементов, не зависящих от конкретного проекта. Это наводит на мысль об использовании истории исправлений других проектов. Данная идея положена в основу системы HireBuild. Логика ее работы (если опустить технические детали) состоит в следующем:

- из протокола работы ремонтируемого сценария сборки извлекается информация об ошибке;
- из этой информации удаляются имена, специфичные для проекта;
- в этом же и других доступных проектах анализируются протоколы сборки и отыскиваются похожие сообщения об ошибке;
- из истории извлекаются исправления, ассоциированные с найденными сообщениями;
- исправления превращаются в шаблоны путем отбрасывания специфики проекта;
- полученные таким образом шаблоны конкретизируются с учетом контекста ремонтируемого проекта, получаются потенциальные заплаты;
- определяются места возможного наложения сгенерированных заплат;
- заплаты проверяются путем запуска сценария сборки и анализа кода его завершения.

Для принятия программной коррекции необходим не только нормальный код завершения, но и совпадение списка откомпилированных файлов с протоколом последней успешной сборки. Это стандартная защита от исправлений, уничтожающих полезную функциональность.

Согласно работе [13] система HireBuild была испытана на наборе из 135 обучающих и 24 тестовых дефектов, из которых исправить удалось 11. Основная причина неудач (6 из 13) — отсутствие подходящего шаблона. Возможно, сказалась ограниченность тренировочных данных. Другие причины — неспособность внести многострочные исправления и/или неспособность учесть специфику проекта. Конечно, для далеко идущих выводов (во всяком случае, положительных) представленных экспериментальных данных недостаточно.

К сожалению, выразительная сила языков сценариев сборки достаточна для того, чтобы сделать задачу ремонта их поведения нетривиальной. Например, диагностическое сообщение об отсутствии какого-либо объекта может стать следствием как ошибки в начальной зачистке проекта, так и выбора не той версии. То есть локализация дефекта и для сценариев сборки, несмотря на детальность протоколов, остается трудной проблемой [14], в общем случае требующей для своего решения обратной отладки. Нельзя недооценивать и специфику проектов. Отметим также, что многократные проверочные запуски сценариев сборки — действие небезопасное и длительное. Настораживает и желание авторов работы [13] применить генетический подход и функции годности для реализации комбинирования заплат и внесения многострочных исправлений. На наш взгляд, потолок возможностей системы HireBuild — генерация типовых, коротких программных коррекций, что, разумеется, само по себе немало.

Ремонт состояния

В данном разделе состояние программной системы понимается в широком смысле — это и регистры компьютера, и память, и исходные данные, и окружение. Ремонт состояния требуется, если программа завершилась аварийно, если нарушены какие-либо контракты или если с точки зрения системы мониторинга (при наличии таковой) поведение программы является подозрительным.

Известные средства обеспечения отказоустойчивости можно трактовать как средства ремонта состояния. К числу этих средств относятся перечисленные далее.

- Внесение избыточности в аппаратно-программную конфигурацию. Это и резервирование разных степеней готовности, и наличие нескольких альтернативных реализаций одного сервиса, и параллельная обработка одного запроса несколькими исполнителями с определением правильного результата голосованием, и т. п.
- Контрольные точки и откат. Это стандарт для систем обработки транзакций, но может применяться и в случае нарушения целостности графа управления вследствие вредоносной активности. Реализация этого средства сопряжена с известными техническими проблемами, поскольку не все действия (например, ввод-вывод) можно откатить.

Среди других средств ремонта поведения можно выделить следующие.

◇ Перезапуск. Он может иметь разный масштаб: от перезапуска распределенной системы, перезагрузки компьютера до рестарта отдельных компонентов и их последующей синхронизации в компонентной архитектуре.

◇ Переконфигурирование. Обычно увеличивают количественные параметры, такие как объем динамической памяти, число обслуживающих процессов и т. п.

Возможен и подход, преследующий цель ликвидации отклонений, чтобы не допустить проявления неисправностей. Такой подход реализован в системе ClearView [15].

Система ClearView предназначена для ремонта состояния программ в бинарном представлении (выполняющихся на аппаратно-программной платформе ×86/Windows), без обращения к исходным текстам и отладочной информации. В ClearView используется среда контролируемого выполнения, в которой программы по сути дела интерпретируются. Базовые блоки по мере необходимости оснащаются и подгружаются в кэш кода, где и происходит выполнение. Если какой-либо базовый блок стал ненужным, он удаляется из кэша кода. Эти механизмы позволяют реализовать наложение и удаление программных коррекций.

Работу системы ClearView можно подразделить на перечисленные далее пять этапов.

- Обучение. Несколько успешных выполнений программы трассируются, и на основе собранных данных выявляются инварианты — унарные и бинарные. В роли переменных в них выступают регистры или ячейки памяти. Унарный инвариант может задавать (небольшое) множество допустимых значений переменной или нижнюю границу значений, бинарный — неравенство между значениями. Основное назначение инвариантов с неравенствами — обнаружить и устранить выход за границы блока памяти. Место в программе, с которым ассоциирован инвариант, это место после инструкции, устанавливающей значение переменной (для бинарных инвариантов имеется в виду инструкция, выполняющаяся позже). На этапе обучения строятся также внутрипроцедурные графы передачи управления.
- Мониторинг. Назначение этого этапа — контролировать корректность поведения программы. Вообще говоря, критерии некорректности могут быть разными, но в описываемой реализации системы ClearView отслеживаются только подозрительные передачи управления и выход за границы блока динамической памяти (с использованием "канареек"). При выявлении некорректного поведения программа терминируется и данные о месте и причине неудачи сохраняются для последующего ремонта. Такой подход лучше всего подходит для сервис-ориентированных архитектур, когда обработка отдельных запросов может закончиться неудачей, но приложение в целом сохраняет работоспособность.

• Идентификация коррелирующих инвариантов. Если монитор регистрирует неисправность, то рассматриваются инварианты, предшествующие неисправности. Здесь требуется выдержать баланс между полнотой рассмотрения и размерам накладных расходов. Для отобранных инвариантов порождаются проверочные заплаты, которые при последующих выполнениях позволят определить, соблюдался инвариант или нет. Коррелирующими с неисправностью считаются инварианты, которые соблюдались при успешных выполнениях и нарушались при неудаче.

• Генерация исправляющих заплат. Для коррелирующих инвариантов порождаются заплаты, исправляющие нарушение. Для одного инварианта таких заплат может быть несколько. Например, для унарного инварианта с множеством возможных значений порождаются команды, устанавливающие эти значения, но если переменная — это указатель на функцию, то соответствующий вызов можно пропустить или сгенерировать возврат из текущей функции. Для унарного инварианта с неравенством устанавливается нижняя граница.

• Оценка исправляющих заплат. Цель этого этапа — отобрать программные коррекции, восстанавливающие нормальную работу.

Система ClearView может ремонтировать сообщество аппаратно-программных комплексов, выполняющих одно и то же приложение, т. е. централизует информацию о неудачах и рассылает программные коррекции. Отметим, что ClearView справляется с ситуацией, когда несколько комплексов, работающих параллельно, терпят различные неудачи.

В работе [16] приведены результаты испытаний системы ClearView. Ей удалось устранить 9 из 10 известных уязвимостей в Firefox и успешно противостоять попыткам их использования. По крайней мере 4 из 9 заплат исправляли не только поведение, но и дефекты в программе, хотя, строго говоря, такая цель не ставилась.

Конечно, для объективной оценки качества какой-либо системы нужно провести много больше экспериментов, но из общих соображений перспективы систем типа ClearView представляются сомнительными. Прежде всего, в их основе лежит тяжеловесная и шаткая конструкция. Бинарный код интерпретируется, а на этом уровне трудно выявить содержательные инварианты. Весьма вероятно, что при отражении атак сработала защита от нарушения целостности графа управления, что полезно, но недостаточно для ремонта состояния при других неисправностях. На наш взгляд, лучшим решением остаются традиционные механизмы обеспечения отказоустойчивости.

Выводы

Рассмотрен широкий спектр идей и подходов к автоматическому ремонту программ. Универсальные системы ремонта поведения не имеют теоретического обоснования, а практические результаты их применения неудовлетворительны.

Более перспективными представляются специализированные системы, ориентированные, например, на исправление дефектов динамического управления памятью. Подобные системы — удачная форма накопления и использования программистских знаний; возможно, будущие системы ремонта поведения будут строиться путем интеграции большого числа специализированных систем.

У систем ремонта состояния имеется проверенная база — наработки в области отказоустойчивости. На наш взгляд, адаптация и развитие этих наработок применительно к ремонту программ — наиболее естественный и перспективный подход.

Список литературы

1. **Nielebock S.** Towards API-Specific Automatic Program Repair // Proc. of ASE'2017. Urbana-Champaign, IL, USA, Doctoral Symposium, 2017. P. 1010—1013.
2. **Dhar A., Purandare R., Dhawan M., Rangaswamy S.** CLOTHO: Saving Programs from Malformed Strings and Incorrect String-Handling // Proc. of ESEC/FSE'15. August 30 — September 4, 2015. Bergamo, Italy, 2015. P. 555—566.
3. **Durieux T., Cornu B., Seinturier L., Monperrus M.** Dynamic Patch Generation for Null Pointer Exceptions using Metaprogramming // SANER'2017. Klagenfurt, Austria, Main Research. IEEE, 2017. P. 349—358.
4. **Tian Y., Ray B.** Automatically Diagnosing and Repairing Error Handling Bugs in C // Proc. of 2017 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, (ESEC/FSE'17). September 4—8, 2017. Paderborn, Germany, 2017. P. 752—762.
5. **Xi Cheng, Min Zhou, Xiaoyu Song, Ming Gu, Jiaguang Sun.** IntPTI: Automatic Integer Error Repair with Proper-Type Inference // Proc. of ASE'2017. Urbana-Champaign, IL, USA, 2017. P. 996—1001.
6. **Qing Gao, Yingfei Xiong, Yaqing Mi, Lu Zhang, Weikun Yang, Zhaoping Zhou, Bing Xie, Hong Mei.** Safe Memory-Leak Fixing for C Programs // Proc. of 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, 2015. P. 459—470.
7. **van Tonder R., Le Goues C.** Static Automated Program Repair for Heap Properties // Proc. of ICSE'18: 40th International Conference on Software Engineering. May 27 — June 3, 2018. Gothenburg, Sweden, 2018. P. 151—162.
8. **Junhee Lee, Seongjoon Hong, Hakjoo Oh.** MemFix: Static Analysis-Based Repair of Memory Deallocation Errors for C // Proc. of the 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'18). November 4—9, 2018. Lake Buena Vista, FL, USA, 2018. P. 95—106.
9. **Lin Y., Kulkarni S. S.** Automatic Repair for Multi-threaded Programs with Deadlock / Livelock using Maximum Satisfiability // Proc. of ISSTA'14. July 21—25, 2014. San Jose, CA, USA, 2014. P. 237—247.
10. **Yan Cai, Lingwei Cao.** Fixing Deadlocks via Lock Pre-Acquisitions // Proc. of IEEE/ACM 38th IEEE International Conference on Software Engineering, ICSE'16. May 14—22, 2016. Austin, TX, USA, 2016. P. 1109—1120.
11. **Haopeng Liu, Yuxi Chen, Shan Lu.** Understanding and Generating High Quality Patches for Concurrency Bugs // Proc. of FSE'16. November 13—18, 2016. Seattle, WA, USA, 2016. P. 715—726.
12. **Yan Cai, Lingwei Cao, Jing Zhao.** Adaptively Generating High Quality Fixes for Atomicity Violations // Proc. of 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'17). September 4—8, 2017. Paderborn, Germany, 2017. P. 303—314.
13. **Hassan F., Wang X.** HireBuild: An Automatic Approach to History-Driven Repair of Build Scripts // Proc. of 40th International Conference on Software Engineering (ICSE'18). May 27 — June 3, 2018. Gothenburg, Sweden, 2018. 12 p.
14. **Al-Kofahi J., Nguyen H. V., Nguyen T. N.** Fault Localization for Build Code Errors in Makefiles // Proc. of ICSE Companion'14. May 31 — June 7, 2014. Hyderabad, India, 2014. P. 600—601.
15. **Perkins J. H., Kim S., Larsen S.** et al. Automatically Patching Errors in Deployed Software // Proc. of SOSP'09. October 11—14, 2009. Big Sky, Montana, USA, 2009. P. 87—102.
16. **Qi Z., Long F., Achour S., Rinard M.** An Analysis of Patch Plausibility and Correctness for Generate-and-Validate Patch Generation Systems // Proc. of ISSTA'15. July 13—17, 2015. Baltimore, MD, USA, 2915. P. 24—36.

Automated Program Repair: Comparative Analysis of Approaches

V. A. Galatenko, galat@niisi.ras.ru, **K. A. Kostyukhin,** kost@niisi.ras.ru, Federal State Institution "Scientific Research Institute for System Analysis of the Russian Academy of Sciences", Moscow, 117218, Russian Federation

Corresponding author:

Kostyukhin Konstantin A., Senior Researcher, Federal State Institution "Scientific Research Institute for System Analysis of the Russian Academy of Sciences", Moscow, 117218, Russian Federation
E-mail: kost@niisi.ras.ru

Received on July 09, 2019
Accepted on August 06, 2019

The article is the second part of the work devoted to the automated program repair. It provides a comparative analysis of ideas and approaches to the automated program repair. The authors tried to reflect the main existing approaches in the article, to consider the advantages and disadvantages of each of them.

The article describes a wide range of ideas and approaches to the automated program repair. Universal systems of behavior repair have no theoretical basis, and practical results of their application are unsatisfactory.

Specialized systems focused, for example, on the correction of dynamic memory management defects are more promising. Such systems are a good form of accumulation and usage of programming knowledge. It is possible that future systems of behavior repair will be built by integrating a large number of specialized systems.

The state repair systems have a proven base — researches in the field of fault tolerance. In our opinion, the adaptation and development of these researches in relation to the repair of programs is the most natural and promising approach.

Keywords: *debugging, automated program repair, program recovery, program re-execution, program self-treatment, data recovery*

For citation:

Galatenko V. A., Kostyukhin K. A. Automated Program Repair: Comparative Analysis of Approaches, *Programmnaya Ingeneria*, 2019, vol. 10, no. 11—12, pp. 419—429.

DOI: 10.17587/prin.10.419-429

References

1. **Nielebock S.** Towards API-Specific Automatic Program Repair, *Proc. of ASE'2017*, Urbana-Champaign, IL, USA, Doctoral Symposium, pp. 1010—1013.

2. **Dhar A., Purandare R., Dhawan M., Rangaswamy S.** CLOTHO: Saving Programs from Malformed Strings and Incorrect String-Handling, *Proc. of ESEC/FSE'15*, August 30 — September 4, 2015, Bergamo, Italy, 2015, pp. 555—566.

3. **Durieux T., Cornu B., Seinturier L., Monperrus M.** Dynamic Patch Generation for Null Pointer Exceptions using Metaprogramming. *SANER'2017*, Klagenfurt, Austria, Main Research, IEEE, 2017, pp. 349—358.

4. **Tian Y., Ray B.** Automatically Diagnosing and Repairing Error Handling Bugs in C, *Proc. of 2017 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, (*ESEC/FSE'17)*, Paderborn, Germany, September 2017, pp. 752—762.

5. **Xi Cheng, Min Zhou, Xiaoyu Song, Ming Gu, Jiaguang Sun.** IntPTI: Automatic Integer Error Repair with Proper-Type Inference, *Proc. of ASE'2017*, Urbana-Champaign, IL, USA, 2017, pp. 996—1001.

6. **Qing Gao, Yingfei Xiong, Yaqing Mi, Lu Zhang, Weikun Yang, Zhaoping Zhou, Bing Xie, Hong Mei.** Safe Memory-Leak Fixing for C Programs, *Proc. of 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015, pp. 459—470.

7. **van Tonder R., Le Goues C.** Static Automated Program Repair for Heap Properties, *Proc. of ICSE'18: 40th International Conference on Software Engineering*, May 27 — June 3, 2018, Gothenburg, Sweden, 2018, pp. 151—162.

8. **Junhee Lee, Seongjoon Hong, Hakjoo Oh.** MemFix: Static Analysis-Based Repair of Memory Deallocation Errors for C, *Proc. of the 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'18)*, November 4—9, 2018, Lake Buena Vista, FL, USA, 2018, pp. 95—106.

9. **Yiyan Lin, Sandeep S. Kulkarni.** Automatic Repair for Multi-threaded Programs with Deadlock/Livelock using Maximum Satisfiability, *Proc. of ISSTA'14*, July 21—25, 2014, San Jose, CA, USA, 2014, pp. 237—247.

10. **Yan Cai, Lingwei Cao.** Fixing Deadlocks via Lock Pre-Acquisitions, *Proc. of IEEE/ACM 38th IEEE International Conference on Software Engineering, ICSE'16*, May 14—22, 2016, Austin, TX, USA, 2016, pp. 1109—1120.

11. **Haopeng Liu, Yuxi Chen, Shan Lu.** Understanding and Generating High Quality Patches for Concurrency Bugs, *Proc. of FSE'16*, November 13—18, 2016, Seattle, WA, USA, 2016, pp. 715—726.

12. **Yan Cai, Lingwei Cao, Jing Zhao.** Adaptively Generating High Quality Fixes for Atomicity Violations: *Proc. of 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'17)*, September 4—8, 2017, Paderborn, Germany, 2017, pp. 303—314.

13. **Hassan F., Wang X.** HireBuild: An Automatic Approach to History-Driven Repair of Build Scripts, *Proc. of 40th International Conference on Software Engineering (ICSE'18),* May 27 — June 3, 2018, Gothenburg, Sweden, 2018, 12 p.

14. **Al-Kofahi J., Nguyen H. V., Nguyen T. N.** Fault Localization for Build Code Errors in Makefiles, *Proc. of ICSE Companion'14*, May 31 — June 7, 2014, Hyderabad, India, 2014, pp. 600—601.

15. **Perkins J. H., Kim S., Larsen S., Amarasinghe S., Bachrach J., Carbin M., Pacheco C., Sherwood F., Sidiroglou S., Sullivan G., Wong W.-F., Zibin Y., Ernst M. D., Rinard M.** Automatically Patching Errors in Deployed Software, *Proc. of SOSP'09*, October 11—14, 2009, Big Sky, Montana, USA, 2009, pp. 87—102.

16. **Qi Z., Long F., Achour S., Rinard M.** An Analysis of Patch Plausibility and Correctness for Generate-and-Validate Patch Generation Systems, *Proc. of ISSTA'15,* July 13—17, 2015, Baltimore, MD, USA, 2015, pp. 24—36.

УДК 004.942; 330.46 DOI: 10.17587/prin.10.430-439

Е. В. Орлова, д-р техн. наук, проф., ekorl@mail.ru, Уфимский государственный авиационный технический университет

Инженерия системного синтеза эффективности инновационных проектов

Рассмотрена проблема комплексной оценки эффективности инновационной деятельности предприятия. Разработан подход к системному синтезу эффективности инновационных проектов, обосновывающий с позиции социальной, экономической и экологической составляющих социальный характер, конкурентоспособность и экологичность проектов.

Ключевые слова: *инженерия комплексной эффективности проектов, инновационные проекты, системный синтез, имитационное моделирование*

Введение

Категория "эффективность" широко используется в разных областях науки и практической деятельности. Проблема оценки эффективности различных видов деятельности предприятий наряду с задачами управления их функционированием и развитием была и остается одной из ключевых тем экономики и управления как в России, так и за рубежом. Выдвижению этой проблемы на передний план современных исследований способствуют переход на инновационный путь развития экономики и необходимость повышения темпов внесения изменений в действующий механизм управления.

Феномен эффективности изучается представителями разных экономических школ и направлений, исследуется учеными в области теории управления, системного анализа и моделирования. Однако до сих пор не выработано подхода, позволяющего комплексно оценивать, прогнозировать и управлять эффективностью деятельности предприятия в условиях инновационной экономики как сложной динамической многоуровневой организационной системой, с учетом множества взаимосвязанных, часто стохастически, параметров и частных показателей эффективности подсистем.

Развитие российской экономики в ближайшие годы связано с перспективами цифровизации, в которой ценным являются нестандартные, индивидуализированные продукты, креативные разработки. Важнейшие национальные цели развития России, связанного с обеспечением ускоренного внедрения цифровых технологий в экономике и социальной сфере, определены в Указе Президента РФ[1]. Государственная политика в сфере применения информационных и коммуникационных технологий, направленных на развитие цифрового общества, формирования национальной цифровой экономики, сформулирована в стратегии развития информационного общества[2].

В паспорте национальной программы "Цифровая экономика РФ"[3] выделены цели, задачи, направления и сроки реализации основных мер политики по созданию необходимых условий для развития в России цифровой экономики, в которой данные в цифровом виде являются ключевым фактором производства во всех сферах социально-экономической деятельности. Это является необходимым условием повышения конкурентоспособности страны, качества жизни граждан, обеспечения экономического роста.

Как следствие, работы, направленные на обеспечение и разработку средств автоматизации и программного обеспечения, автоматизирующего процессы оценки эффективности деятельности предприятий, выполнение отдельных проектов, становятся все более востребованными. Они нацелены на создание технологий, методов и средств стимулирования развития предприятий как базового производительного элемента экономики.

1. Существующие подходы к оценке эффективности предприятия как социально-экономической системы

Эффективность является одной из характеристик качества системы, ее результативностью, и представляет собой способность производить определенный эффект. Результативность системы определяется соотношением затрат и результатов функционирования системы. В зависимости от того, какие затраты и какие результаты принимаются во внимание, раз-

[1] Указ Президента РФ от 07.05.2018 № 204 (ред. от 19.07.2018) "О национальных целях и стратегических задачах развития Российской Федерации на период до 2024 года".

[2] Указ Президента РФ от 09.05.2017 № 203 "О Стратегии развития информационного общества в Российской Федерации на 2017—2030 годы".

[3] Паспорт национальной программы "Цифровая экономика Российской Федерации", утв. президиумом Совета при Президенте Российской Федерации по стратегическому развитию и национальным проектам, протокол от 24 декабря 2018 г. № 16.

личают виды эффектов и эффективности — организационно-производственную, коммерческую, социальную, экологическую, бюджетную.

Экономическая эффективность отражает способность системы производить в процессе своего функционирования экономический эффект. Чем больше экономический эффект (результат) и меньше для получения этого результата затраты ресурсов, тем выше экономическая эффективность. В ряде случаев оценка экономической эффективности ограничивается соотношением между полученными результатами и затратами на его получение. Такое определение сформировалось исторически в работах неоклассической экономической школы (Дж. М. Кейнс, А. Пигу, Х. Лейбенстайн; И. Ансофф, П. Друкер, Г. Минцберг) [1—4].

Принципы предельной экономической эффективности, сформулированные В. Парето, относятся к центральным понятиям в современной экономической науке. Согласно этим принципам экономически эффективным можно считать такой уровень организации экономики и производства, при котором государство и общество извлекают максимум пользы из существующих и затрачиваемых на решение конкретных задач ресурсов. При этом следует заметить, что невозможно улучшить значения одних характеристик системы (предприятия), не ухудшая другие. На этом определении строится теория благосостояния, исследующая процессы справедливого распределения ресурсов в экономике и ее эффективности.

С позиций системного подхода предприятие обычно представляется в виде черного ящика, при этом внутренняя структура системы не раскрывается, а исследованию подвергают только вход в модель (ресурсы) и выход из нее (эффекты) [5—11].

Такое понимание (отождествление эффективности функционирования системы с ресурсоемкостью результата) имеет ряд очевидных недостатков [11]. Так, ресурсоемкость не отражает своевременность, необходимость и достаточность получения полезного эффекта. Одно и то же частное от деления эффекта на затраты может достигаться при различных сочетаниях числителя и знаменателя, тогда как оценка эффективности требует, в том числе, учета их абсолютных значений. Кроме того, не ясны источники возникновения эффективности, следовательно, не понятно, как ею управлять. В условиях рынка прибыль в большей степени детерминируется рыночной ситуацией, чем затратами. Наконец, не специфицируется, какие эффекты и затраты следует включать в общую оценку качества системы, имея в виду мультипликативные (синергетические) эффекты, возникающие при взаимодействии результатов деятельности различных подсистем. Поэтому при оценке эффективности необходимо раскрывать сложную структуру причинно-следственных связей между результатом и обусловившими его факторами в виде затрат, ресурсного потенциала, способности к развитию и т. д.

В работах по системному анализу и управлению под эффективностью понимается комплексное операционное свойство, характеризующее качество процесса функционирования системы и степень ее приспособленности к достижению цели [12—22]. В понятие "цель" вкладываются различные оттенки — "от идеальных устремлений" (цель как выражение активности агента) до конкретных целей — конечных результатов, достижимых в пределах некоторого временного интервала. Целевой подход более точно позволяет отразить различные режимы функционирования предприятия, например, нормальный (при простом или расширенном воспроизводстве), кризисный (выживание в экстремальных ситуациях), переходный (восстановление нормального функционирования после кризиса). Отметим, что вопросы, связанные с целеполаганием, составляют одно из важнейших направлений в системном анализе. Существуют различные методики структуризации целей, отличающиеся способами представления системы, учетом стадий ее жизненного цикла, формой выражения целей.

В теории управления эффективность функционирования управляемой системы как степень соответствия ее результатов и целей зависит от состояния системы и от управляющих воздействий. В случае если известна зависимость состояния управляемой системы от управления, можно получить зависимость эффективности функционирования системы только от управляющих воздействий. Этот критерий является критерием эффективности управления системой. В этом случае задача управления может быть сформулирована в виде прямой задачи, т. е. задачи синтеза оптимальных управляющих воздействий, имеющих максимальную эффективность, или в виде обратной задачи — найти множество управляющих воздействий, переводящих управляемую систему в заданное состояние. Если эффективность измерима, то целью управления является оптимизация эффективности, т. е. ее максимизация при заданных ограничениях в определенных условиях пространства и времени.

Согласно иному подходу к оценке эффективности, предприятие рассматривается не изолированно от внешней среды, а в комплексе с дополняющей его надсистемой и ее элементами — институциональной средой, конкурентами, поставщиками ресурсов, потребителями. В рамках такого подхода эффективной можно признать работу предприятия не в том случае, когда оно затрачивает меньше ресурсов, чем его конкуренты, а когда весь контур "предприятие–надсистема" устойчиво и бесперебойно функционирует в течение значимого периода времени. Подобная ситуация возникает, когда спрос на продукцию предприятия, с одной стороны, и спрос на привлекаемые ресурсы, с другой стороны, изменяются взаимосвязано. Иными словами, эффективным является предприятие, гармонизированное с внешним миром по спросу и предложению в отношении всех видов осуществляемых им функций [21].

Проведенный анализ существующих подходов в области идентификации и оценивания эффективности деятельности предприятий показал, что в настоящее время не существует единого методо-

логического подхода к ее оценке. Эффективность предприятия должна оцениваться комплексно, с учетом всех видов бизнес-деятельности (производственной, финансовой, инвестиционной, инновационной), а также с учетом воздействия этих видов деятельности на окружающую среду — природу, общество.

2. Предлагаемая концепция к оценке и управлению эффективностью деятельности предприятия

У предприятия как у системы можно выделить ряд существенных свойств. Эти свойства, в свою очередь, можно условно разбить на три группы: общесистемные свойства (целостность, устойчивость, управляемость, наблюдаемость, детерминированность, открытость, динамичность); структурные свойства (связность, организация, сложность, масштабность, централизованность); поведенческие (функциональные, операционные) свойства (результативность, ресурсоемкость, оперативность, производительность, точность, быстродействие, экономичность). Первые две группы свойств, обусловливающих ее пригодность для использования по назначению, определяют качество системы. Третья группа свойств определяет операционную эффективность, т. е. эффективность процесса функционирования системы, и характеризует его приспособленность к достижению цели системы. Тогда эффективность предприятия (проекта, программы) можно представить в виде комплексной (системной) оценки качества его структуры и качества функционирования. Она определяется рядом разнородных составляющих и показателей как количественного, так и качественного характера. Комплексное определение эффективности сводится к процедуре выявления существенных составляющих, определению условий возникновения системного эффекта и обоснованию факторов, определяющих этот эффект.

Представим хозяйственный процесс социально-экономической системы (предприятия) в виде двух типов операций (процессов). Первый тип операций определяет основные процессы функционирования предприятия — процессы производства, распределения, обмена и потребления, требующие для их реализации совокупность ресурсов. Второй тип операций — это процессы управления — процессы планирования, организации, учета, контроля и координации.

Эффективность хозяйственного процесса или процесса функционирования предприятия может быть идентифицирована с помощью двух групп характеристик [23—25]:

- аллокативной эффективности, характеризующей продуктивность распределения ресурсов, а также экономичность их расходования;
- адаптивной эффективности, оценивающей результативность выполнения поставленных целей и задач и степень приспособленности различных подсистем к изменениям внутренней и внешней среды в процессе достижения поставленных целей.

По специализации бизнес-процессов аллокативная эффективность объединяет эффективности процессов производства, распределения, обмена и потребления; адаптивная эффективность — эффективности процессов планирования, организации, учета, контроля и координации. По видам и сферам распространения результатов функционирования и управления предприятием выделим коммерческий (экономический), бюджетный, социальный, экологический виды эффектов и эффективностей.

Важным обстоятельством является то, что операционная эффективность предприятия в целом определяется совокупностью частных показателей, отражающих аллокативную и адаптивную эффективности, а также их агрегацией по определенным правилам. Поэтому исследования, направленные на оценку взаимодействия и выявление связи между эффективностями различного типа, а также их вклада в общую эффективность системы, представляются весьма перспективными.

Предлагается системная модель управления эффективностью предприятия, которая задается оператором $F(A, E)$, определяющим принцип оптимальности на основе соизмерения оценок аллокативной (A) и адаптивной (E) эффективностей.

Аллокативная эффективность определяет, насколько продуктивно распределяются те или иные ресурсы, а также насколько экономно они расходуются. Эффективное распределение ресурсов отражает оптимальный уровень комбинации продукции, произведенной с помощью наиболее эффективной комбинации ресурсов. При этом под оптимальной комбинацией ресурсов подразумевается такой выпуск продукции, который приобретается потребителями на конкурентном рынке по цене, основанной на реальных издержках производства, а под эффективной комбинацией ресурсов — производство этой продукции с минимальными издержками. Экономика функционирует эффективно, если она может обеспечить потребителей именно теми товарами и услугами, которые им необходимы при существующем уровне технологии и количестве ресурсов.

К количественным характеристикам, описывающим экономическую результативность функционирования предприятия и отражающим распределение капитала, рабочей силы, имущества и оборотных активов, относится совокупность следующих показателей: ликвидность и платежеспособность, имущественное положение, финансовая устойчивость, рентабельность, эффективность использования ресурсов по видам, безубыточность и операционный рычаг.

Адаптивная эффективность характеризует результативность выполнения поставленных целей и задач, а также степень приспособленности различных подсистем к изменениям внутренней и внешней среды в процессе достижения поставленных целей. В качестве характеристики объективной возможности не добиться заданной цели, не получить в заданном объеме желаемых результатов экономической деятельности предприятия будем использовать уровень экономического риска. Экономический риск отражает меру отклонения от цели экономической

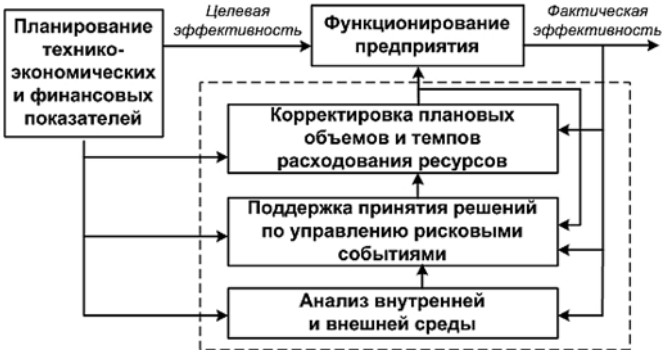

Рис. 1. Иерархическая структура системы управления эффективностью предприятия

деятельности предприятия и масштаб потенциального ущерба, обусловленного этим отклонением. Под риском понимается возможная актуализация в будущем неопределенных и непредсказуемых результатов принимаемых решений в деятельности предприятия с точки зрения достижения поставленных целей. Важнейшими задачами при управлении рисками на предприятии являются: идентификация объективных и субъективных факторов, влияющих на конкретный вид риска; анализ выявленных факторов и оценка степени их влияния на результаты экономической деятельности; установление допустимого уровня риска; разработка мероприятий по снижению риска. Методики управления конкретными видами риска подробно рассмотрены, например, в работах [26, 27].

Структура системы управления эффективностью предприятия, включающая блок управления рисками, представлена на рис. 1. Система управления ресурсами решает задачи регулирования, организации, стимулирования, а на систему управления рисками возложены задачи анализа, прогнозирования и контроля реализации поставленных целей предприятия.

Технология управления эффективностью предприятия в целом должна учитывать активность экономических агентов, которая может проявляться в искажении информации, в выборе решений, не совпадающих с планом, недобросовестном поведении. Для повышения эффективности управления в системе с активными элементами необходимо моделировать их поведение, т. е. иметь инструментарий прогнозирования их реакции на управленческие воздействия [28—30].

3. Подход к комплексной оценке эффективности инновационной деятельности предприятия

Традиционно инновационные проекты предприятия описываются с позиции их коммерческой (экономической) эффективности с использованием таких показателей, как чистый дисконтированный доход, срок окупаемости и индекс рентабельности, либо характеризуются социальной значимостью или экологической эффективностью. Однако не существует сколько-нибудь формализованного подхода для одновременной оценки разных видов воздействий проекта на среду и разных видов его эффективности. Для ряда проектов, значительно воздействующих на окружающую среду и социум, такая оценка представляется наиболее адекватной и соответствующей природе самого проекта. К примеру, существуют проекты, которые имеют значительное влияние на экологическое состояние окружающей среды (загрязнение водных, воздушных и почвенных ресурсов), а также проекты, имеющие только социальную направленность и обладающие лишь социальной эффективностью (улучшают качество жизни). Другие проекты имеют ярко выраженные экономический, социальный и экологический эффекты, например, проекты по созданию новых видов технологий по производству лекарственных препаратов. Поэтому необходим комплексный многомерный критерий оценки эффективности инновационных проектов, который обеспечивал бы обоснование качества проекта и его количественную оценку с учетом экологической, социальной и экономической составляющих его эффективности.

Отметим, что далее оценивается лишь операционная составляющая системной эффективности, отражающая процесс реализации проекта. Структурная составляющая системной эффективности не рассматривается.

Формализованное представление такого критерия оценки системной эффективности инновационного проекта имеет вид $SE = f(EE, CE, EcE)$, где SE — системная эффективность; EE — экономическая эффективность; CE — социальная эффективность; EcE — экологическая эффективность.

Концептуально подход к поэтапному оцениванию системной эффективности проекта, отражающего последовательную оценку социальной, экономической и экологической эффективности, а также инструментальные средства и результаты анализа отдельных видов эффективности в единый критерий, показан на рис. 2. При этом тип инновации может быть различным — новый продукт или новая технология производства известного продукта.

Оценка системной эффективности инновационного проекта состоит из трех этапов. На этапе 1 осуществляется отбор однородных типов продуктов, представленных на рынке, для производства которых разрабатывается инновационная технология, с позиции социальной эффективности. Для того чтобы проранжировать продукты по степени их полезности с точки зрения множества критериев выбора, используется метод анализа иерархий. Метод анализа иерархий включает процедуры синтеза множественных суждений, выявления приоритетности критериев и нахождения альтернативных решений. Результатом такого анализа является вектор приоритетов различных продуктов, упорядоченных в порядке убывания социальной эффективности от применения этого продукта. Примером такого продукта может явиться производство лекарственных препаратов, нацеленных на лечение конкретного вида заболевания. Изучению подлежат различные лекарственные препараты, имеющие ту или иную эффективность применения с учетом критериев их полезности и

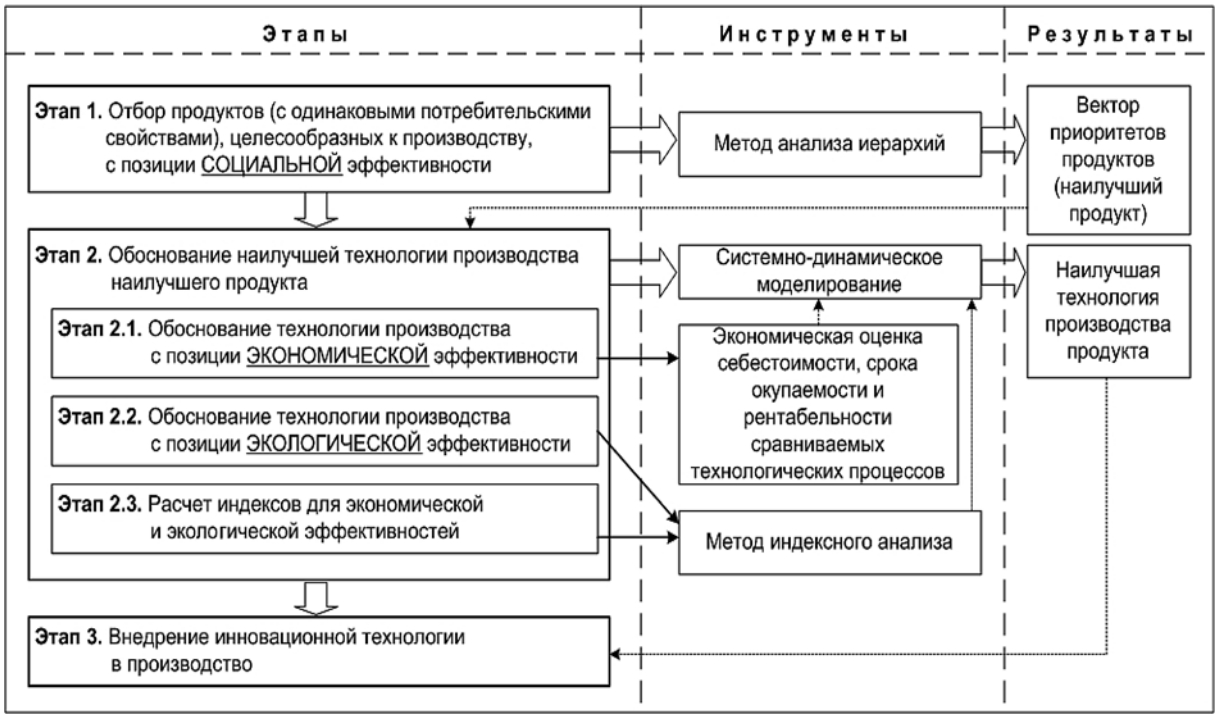

Рис. 2. Концептуальная схема подхода к комплексной оценке эффективности инновационных проектов

вреда для человека. Наибольший вес в векторе приоритетов будет иметь тот препарат, который имеет максимальный социальный эффект.

Этап 2 направлен на обоснование наиболее перспективной технологии производства данного продукта и состоит из трех подэтапов. Сначала происходит оценка экономической эффективности производства данного продукта на базе использования трех показателей — чистого дисконтированного дохода, срока окупаемости и индекса рентабельности. В качестве примера можно привести различные технологии производства отобранного в качестве наилучшего на этапе 1 лекарственного препарата. На следующем шаге необходимо оценить экологическую эффективность выбранного типа производства наилучшего препарата с учетом показателей загрязнения окружающей среды. На этапе 2.3 осуществляется расчет индексов по экономической и экологической эффективностям производства продукта. Для оценки индексов экономической эффективности сопоставляются показатели чистого дисконтированного дохода, срока окупаемости и индекса рентабельности по сравниваемым технологиям производства. Это могут быть технологии синтеза действующих веществ для производства лекарственного препарата в лабораторных условиях и синтеза этих веществ на основе биосырья, полученного методом сбора в естественных природных условиях. Если значения индексов превосходят единицу, значит, сравниваемая технология (показатель которой находится в числителе) является более экономически эффективной. Экологическая эффективность представляет собой оценку влияния технологического процесса производства продукта на окружающую среду и ее степень загрязненности. Детерминантами экологической эффективности выступают показатели загрязненности почв, воздуха, воды. Это показатели опустынивания и ухудшения качества почв, снижения урожайности и др. Синтез индекса экологической эффективности реализуется на основе попарного сопоставления соответствующих показателей по сравниваемым технологиям. Приоритетная технология производства должна иметь характеристики степени загрязненности с меньшими значениями по сравнению с конкурирующей (менее экологичной) технологией.

На этапе 3 "Внедрение инновационной технологии в производство" реализуется процесс внедрения в производство наилучшей технологии производства наилучшего продукта. Этап затрагивает организационные и экономические механизмы реализации производственного процесса.

4. Имитационная модель системного синтеза эффективности инновационного проекта

Для моделирования процессов комплексной оценки эффективности реализации инновационного проекта предприятия в условиях неопределенности ряда факторов и сложных связей между ними как во времени, так и в пространстве использованы подход системно-динамического моделирования и программный продукт имитационного моделирования Anylogic. Преимущество применения метода имитационного моделирования при решении задачи системного синтеза эффективности инновационных проектов состоит в том, что можно достаточно глубоко анализировать сложные причинно-следственные связи факторов экономической, экологической

и социальной эффективностей, исследовать протекаемые в системе процессы в условиях стохастичности факторов, проводить множество экспериментов при различных значениях входных переменных и параметров.

Далее в качестве объекта, на котором будет иллюстрироваться предлагаемый автором подход, рассмотрим инновационный проект "Биотехнологическое производство сырья для лекарственного препарата". Целью данного проекта является высокотехнологичное производство лекарственного препарата для лечения аритмии. Его новизна состоит в синтезе в лабораторных условиях корней аконита и извлечения из них вещества лаппаконитина как основного действующего вещества, входящего в лекарственные препараты для лечения аритмии. Новизна данной технологии заключается в отказе от сбора в естественных природных условиях корней аконита, что способствует сохранению биомассы растений.

Эффективность этого проекта зависит от множества разноразмерных факторов, которые с течением времени подвержены изменению. Для того чтобы обеспечить возможность проведения ряда экспериментов для оценки влияния этих изменений на отдельные составляющие комплексной эффективности, сформирована системно-динамическая модель оценки влияния факторов на результирующие показатели эффективности проекта (рис. 3). В качестве результирующих показателей при моделировании используются показатели, связанные с экономической эффективностью. К ним относятся чистый дисконтированный доход (ЧДД), срок окупаемости, индекс рентабельности и показатель, характеризующий экологическую эффективность, — индекс экологической эффективности.

В рамках имитационных экспериментов исследовали влияние ряда факторов на показатели эффективности проекта и на их чувствительность к изменению этих факторов. К их числу относятся: объем производства корней; тариф на электроэнергию; производственная площадь, необходимая для размещения лабораторного оборудования.

Выбранные факторы обоснованы следующими обстоятельствами. В первую очередь, "тариф на электроэнергию" важен, так как получаемые на его основе и совокупные затраты на электроэнергию в структуре себестоимости производимой продукции (корней аконита) занимают существенное значение, около 50 %. Фактор "производственная площадь" и "объем производства" важны с точки зрения значительного воздействия на рынок и потребление готовой продукции, получаемой из корней аконита. Объемы спроса на эту продукцию составляют 12 000 т корней аконита в год. На площади размером 20 м^2 можно разместить 8 биореакторов, обеспечивающих производство 600 кг корней в год, из которых будет произведено 4 млн упаковок лекарственного препарата. Потребность в лекарственном препарате данного класса составляет 90 млн упаковок в год.

Серия имитационных экспериментов была проведена в зависимости от изменения трех указанных факторов. Изучали как изменение каждого фактора в отдельности, так и их совместное изменение, что влияет на результирующие показатели эффективности проекта. Результаты имитационных экспериментов в различных комбинациях отдельного изменения факторов и их совместном изменении представлены в таблице.

Анализ результатов имитационных экспериментов показал, что последствия сценария 4, а именно увеличение объема производства корней на 12 % в год, приводят к росту индекса экологической эффективности, индекса рентабельности и чистого дисконтированного дохода. Однако при этом уменьшается срок окупаемости проекта. Последствия сценария 11, т. е. рост тарифа на электроэнергию на 12 %, ведут к уменьшению индекса экологической эффективности и росту срока окупаемости, уменьшению чистого дисконтированного дохода проекта и индекса рентабельности проекта. Сценарий 18, напротив, при незначительном 12 %-ном росте производственных площадей обеспечивает увеличение индекса экологической эффективности и чистого дисконтированного дохода проекта, снижает срок окупаемости проекта. Одновременное использование двух механизмов (сценарий 24) — рост объема производства корней на 10 % и увеличение производственной площади на 15 % — обеспечивает больший прирост индекса экологической эффективности, чистого дисконтированного дохода и индекса рентабельности при одновременном снижении срока окупаемости даже при 5 %-ном росте тарифа на электроэнергию.

Таким образом, поэтапная реализация предлагаемого подхода к комплексной оценке эффективности инновационного проекта "Биотехнологическое производство сырья для лекарственного препарата" позволяет отметить ряд его преимуществ. По сравнению с конкурирующей технологией, состоящей в производстве лекарственного антиаритмического препарата из корней аконита, полученных в естественных природных условиях, технология его лабораторного синтеза имеет определенные преимущества.

- Во-первых, для обеспечения требуемых для российского рынка порядка 4 млн упаковок лекарственного препарата в год требуется производить около 600 кг корней аконита в год. Экологичность новой технологии в 2 раза превышает экологичность старой технологии производства, что является несомненным ее преимуществом.
- Во-вторых, такой объем лекарственного препарата обеспечит максимальную социальную эффективность проекта, состоящую в росте здоровьесбережения населения и качества жизни.
- В-третьих, доказан более высокий уровень экономичности технологии: затраты труда для данного проекта в 5 раз ниже, чем для конкурирующего, себестоимость производства в 2 раза ниже. Расчетные значения показателей экономической эффективности проекта на достаточно высоком уровне — чистый дисконтированный доход — более 800 тыс. руб., срок окупаемости проекта — 42 месяца, индекс доходности проекта — 2,45.

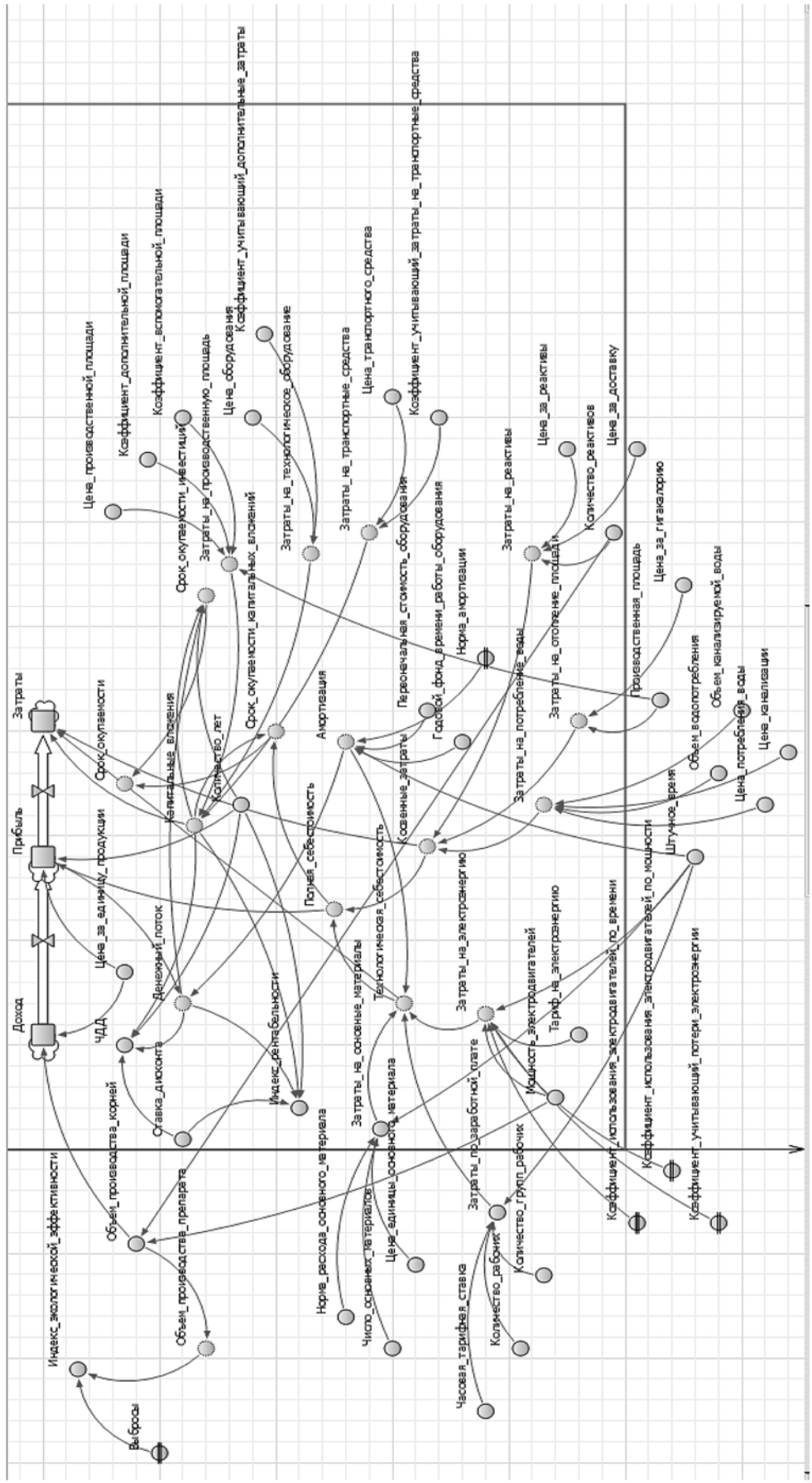

Рис. 3. Системно-динамическая модель оценки комплексной эффективности инновационного проекта

Результаты имитационных экспериментов

Сценарий	Входные показатели			Выходные показатели			
	Объем производства корней, кг/биореактор	Тариф на электроэнергию, руб./(кВт·ч)	Производственная площадь, м^2	Индекс экологической эффективности	Чистый дисконтированный доход, тыс. руб.	Срок окупаемости, мес	Индекс рентабельности
0 (базовый)	5	3,5	20	1,84	800	42	2,45
1	5,05	3,5	20	1,84	801	42	2,45
2	5,25	3,5	20	1,85	805	42	2,46
3	5,5	3,5	20	1,87	810	40	2,48
4	5,6	3,5	20	1,9	850	36	2,54
5	4,75	3,5	20	1,79	795	42	2,43
6	4,5	3,5	20	1,75	790	43	2,41
7	4,25	3,5	20	1,73	787	45	2,37
8	5	3,535	20	1,84	799	42	2,45
9	5	3,675	20	1,95	794	46	2,41
10	5	3,85	20	1,99	791	48	2,38
11	5	3,92	20	1,80	786	52	2,35
12	5	3,325	20	1,82	815	41	2,46
13	5	3,15	20	1,79	824	38	2,51
14	5	2,975	20	1,71	843	35	2,54
15	5	3,5	20,02	1,89	807	40	2,44
16	5	3,5	21	1,95	815	38	2,41
17	5	3,5	22	2,09	829	35	2,39
18	5	3,5	22,4	2,16	845	30	2,36
19	5	3,5	19	1,78	793	45	2,47
20	5	3,5	18	1,63	787	49	2,49
21	5	3,5	17	1,58	774	52	2,51
22	5,05	3,675	21	1,85	802	41	2,46
23	5,05	3,15	21	1,91	815	38	2,52
24	5,6	3,675	23	1,99	822	37	2,54
25	4,75	3,85	20,02	1,83	798	44	2,42
26	4,5	3,92	22	1,61	753	53	2,31
27	5,05	2,975	19	1,89	815	41	2,44
28	4,75	3,535	22,4	1,82	803	40	2,46
29	5,6	3,92	22	1,92	821	38	2,51
30	4,25	2,975	17	1,78	784	45	2,50

Заключение

В ходе проведенного исследования установлено, что традиционные подходы и методы комплексной (системной) оценки эффективности предприятия (проекта) не обеспечивают одновременный учет показателей социальной, экономической и экологической эффективностей. Они не могут быть использованы при оценке эффективности проектов, имеющих социальную направленность, и не позволяют диагностировать проекты с позиции их конкурентоспособности и экологичности.

Разработан комплексный многомерный критерий комплексной оценки эффективности инновационных проектов, обеспечивающий обоснование качества проекта и его количественную оценку с учетом социальной, экономической и экологической составляющих его эффективности. Предложена концептуальная схема подхода к оценке эффективности проектов на основе принципа нелинейного синтеза составляющих показателей эффективности, отражающая поэтапный процесс ее идентификации. Данная схема ранее не была представлена в существующих методах оценки эффективности инновационных проектов.

Предложена системно-динамическая модель комплексной оценки эффективности инновационного проекта по производству сырья для антиаритмического лекарственного препарата. Такая модель позволяет осуществить ситуационный анализ и моделирование влияния изменений факторов эффективности проекта. Модель отличается от существующих тем, что обеспечивает учет мультипликативного влияния факторов составляющих эффективности и позволяет воспроизводить системный (синергетический) эффект от внедрения проекта.

Проведены экспериментальные исследования предложенного подхода на фактических данных о реализации конкретного проекта и доказана достоверность предлагаемых теоретических положений и разработанного метода комплексной оценки эффективности инновационного проекта. Результаты апробации предлагаемых теоретических положений и подхода к комплексной оценке эффективности инновационного проекта имеют практическую значимость.

Список литературы

1. **Pigou A. C.** The economics of welfare, 1920. URL: http://www.econlib.org/library/NPDBooks/Pigou/pgEW1.html (дата обращения 25.04.2019).
2. **Leibenstein H.** Allocative Efficiency and X-Efficiency // The American Economic Review. 1966. Vol. 56. P. 392—415.
3. **Друкер П.** Эффективное управление. М.: Издательско-торговый дом "Гранд", 2003. 95 с.
4. **Алле М.** Условия эффективности в экономике. М.: НИЦ "Наука для общества", 1998. 299 с.
5. **Arrow K. J.** The Potentials and Limits of the Market in Resource Allocation. Chapter in book: / Eds. G. R. Feiwel // Issues in Contemporary Microeconomics and Welfare, 1985. P. 107—124.
6. **North D. C.** Economic Performance through Time. Lecture to the memory of Alfred Nobel, December 9, 1993. URL: https://www.nobelprize.org/prizes/economic-sciences/1993/north/lecture/ (дата обращения 15.08.2019)
7. **Норт Д.** Институты, институциональные изменения и функционирование экономики. М.: Фонд экономической книги "Начала", 1997. 180 с.
8. **Нейман Д., Моргенштерн О.** Теория игр и экономическое поведение. М.: Наука, 1970. 780 с.
9. **Канторович Л. В.** Математико-экономические работы. Новосибирск: Наука, 2011. 760 с.
10. **Блауг М.** Экономическая мысль в ретроспективе. М.: Дело Ltd, 1994. 720 с.
11. **Гонтарева И. В., Нижегородцев Р. М.** Системная эффективность предприятия: сущность, факторы, структура. М.—Киров: ВСЭИ, 2012. 182 с.
12. **Лившиц В. Н.** Основы системного мышления и системного анализа. М.: Институт экономики РАН, 2013. 54 с.
13. **Анфилатов В. С., Емельянов А. А., Кукушкин А. А.** Системный анализ в управлении. М.: Финансы и статистика, 2008. 358 с.
14. **Клейнер Г. Б.** Стратегия предприятия. М.: Дело АНХ, 2008. 568 с.
15. **Клейнер Г. Б.** Системная экономика как платформа развития современной экономической теории // Вопросы экономики. 2013. № 6. С. 4—28.
16. **Флейшман Б. С.** Теория потенциальной эффективности сложных систем. М.: Совет. радио, 1971. 224 с.
17. **Сухарев О. С.** Теория эффективности экономики. М.: Финансы и статистика. 2009. 368 с.
18. **Сухарев О. С.** Экономический рост, институты и технологии. М.: Финансы и статистика, 2014. 464 с.
19. **Сухарев О. С.** Теория дисфункции экономических систем и институтов. М.: Ленард, 2014. 144 с.
20. **Волкова В. Н., Денисов А. А.** Теория систем. М.: Высш. шк., 2006. 511 с.
21. **Новиков Д. А.** Теория управления организационными системами. М.: Физматлит, 2012. 604 с.
22. **Угольницкий Г. А.** Управление устойчивым развитием активных систем. Ростов-на-Дону: Изд-во ЮФУ, 2016. 940 с.
23. **Орлова Е. В.** Оценка экономической эффективности технических решений в дипломных проектах: учеб. пособие. Уфа: УГАТУ, 2009. 100 с.
24. **Орлова Е. В.** Системный анализ и моделирование экономической эффективности проектов: методический подход // Экономика и предпринимательство. 2013. № 12—4. С. 550—558.
25. **Орлова Е. В.** Управление эффективностью предприятия // Проблемы теории и практики управления. 2014. № 6. С. 123—129.
26. **Орлова Е. В.** Оценка кредитного риска на основе методов многомерного анализа // Компьютерные исследования и моделирование. 2013. Т. 5, № 5. С. 893—901.
27. **Орлова Е. В.** Идентификация и прогнозирование рисков экономической системы на основе имитационного моделирования // Проблемы анализа риска. 2014. Т. 11, № 1. С. 40—49.
28. **Орлова Е. В.** Механизмы принятия решений в многоагентных экономических системах: системно-синергетический подход. Уфа: УГАТУ, 2016. 187 с.
29. **Орлова Е. В.** Механизм, модели и алгоритмы управления производственно-экономическими системами на принципах согласования критериев заинтересованных агентов // Программная инженерия. 2016. № 2. С. 86—96.
30. **Orlova E. V.** Modeling and Coordinated Control for the Production and Economic System // Proceedings of the Mathematical Modeling Session at the International Conference Information Technology and Nanotechnology (MM-ITNT 2017). Samara, Russia, 2017. Vol. 1904. P. 1—6.

Engineering of System Synthesis for Innovative Projects Efficiency

E. V. Orlova, ekorl@mail.ru, Ufa State Aviation Technical University, Ufa, 450008, Russian Federation

Corresponding author:

Orlova Ekaterina V., Dr. of Sc., Professor, Ufa State Aviation Technical University, Ufa, 450008, Russian Federation
E-mail: ekorl@mail.ru

Received on August 20, 2019
Accepted on September 09, 2019

The problem of a comprehensive assessment of the effectiveness of enterprises' innovative activities is considered. It was found that traditional approaches and methods of a comprehensive (system) assessment of the enterprises' (project) efficiency do not provide simultaneous using of different indicators like social, economic and environmental effectiveness and cannot be used to evaluate an projects efficiency with a social orientation, and do not compare projects from a position their competitiveness and environmental friendliness.

The multidimensional criterion has been developed for a comprehensive assessment of innovative projects efficiency, providing a justification for the quality of the project and its quantitative assessment, taking into account the social, economic and environmental components of its efficiency. A conceptual scheme of the approach to assessing innovation projects efficiency on the basis of the principle of non-linear synthesis of indicators, which reflects the multi-step process for its identification, has been proposed. This scheme was not previously presented in existing methods for innovative projects efficiency evaluation.

The system-dynamic model for a comprehensive assessment of an innovative project efficiency for the production of raw materials for an antiarrhythmic drug is proposed. This model allows to draw the situational analysis of the factors' changes impact into the project efficiency. The model differs from the existing ones in that it takes into account the multiplicative influence of the factors that allows project manager to reproduce the systemic (synergetic) effect of the project.

Experimental studies of the proposed approach on the actual data of a specific project were carried out and the reliability of the proposed theoretical principles and the developed method of a comprehensive assessment of innovative project efficiency were proved. The testing results of the proposed theoretical principles and approach have practical significance.

Keywords: *engineering of integrated project efficiency, innovative projects, systemic synthesis, simulation modeling*

For citation:

Orlova E. V. Engineering of System Synthesis for Innovative Projects Efficiency, *Programmnaya Ingeneria,* 2019, vol. 10, no. 11—12, pp. 430—439.

DOI: 10.17587/prin.10.430-439

References

1. **Pigou A. C.** *The economics of welfare*, 1920, available at: http://www.econlib.org/library/NPDBooks/Pigou/pgEW1.html (accessed 04.25.2019).
2. **Leibenstein H.** Allocative Efficiency and X-Efficiency, *The American Economic Review*, 1966, vol. 56, pp. 392—415.
3. **Drucker P.** *Effective management,* Moscow, Publishing and trading house "Grand", 2003, 95 p. (in Russian).
4. **Alle M.** *Conditions of efficiency in the economy*, Moscow, SIC "Science for society", 1998, 299 p. (in Russian).
5. **Arrow K. J.** *The Potentials and Limits of the Market in Resource Allocation*. Chapter in book: ed. G. R. Feiwel. Issues in Contemporary Microeconomics and Welfare, 1985, pp. 107—124.
6. **North D. C.** *Economic Performance through Time*. Lecture to the memory of Alfred Nobel, December 9, 1993, available at: https://www.nobelprize.org/prizes/economic-sciences/1993/north/lecture/
7. **North D.** *Institutions, institutional change and the functioning of the economy*, Moscow, Fund of the economic book "Beginnings", 1997, 180 p.
8. **Neumann D., Morgenstern O.** *Game theory and economic behavior*, Moscow, Nauka, 1970, 780 p. (in Russian)
9. **Kantorovich L. V.** *Mathematical and economic work*, Novosibirsk, Nauka, 2011, 760 p. (in Russian).
10. **Blaug M.** *Economic thought in retrospect*, Moscow, Delo Ltd, 1994, 720 p. (in Russian).
11. **Gontareva I. V., Nizhegorodtsev R. M.** *Systemic effectiveness of an enterprise: essence, factors, structure*, Moscow—Kirov, All-Russian Higher Economic Research Institute, 2012, 182 p. (in Russian).
12. **Livshits V. N.** *Fundamentals of systems thinking and systems analysis*, Moscow, Institute of Economics, RAS, 2013, 54 p. (in Russian).
13. **Anfilatov V. S., Emelyanov A. A., Kukushkin A. A.** *System analysis in management,* Moscow, Finance and Statistics, 2008, 358 p. (in Russian).
14. **Kleiner G. B.** *Strategy of the enterprise*, Moscow, Delo ANH, 2008, 568 p. (in Russian).
15. **Kleiner G. B.** System economics as a platform for the development of modern economic theory, *Ekonomika,* 2013, no. 6, pp. 4—28 (in Russian).
16. **Fleishman B. S.** *The theory of the potential effectiveness of complex systems,* Moscow, Soviet Radio, 1971, 224 p. (in Russian).
17. **Sukharev O. S.** *Theory of economic efficiency,* Moscow, Finance and statistics, 2009, 368 p. (in Russian).
18. **Sukharev O. S.** *Economic growth, institutions and technology,* Moscow, Finance and Statistics, 2014, 446 p. (in Russian).
19. **Sukharev O. S.** *The theory of dysfunction of economic systems and institutions,* Moscow, Lenard, 2014, 144 p. (in Russian).
20. **Volkova V. N., Denisov A. A.** *Theory of systems,* Moscow, Higher school, 2006, 511 p. (in Russian).
21. **Novikov D. A.** *Theory of management of organizational systems,* Moscow, Fizmatlit, 2012, 604 p. (in Russian).
22. **Ugolnitsky G. A.** *Management of sustainable development of active systems,* Rosto-on-Don, Publisher SFU, 2016, 940 p. (in Russian).
23. **Orlova E. V.** *Evaluation of the economic efficiency of technical solutions in graduation projects,* Ufa, USATU, 2009, 100 p. (in Russian).
24. **Orlova E. V.** System analysis and modeling of economic efficiency of projects: a methodological approach, *Ekonomika i predprinimatel'stvo*, 2013, no. 12—4, pp. 550—558 (in Russian).
25. **Orlova E. V.** Management of enterprise efficiency, *Problemy teorii i praktiki upravleniya*, 2014, no. 6, pp. 123—129 (in Russian).
26. **Orlova E. V.** Credit risk assessment based on multivariate analysis methods, *Komp'yuternyye issledovaniya i modelirovaniye*, 2013, vol. 5, no. 5, pp. 893—901 (in Russian).
27. **Orlova E. V.** Identification and forecasting of risks of the economic system based on simulation, *Problemy analiza riska,* 2014, vol. 11, no. 1, pp. 40—49 (in Russian).
28. **Orlova E. V.** *Decision-making mechanisms in multi-agent economic systems: a system-synergetic approach,* Ufa, USATU, 2016, 187 p. (in Russian).
29. **Orlova E. V.** The mechanism, models and control algorithms of production and economic systems on the principles of matching criteria of interested agents, *Programmnaya Ingeneria*, 2016, no. 2, pp. 86—96 (in Russian).
30. **Orlova E. V.** Modeling and Coordinated Control for the Production and Economic System, *Proceedings of the Mathematical Modeling Session at the International Conference Information Technology and Nanotechnology (MM-ITNT 2017)*, Samara, Russia, 2017, vol. 1904, pp. 1—6.

УДК 004.4, 004.7 DOI: 10.17587/prin.10.440-450

С. А. Марченков, аспирант, мл. науч. сотр., marchenk@cs.petrsu.ru,
Петрозаводский государственный университет

Автоматизация процессов программирования агентов на основе кодогенерации при построении семантических сервисов интеллектуальных пространств. Часть 2*

Настоящая работа является частью 2 статьи, опубликованной ранее. Основным результатом, который представлен в части 1 статьи, является проектное решение по созданию генератора программного кода агентов для объектно-ориентированных языков программирования на основе онтологий сервисов. На основе предложенного решения для достижения цели по упрощению разработки и сопровождения приложений интеллектуальных пространств в настоящей работе введено унифицированное онтологическое описание семантики процессов построения сервисов. Представлены алгоритмы автоматизации процессов программирования агентов для реализации структур объектной модели данных и взаимодействия агентов при программировании сервисов.

Ключевые слова: *интеллектуальные пространства, семантические сервисы, онтологическая модель, объектная модель данных, кодогенерация*

Введение

Современные тенденции в области разработки распределенных вычислительных сред основываются на концепции Интернета вещей (*Internet of things*, IoT) и определяющих ее технологий [1]. В разработку вовлекается большое число участников, взаимодействующих для решения возникающих задач с использованием разнообразных информационных и технических ресурсов, а также ресурсов самих участников. Для интеграции имеющегося множества независимых ресурсов в целях качественного и количественного развития предоставления цифровых сервисов в IoT-средах создают специализированные информационные окружения [2]. Предоставляемая информационная поддержка в таких окружениях может быть расширена за счет использования контекстно-зависимых сервисов с привлечением элементов окружающего интеллекта. В результате создаются так называемые "интеллектуальные пространства" (ИП) — технологически оснащенные цифровые информационно-вычислительные среды, которые способны распознавать текущую ситуацию, анализировать поведение человека и удовлетворять его потребности, когда в этом возникает необходимость, и, по возможности, делать это в упреждающем режиме [3].

Архитектура М3 определяет основные принципы организации ИП с точки зрения программной инженерии, основываясь на технологиях Интернета вещей и методах Семантического веба [4, 5]. Аббревиатура М3 указывает на свойства *multi-device* (множество устройств), *multi-vendor* (множество производителей аппаратуры) и *multi-domain* (множество предметных областей). Программные агенты, именуемые процессорами знаний (*Knowledge Processor*, далее — агент KP), являются основными программными компонентами и участниками взаимодействия. Агенты KP выполняют динамическое построение общего информационного содержимого (ОИС), представленного в виде RDF-графа. В ОИС создаются семантически связанные виртуальные образы участников, задействованных ресурсов и происходящих процессов. Организация ИП направлена на создание сервис-ориентированных приложений для различных предметных областей (совместная деятельность, цифровые музеи и др.). Построение таких сервисов реализуется как распределенный вычислительный процесс на основе событийно- и информационно-управляемого взаимодействия автономных агентов [6].

Сервис в ИП-приложениях представляет собой распределенную систему агентов KP, которая реализует предписанную функцию с привлечением доступных ресурсов. Разработка сервисов основывается на стандартах, определенных в концепции Семантического веба с ориентацией на локализованные IoT-среды, а не на всю систему веб-ресурсов [7]. В результате сервис рассматривается как программная система, доступ к которой предоставляется с помощью описанного заранее интерфейса и осуществляется в соответствии с ограничениями и политиками, определенными в описании сервиса, а построение следует заданному набору процессов. Сервис определяется

* Часть 1 опубликована в журнале "Программная инженерия". 2019. Т. 10, № 6. С. 257—264.

как семантический, т. е. имеет однозначно описанную семантику, доступен в других информационных окружениях, применим для автоматизированного поиска, композиции, построения и доставки.

Работа является частью 2 статьи [8], где сформулированы проблемные вопросы и требования, связанные с упрощением процессов разработки и сопровождения ИП-приложений за счет создания средств автоматизации процессов программирования агентов при построении семантических сервисов. В настоящей работе предложены решения, направленные на удовлетворение выделенных ранее требований на пути к созданию генератора программного кода агентов для объектно-ориентированных языков программирования на основе онтологий сервисов. За счет расширения онтологии OWL-S введено унифицированное онтологическое описание семантики процессов построения сервисов. Такое расширение позволяет наделить их качествами семантических сервисов и сделать пригодными для автоматизации процессов программирования. Предложены алгоритмы автоматизации процессов программирования агентов для реализации структур объектной модели данных и взаимодействия агентов при программировании сервисов на основе генерации программного кода с использованием онтологии сервиса. Полученные результаты рассмотрены на представленных в части 1 [8] сервисах для интеллектуальных окружений: сервисе определения и анализа активности пользователей (интеллектуальный зал) и сервисе совместного обогащения информационного содержимого историческими данными (умный музей).

Онтологическая модель сервиса

Интеллектуальные пространства позволяют эффективно организовать взаимодействие его участников и совместное использование ими информации [3]. В частности, для каждого участника создается виртуальный цифровой образ, который семантически связывается с другими участниками и ресурсами. Онтологическая модель сервиса для ИП позволяет описывать контекст окружений и его участников, взаимодействующих агентов и задействованных ресурсов. Онтологическая модель строится на основе концептуального описания семантических веб-сервисов с использованием моделей организации информационно-управляемого взаимодействия агентов [9], определенных в рамках подхода ИП. За счет такого унифицированного онтологического описания процессов построения сервиса достигается повышение качества проектирования сервисов. Сервисы становятся пригодными для автоматизированного поиска, композиции, построения и доставки пользователям посредством описания интерфейса (назначение сервиса, входные и выходные данные, и т. д.) и происходящих процессов.

Известна онтология OWL-S, разработанная на основе OWL для описания семантических веб-сервисов [10, 11]. Основная цель онтологии OWL-S — предоставить пользователям возможность с высокой степенью автоматизации обнаруживать, выбирать, вызывать, составлять, отслеживать и контролировать веб-ресурсы. Онтология OWL-S позволяет описывать характеристики сервисов с детализацией с использованием трех концепций верхнего уровня, представленных в виде следующих онтологий: профиль сервиса (*service profile*); модель процесса сервиса (*service model*); основание сервиса (*service grounding*).

Назначение профиля сервиса — определить сервис единообразно для дальнейшего использования, с детализацией содержания запросов и условий, при которых будут возникать конкретные результаты, и при необходимости поэтапных процессов, приводящих к этим результатам. Модель процесса описывает, как получать доступ к сервису и что происходит, когда выполняется сервис. Также модель процесса сервиса описывает сервис как набор атомарных и составных процессов. Атомарный процесс соответствует одношаговой процедуре, которая получает входные параметры, обрабатывает их и затем возвращает результат. Составные процессы разлагаются на другие процессы, тем самым определяя их декомпозицию с помощью управляющих конструкций (например, *If-Then-Else*, *Split, Repeat-While*). Основание сервиса на базе протокола взаимодействия и формата сообщений определяет, каким образом сервис вызывается потребителем сервиса. В случае архитектуры M3 [4] онтология основания сервиса является необязательной частью, так как взаимодействие агентов KP организуется посредством протокола SSAP, который не требует отдельного описания.

Предлагается унифицированное онтологическое описание семантики сервисов для ИП, содержащее набор терминологических аксиом (рис. 1). Далее опишем только основные онтологические понятия и их отношения, приведенные на рис. 1, остальные опустим в силу ненадобности. Профиль сервиса в онтологии описывает сервис с помощью модели IOPEs: входные параметры (input), выходные параметры (output), предварительные условия (precondition), эффекты выполнения (effects). Свойство has_output используется для представления выходных данных от ресурса сервиса к потребителю. Задание входных данных для обработки с помощью ресурса или управляющего воздействия в виде уведомлений определяется с помощью свойства has_input.

Исходное (начальное) состояние ОИС для инициализации процесса построения сервиса задается с помощью предварительного условия (has_precondition). Желаемое состояние ОИС определяется с помощью результирующего условия (has_result). Предусловия и результаты задаются с помощью логических выражений. Существует несколько возможных подходов, которые позволяют использовать правила и логику для RDF/OWL-представления [12]. Ключевой идеей является рассмотрение логических выражений как литералов (literals) с использованием SPARQL-выражений, которые определяются с помощью класса *S*PARQL-Expression.

Свойство has_category описывает категории сервисов, основанные на категориях сервисов для ИП. Категории определяются в зависимости от таких критериев, как оказываемое воздействие, основополагающий ресурс,

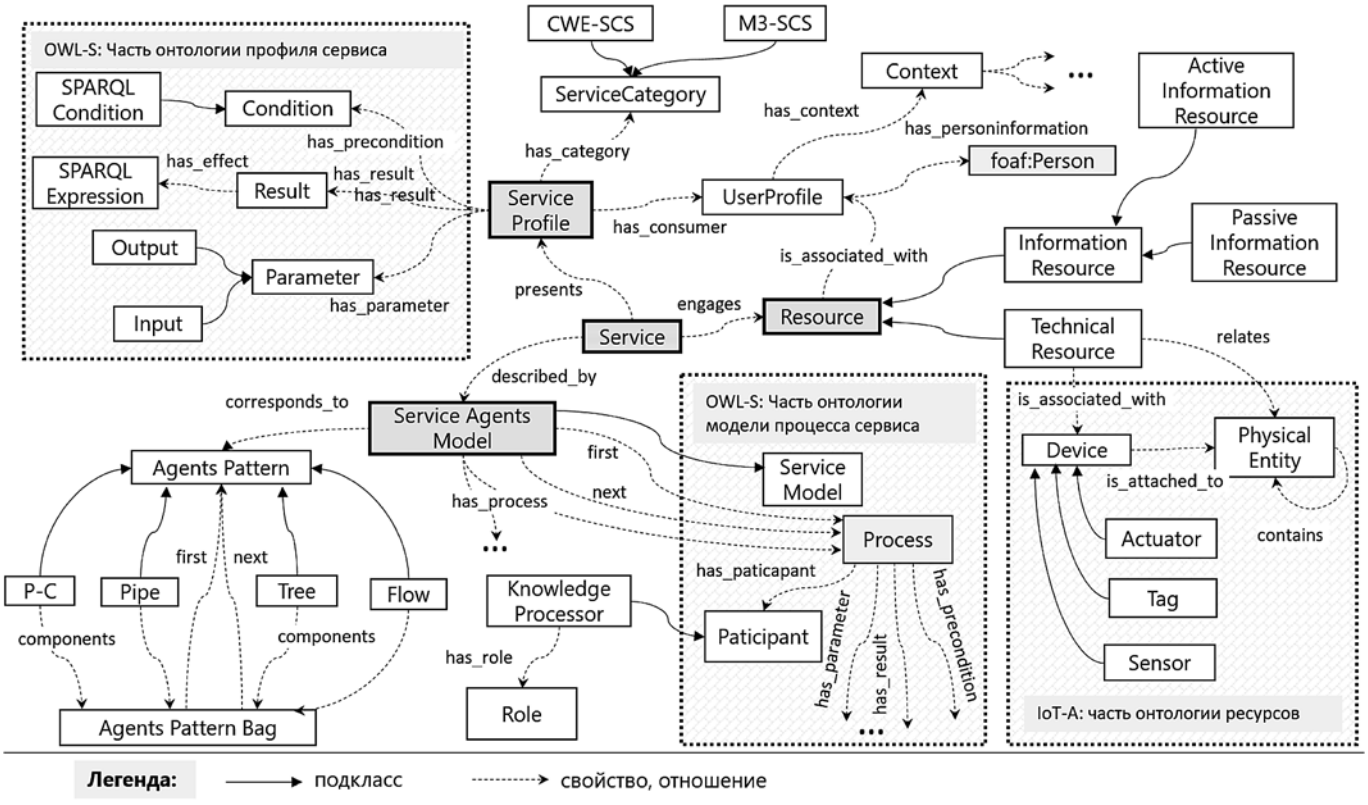

Рис. 1. Онтология семантического сервиса для ИП

способ интеграции, выполняемая функция. Категория определяет типовую модель информационно-управляемого взаимодействия агентов КР.

Онтология OWL-S довольно гибкая, однако решений с ее использованием недостаточно, чтобы без изменений применять ее для описания семантики сервисов для ИП. Вводятся новые сущности, чтобы учесть приоритеты и предпочтения потребителя (если потребитель является человеком), что обеспечивает персонификацию сервиса. Существующая онтология профиля расширяется свойством has_consumer и его объектом класса UserProfile, который описывается в терминах пользовательской контекстной информации. Такая информация отражает состояние окружения пользователя (has_context), а также его предпочтения, интересы, персональную информацию, представленные частью онтологии FOAF (has_personinformation) [13]. Онтология OWL-S также не поддерживает многоагентный подход для описания модели построения и доставки сервиса. Ви́дение агентов в онтологии OWL-S является редуцированным: агенты учитываются, прежде всего, как потребители или "искатели" сервисов. В то время как процесс построения сервиса в парадигме ИП представляет собой процесс взаимодействия нескольких агентов.

Онтология модели процесса сервиса расширяется за счет внедрения класса Service Agents Model для описания процесса построения сервисов для ИП. Процесс описывается моделью информационно-управляемого взаимодействия агентов КР, которая определяется в зависимости от категории сервиса. Взаимодействие каждого агента (Knowledge Proccessor), представленного как расширение класса участника, определяется его функциональной ролью (свойство has_role), выполняемой в модели. Функциональная роль — абстрактное описание функциональных свойств агента. Роль агента позволяет определить общие принципы реализации индивидуальной внутренней логики агента, а также принципы взаимодействия с другими агентами. Логика отдельного агента ведет к взаимодействию агентов во время построения сервиса на основе шаблонов взаимодействия (Agents Pattern), описанных с помощью таких архитектурных абстракций, как поставщик-потребитель (P-C), конвейер (Pipe), дерево (Tree), поток (Flow) [6]. Процесс взаимодействия агентов может состоять из нескольких шаблонов, представленных в некоторой последовательности (Agents Pattern Bag).

Приведем примеры разработанных онтологий на основе представленного унифицированного онтологического описания семантики сервисов для ИП.

Пример 1: *сервис определения присутствия и анализа активности пользователей* (S_{ua}). Сервис использует метод пассивной радиолокации, основанный на измерениях мощности принимаемого сигнала мобильных устройств [14]. Мобильные устройства подключены к беспроводной локальной сети вычислительной среды. Прием и передача информации идут посредством радиосигналов, генерируемых устройствами. Мощность сигнала (RSSI) может быть измерена выделенным сетевым сенсором, подключенным к той же сети. Сервис, используя сетевой

сенсор, позволяет отслеживать, изменять и формировать информацию об уровне присутствия участника, а также периодически обновляет параметры сетевой активности. Сервис может быть представлен как семантический с использованием онтологии сервиса для ИП на основе OWL-S.

Выделенный RSSI-сенсор определения присутствия является основополагающим техническим ресурсом этого сервиса. Сервис является информационным, он предоставляет информацию о состоянии сенсора и его измерениях в ИП, а также обнаруживает новые знания об активности участников на основе этой информации. На рис. 2 представлена часть онтологии данного сервиса, включающей набор утверждений об индивидах. Экземпляр класса Service Profile (#Profile_UserActivity_Service) предоставляет необходимую информацию о функциональном описании и категориях сервиса. Например, входные параметры определяются следующими индивидами: мощность сигнала (ua;#RSSI), измеренная RSSI-сенсором; зарегистрированные MAC-адреса мобильных устройств (ua;#MAC). Каждый процесс сервиса определяется с помощью необходимых индивидов и их свойств с указанием модели IOPEs.

Описанные далее агенты КР и их процессы участвуют в построении сервиса.

- Агент-адаптер КР сенсора (PresenceProcessorKP) взаимодействует с сетевым RSSI-сенсором, который работает на выделенном компьютере и включает в себя функциональные возможности HTTP-сервера для получения от сенсора измерений в формате JSON. Агент обрабатывает полученные данные сенсора (временная метка, MAC-адрес, значение RSSI) и публикует их в ОИС. Агент принимает участие в реализации следующих процессов, представленных экземплярами класса AtomicProcess (атомарный процесс): процесс UpdatePresenceInforamtion (обновление информации о присутствии пользователя); процесс SendActivityInformation (публикация информации об активности в ОИС).
- Агент-агрегатор КР присутствия (PresenceDetectorKP) подписывается на обновление информации о присутствии мобильных устройств. Любое обновление ставится в соответствие конкретному пользователю с помощью наличия в профиле информации о MAC-адресе устройства. В заключительной части мероприятия или по запросу накопленных данных агентом определения присутствия для каждого пользователя вычисляются такие показатели сетевой активности, как уровень сетевой активности, степень активности и среднее значение RSSI. Агент принимает участие в реализации одного процесса, также представленного экземпляром класса AtomicProcess — процесс SetUserPresenceLevel (публикация значений показателей сетевой активности в ОИС).
- Агент-монитор КР активности (ActivityMonitorKP) анализирует вычисленные показатели сетевой активности пользователей. Агент также может визуализировать активность и доставлять ее другим сервисам окружения совместной деятельности для отображения на общественных экранах. Процесс AnalyseUsersActivity (анализ метрик сетевой активности) представлен как экземпляр класса CompositeProcess (составной процесс). Этот составной процесс раскладывается на следующие атомарные (простые) процессы: InitUsersActivity (получение

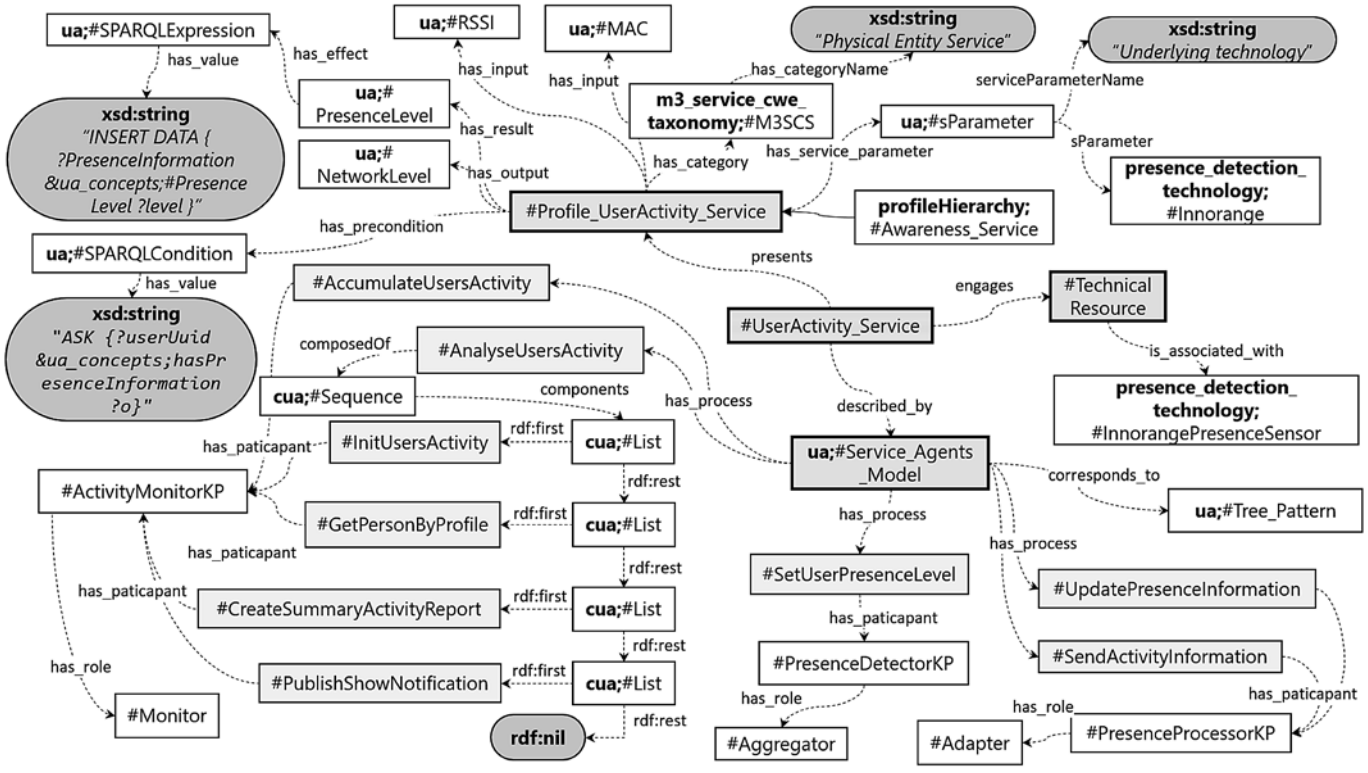

Рис. 2. Часть онтологии, содержащей утверждения об индивидах, для сервиса S_{ua}

с ОИС метрик сетевой активности), GetPersonByProfile (получение информации о пользователе на основе его профиля), CreateSummaryActivityReport (подготовка визуального отчета об активности пользователей в формате PNG) и PublishShowNotification (публикация уведомления в ОИС для оповещения других агентов). Их декомпозиция задается с помощью управляющей конструкции "последовательность" (Sequence). Агент участвует также в реализации одного независимого атомарного процесса, а именно — процесса AccumulateUsersActivity (локальное накопление информации о сетевой активности).

Пример 2: *сервис совместного обогащения информационного содержимого историческими данными* (S_{en}). Сервис обеспечивает обогащение семантической сети окружения умного музея информацией из различных ресурсов, к числу которых относятся интернет-ресурсы исторических данных; индивидуальная информация и исторические знания от посетителей музея; музейная информационная система [15]. Сервис важен для музейных сотрудников, поскольку позволяет дополнять описание хранимых экспонатов различной информацией, делая их представление более содержательным. Посетители также приобретают возможность не только быть потребителем информации при изучении экспонатов, но и пополнять информационное описание.

Часть онтологии данного сервиса, включающей набор утверждений об индивидах профиля и модели процесса, представлена на рис. 3. Функциональные возможности сервиса и его интерфейс описываются с помощью индивида Profile_Enrichment_Service. Сервис является информационным, он предоставляет информацию из различных источников исторических данных. В качестве входного параметра для сервиса указывается URL-адрес для внешней SPARQL-точки доступа или музейной информационной системы (en;#EndpointUrl). Предусловием построения сервиса является наличие информации в ОИС об экспонатах, для которых выполняется процесс обогащения (задается с помощью запроса SPARQL-ASK).

Описанные далее агенты КР и их процессы участвуют в построении сервиса.

- Агент-искатель КР исторической информации (ExternalFinderKP) отвечает за взаимодействие с внешними интернет-ресурсами (например, сервис DBpedia) и музейной информационной системой. Агент обеспечивает процессы совместной деятельности посетителей (организация дискуссии по экспонату, осмотр экспозиции) и объекты музейной экспозиции дополнительной информацией. Агент выступает в качестве семантического посредника, выполняя преобразования информации из внешних источников в RDF-тройки и обратно, формируя семантическую сеть в ОИС. Агент реализует следующие атомарные процессы, направленные на пополнение информации об экспонате: процесс PopulateExhibitInfo (пополнение общей описательной информации), процесс PopulateExhibitRelations (пополнение информации о связях с другими

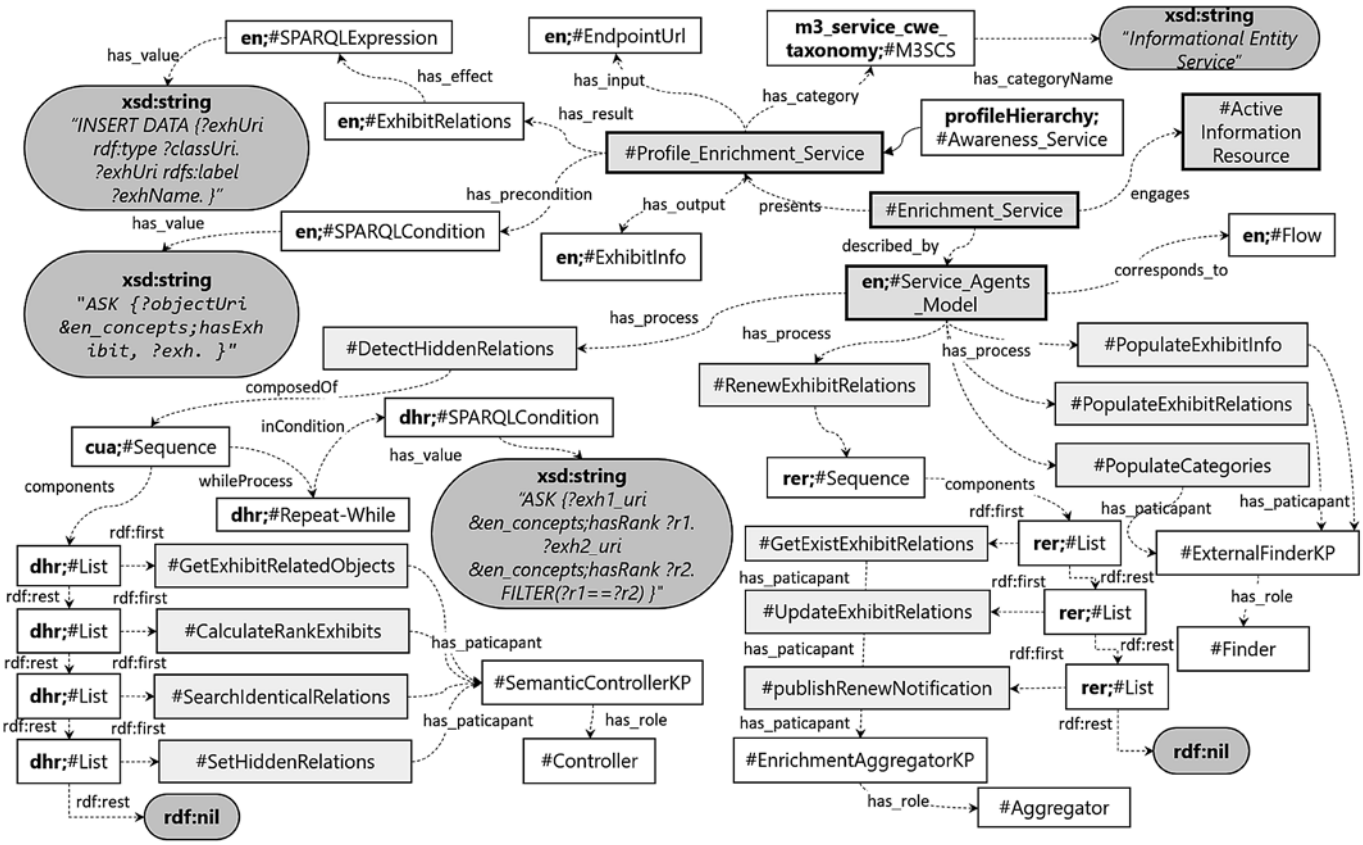

Рис. 3. Часть онтологии, содержащей утверждения об индивидах, для сервиса S_{en}

Таблица 1

Метрики разработанных онтологий сервисов

Сервис	Часть онтологии	Число свойств данных	Число объектных свойств	Число классов и их индивидов	Число RDF-троек
S_{ua}	Профиль	19	6	29	162
	Модель процесса	52	119	136	921
	Итого	71	125	165	1083
S_{en}	Профиль	13	5	19	111
	Модель процесса	83	184	152	1257
	Итого	96	189	171	1368

объектами) и процесс PopulateCategories (пополнение информации о категориях экспоната).

- Агент-агрегатор КР исторической информации (EnrichmentAggregatorKP) анализирует предоставленную пользователями информацию, чтобы обогатить семантическую сеть более подробными описаниями и отношениями. Пользователи могут добавлять описательную информацию об объектах музейной экспозиции, а также их связи с другими объектами. Один из составных процессов RenewExhibitRelations реализует пополнение связей между экспонатами с помощью сведений, полученных от посетителей. Этот составной процесс раскладывается на следующие атомарные процессы: GetExistExhibitRelations (получение существующих связей между экспонатами); UpdateExhibitRelations (обновление связей экспонатов на основе полученных данных от посетителей и других источников); publishRenewNotification (отправка уведомления другим агентам через ОИС). Их декомпозиция задается с помощью управляющей конструкции "последовательность" (Sequence).
- Агент-контроллер КР обнаружения исторических связей (SemanticControllerKP) выполняет связывание виртуальных образов в семантической сети. Агент извлекает узлы семантической сети, представляющие исторические личности и события, музейные экспонаты и профили пользователей, чтобы впоследствии установить новые семантические отношения между ними, используя реализации алгоритмов рассуждения. Примером такого алгоритма является алгоритм семантического сопоставления похожих отношений. Например, для случая экспоната алгоритм выполняет поиск объектов в семантической сети, которые имеют похожие с ним связи. На основе анализа связей для найденных похожих объектов выявляются скрытые связи у целевого экспоната. Алгоритм является итеративным, на каждом его шаге выполняется пересчет рангов. Составной процесс DetectHiddenRelations, реализующий описанный выше алгоритм, раскладывается на следующие атомарные процессы: GetExhibitRelatedObjects (получить связанные с экспонатом объекты); CalculateRankExhibits (вычислить значение рангов объектов); SearchIdenticalRelations (найти у объектов связи, идентичные с целевым экспонатом); SetHiddenRelations (опубликовать в ОИС найденные скрытые связи). Последовательность выполнения этих процессов задается управляющей конструкцией "Цикл с предусловием", где предусловие задается с помощью запроса SPARQL-ASK, реализующего условие выхода из цикла (равенство рангов объектов).

Основные метрики разработанных онтологий для описанных примеров сервисов представлены в табл. 1. Общее онтологическое представление для сервиса S_{ua} и S_{en} примерно сопоставимо, 1083 и 1368 RDF-троек соответственно. Следовательно, в крупномасштабных развертываниях ИП-приложений (несколько десятков сервисов) процесс построения и доставки сервиса может оказаться трудоемким для ресурсно-ограниченных устройств (одноплатные компьютеры, мобильные устройства). Тем не менее хранение всех семантических данных на обладающей большими ресурсами серверной машине и использование различных технологий для поддержки эффективной передачи данных по маломощным и низкоскоростным устройствам и сетям позволяют решить эту проблему [16]. Представленная онтология позволяет получать единообразные онтологические модели сервисов и может использоваться для автоматизации дальнейших процессов программирования сервисов и их агентов.

Алгоритмы автоматизации процессов программирования агентов

Автоматизация процессов программирования агентов КР, участвующих в построении и доставке сервисов, достигается за счет использования генератора программного кода. В результате выполнения этапа проектирования ИП-приложения разработчик имеет набор онтологий, которые делятся на две группы: онтологии проблемной области и онтологии спецификации сервиса для ИП на основе OWL-S. Онтологии предоставляют необходимую семантику, которая используется для генерации кода объектно-ориентированных языков программирования. Генератор, проектное решение которого предложено в части 1 [8], использует алгоритмы автоматизации процессов программирования агентов для реализации структур объектной модели данных и для взаимодействия агентов при программировании сервисов. Алгоритм кодогенерации объектной модели данных агентов принимает в качестве входного параметра онтологию предметной области. Алгоритм кодогенерации взаимодействия агентов принимает в качестве входного параметра онтологию спецификации сервиса для генерации блоков программного кода агентов КР.

Объектная модель объединяет данные и функциональные возможности в переменной абстрактного типа данных (класса) — в объекте. Объектная модель позволяет представить (описать) понятия и объекты реального мира, которые важны для разрабатываемого ИП-приложения. В то время как структура онтологии содержит определения понятий (классов) и отношений между ними (свойства, аспекты, параметры), объектная модель использует классы для представления объектов и функции для моделирования отношений объектов и атрибутов. Сходство онтологии и объектной модели данных позволяет применять объектно-ориентированный подход для моделирования онтологий [17]. Однако онтология представляет собой более богатую информационную модель, чем объектно-ориентированная модель языка программирования (например, Java). Онтология поддерживает такие отличительные особенности, как наследование свойств, симметричные/транзитивные/обратные свойства, полное множественное наследование среди классов и свойств [18].

Принимая во внимание упомянутые выше аспекты и ограничения, а также результаты исследований в этой области [19, 20], в табл. 2 сведены правила отображения, используемые при генерации исходного кода объектной модели данных по онтологии. На рис. 4 представлен разработанный автором алгоритм кодогенерации объектной модели данных агентов, использующий правила отображения из соответствующей сводной таблицы. Основная идея этого алгоритма заключается в создании набора классов и объектов таким образом, чтобы каждый онтологический класс со своими экземплярами, свойствами, слотами, аспектами и ограничениями имел свой эквивалент в структурах объектно-ориентированного языка программирования.

Построение сервиса для ИП может быть рассмотрено как набор вызовов программных функций агентов. Онтология сервиса для ИП на основе

Таблица 2

Сводное изложение правил отображения онтологии в объектную модель

№ правила	Обозначение	Разъяснение
1	*Ontology classes* → *Object classes*	Онтологические классы близки к классам в объектной модели, они представляют абстрактные группы физических или логических объектов, тогда как объектный класс может быть рассмотрен как определение типа для объекта. Классы онтологии отображаются как классы объектов
2	*Instances* → *Objects*	Экземпляры в онтологии используются для представления определенных представителей классов. Класс объекта служит шаблоном описания его объекта. Экземпляры в онтологии отображаются как объекты классов
3	*Data type properties or slots* → *Data attribute variables & get/set methods*	Слоты данных представляют свойства (атрибуты) данных класса онтологии. Эти слоты описываются простыми типами (целочисленные, логические, строковые и т. д.) и наборами значений переменных атрибутов этих типов. Переменные атрибутов в объектной модели описывают характеристики объектов различными типами данных. Свойства данных или слоты отображаются как переменные данных атрибутов вместе с комбинацией методов get/set
4	*Object type properties or slots* → *Object attribute variables & get/set methods*	Объектные свойства или слоты используются для описания взаимосвязи между двумя понятиями (концептами). Первый концепт должен быть экземпляром класса, который является доменом слота (domain), второй концепт — экземпляром класса, который описан допустимым диапазоном слота (range). Объектные свойства или слоты отображаются как объектные переменные атрибутов вместе с комбинацией методов get/set
5	*Value-type/space facets* → *Attribute variables types & if-then-else statements*	Аспекты или грани (facets) для типов данных в онтологии — это некоторое бинарное отношение, которое присоединяется к слоту и описывает, какие типы значений данных могут заполнять слот и какие ограничения они имеют (максимальное и минимальное значения и т. д.). Грани применимы к переменным атрибутов в объектной модели, например, аспект xsd:type может использоваться для обозначения типа переменной, а аспект xsd:lenght может быть представлен набором операторов if-then-else в методе set для соответствующего атрибута, задавая ограничения на длину значения переменной
6	*Cardinality facets* → *Additional attributes & if-then-else statements*	Аспекты или грани кардинальности определяют, сколько значений могут иметь свойство или слот. Некоторые аспекты кардинальности позволяют указать минимальное и максимальное значения кардинальности для более точного описания числа значений слотов. Дополнительные переменные атрибутов могут быть добавлены к объектной модели для указания аспектов кардинальности. Некоторые из этих аспектов могут быть представлены как набор операторов if-then-else в методе set для соответствующего атрибута, ограничивая его кардинальность
7	*Single class inheritance* → *Single object inheritance*	Одиночное наследование классов в онтологии отображается в одиночное наследование классов объектов. Для этого название родительского класса в форме онтологического наследования подставляется в форму наследования для выбранного объектно-ориентированного языка программирования. Родительский класс должен быть определен в объектной модели данных
8	*Multiple inheritance* → *Single inheritance & multiple interface inheritance*	Некоторые языки программирования (например, Java) не поддерживают множественное наследование классов в силу так называемой проблемы ромба (diamond problem). Однако для решения большинства задач может применяться множественное наследование интерфейсов наряду с одиночным наследованием

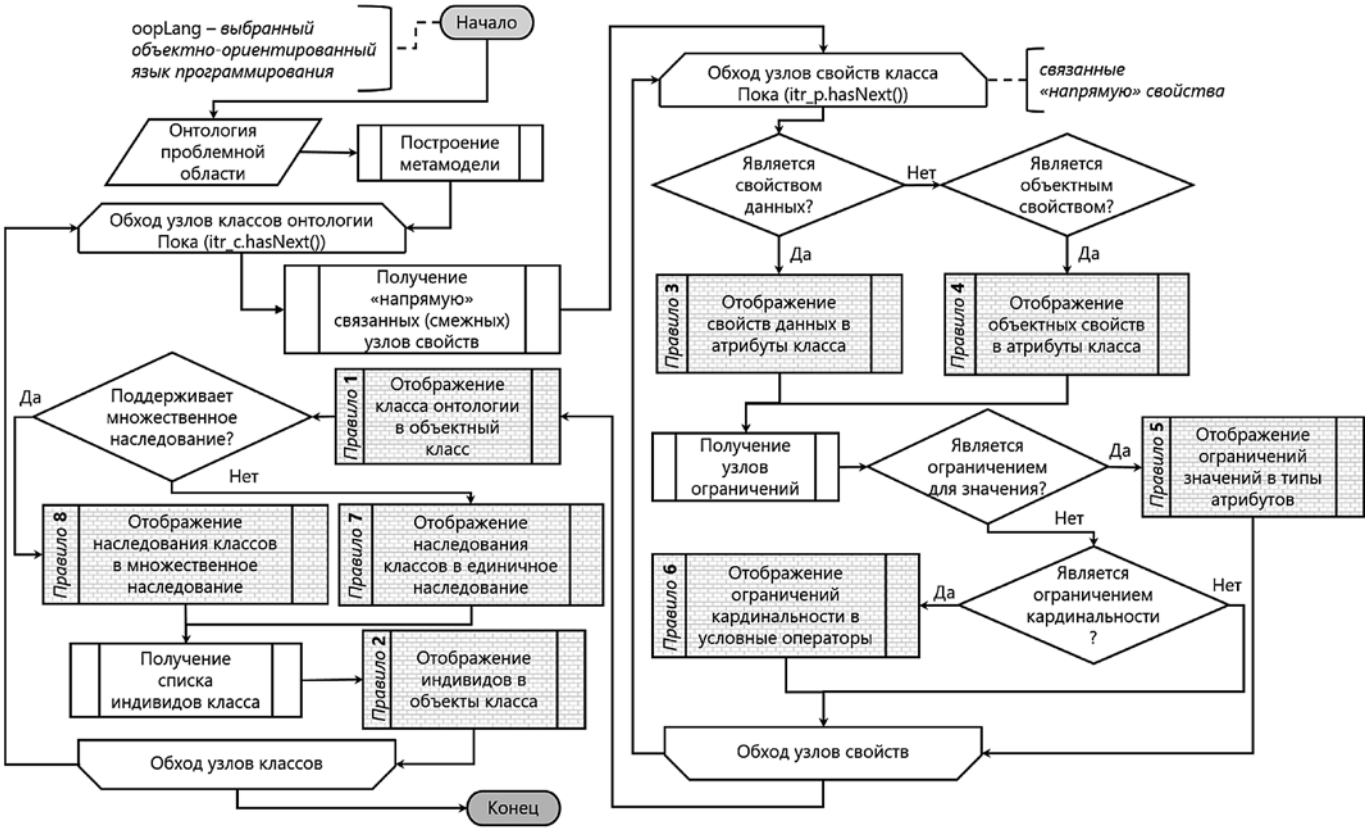

Рис. 4. Алгоритм кодогенерации объектной модели данных агентов

OWL-S предоставляет декларативное, интерпретируемое компьютером описание, включающее семантику модели IOPEs, которая должна быть указана для каждого процесса. Сущность процесса из онтологии сервиса ИП может быть использована для генерации процедур, функций и других элементов целевого языка программирования, реализующих информационно-управляемое взаимодействие и необходимую для этого внутреннюю логику агентов. На рис. 5 с помощью блок-схемы представлен алгоритм кодогенерации взаимодействия агентов.

Экземпляры класса AtomicProcces (атомарный или простой процесс) используются для генерации кода функций. Атрибут rdf:ID класса AtomicProcess определяет имя функции. Каждое свойство has_input с атрибутом rdf:ID соответствует входным параметрам функции. Тип входного параметра может быть получен путем просмотра свойства parameter_type.

Основная внутренняя логика функций реализуется с использованием запросов SPARQL и наборов операторов языка программирования. Свойство has_precondition с выражением SPARQL определяет блок кода для проверки предварительного условия, описывающего начальное состояние информационного содержимого, необходимое для инициализации построения сервиса. Генерируется блок кода, который вызывает функцию API промежуточного ПО (платформы ИП) для выполнения запроса SPARQL (обычно запрос ASK) и проверяет результат запроса с помощью оператора if-then-else. Аналогичный процесс генерации выполняется для блока кода, представляющего результирующее условие (has_result), набор действий, выполняемых в конце вызова функции.

Свойство has_output со свойством parameter_type соответствует выходному параметру и определяет возвращаемое значение функции. С помощью языка описания схем XSD (XML-Schema) описываются необходимые типы данных (string, unsignedLong и т. д.). Элементы объектной модели могут использоваться в качестве входных и выходных параметров функций. Класс CompositeProcess по аналогии с составным процессом определяет функцию, которая вызывает внутри себя другие функции, которые описываются сущностями CompositeProcess или AtomicProcess. В этом случае вызовы внутренних функций могут быть заданы с помощью управляющих конструкций (If-Then-Else, Repeat-While, и т. д.), которые, в свою очередь, могут быть преобразованы в соответствующие выражения (конструкции) языка программирования.

Экземпляр класса Service Agents Model используется агентами КР для определения их роли в процессе построения и доставки сервисов, а также способа информационно-управляемого взаимодействия, основанного на модели "публикации/подписки". Для этого в коде каждого агента генерируется блок с необходимыми операциями подписки с использованием внутренних функций, обработчиков и функций API. Каждый запрос операции подписки задается запросом SPARQL, который может быть получен из свойства has_subprecondition, где выражение подписки задается с использованием SPARQL-запросов, которые используют классы и свойства предметной

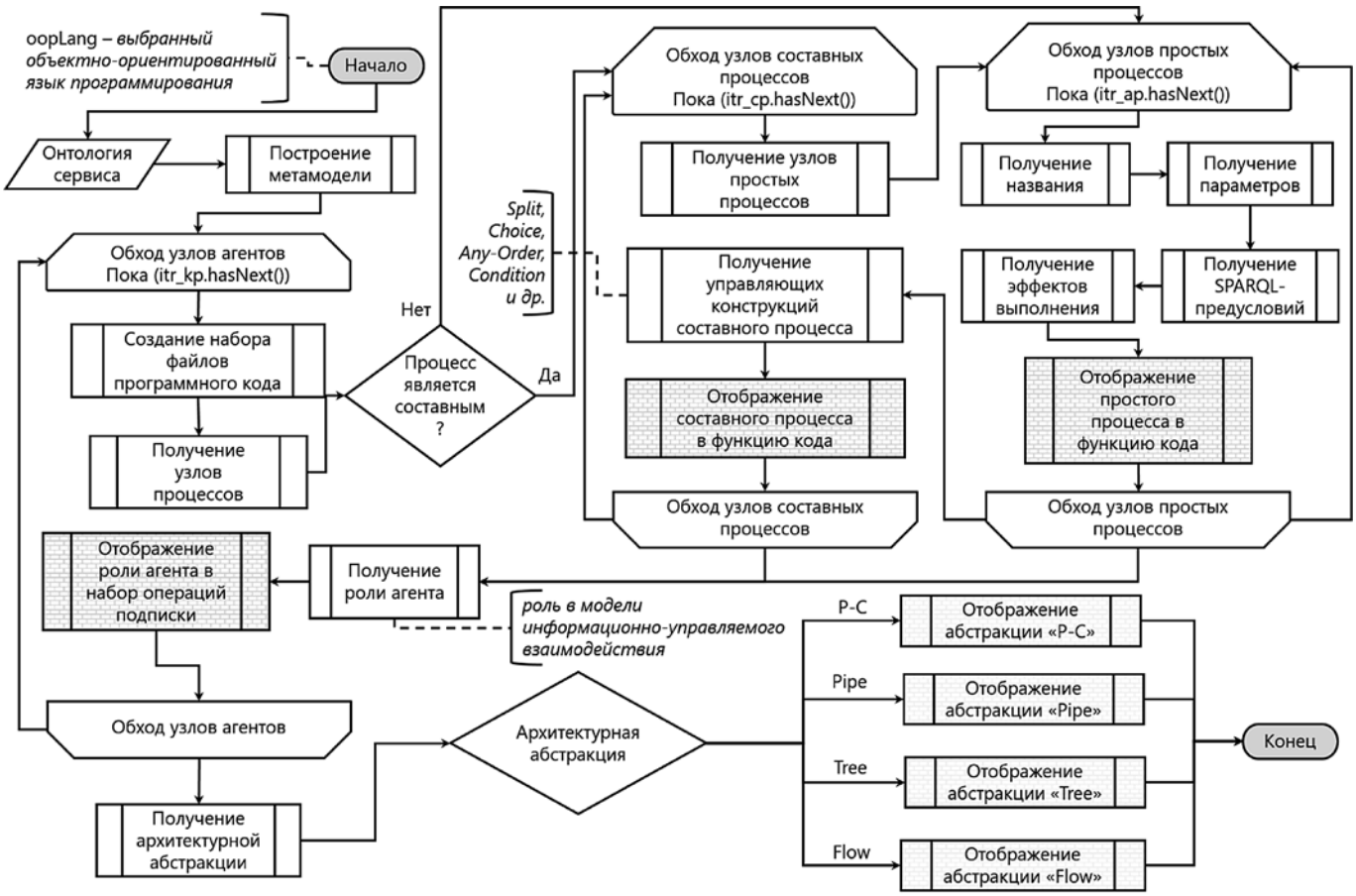

Рис. 5. Алгоритм кодогенерации взаимодействия агентов

области. Кроме этого, операции подписки могут быть определены на основе результатов выполнения процессов агентов, определенных в классах Result. Каждый такой класс определяет, какие изменения ОИС проводятся во время выполнения процессов агента КР. Во время выполнения операций агентами может возникать их самоорганизация, поскольку каждый агент КР осведомлен о своей роли в модели информационно-управляемого взаимодействия, основанной на шаблонах взаимодействия (AgentsPattern). Приведем примеры сгенерированного программного кода, полученного на основе предложенного алгоритма автоматизации программирования агентов.

Пример 1: *сервис определения присутствия и анализа активности пользователей.* Онтология спецификации сервиса S_{ua} используется для генерации кода соответствующих программных агентов КР, реализующих построение и доставку этого сервиса. На рис. 6 (см. вторую сторону обложки) представлен пример того, как индивид класса атомарного процесса SetUserPresenceLevel преобразуется в код функции на языке C++ для агента-агрегатора КР присутствия. Объекты классов (например, PresenceInfo), используемые в функции, генерируются из онтологии проблемной области (например, класс PresenceInfo). Предусловие процесса преобразуется в блок кода, реализующий выполнение и обработку SPARQL-ASK запроса на проверку наличия необходимых RDF-троек в ОИС для объекта класса PresenceInformaion (информация о присутствии пользователя).

Пример 2: *сервис совместного обогащения информационного содержимого историческими данными.* На основе разработанной онтологии для сервиса S_{en} генерируется код, участвующий в построении и доставке данного сервиса. На рис. 7 (см. вторую сторону обложки) представлен пример генерации программного кода на языке C++, реализующего составной процесс DetectHiddenRelations, для агента-контроллера КР обнаружения исторических связей. По аналогии с предыдущим примером атомарного процесса составной процесс также реализуется программной функцией, имя, входные параметры и другие атрибуты которой формируются на основе соответствующих элементов онтологии, с использованием представленных алгоритмов кодогенерации. Особенностью генерации в данном случае является преобразование управляющей конструкции Repeat-While в оператор цикла while, условие выполнения которого задается SPARQL-ASK-запросом. Составляющие атомарные процессы преобразуются в вызовы соответствующих им функций с заданными параметрами и возвращаемым значением. На основе заданного в онтологии эффекта выполнения процесса генерируется код, реализующий выполнение соответствующего SPARQL-INSERT-запроса.

В результате при генерации программного кода для сервисов S_{ua} и S_{en} на основе разработанных онтологий средняя доля сгенерированного программного кода по отношению ко всему исходному коду для

агентов составила 23,4 %, причем наибольшие результаты пришлись на блок "информационно-управляемое взаимодействие". Для объектной модели данных генератор также показал высокие показатели, средний процент покрытия соответствующего кода объектной модели сгенерированным составил 68 %.

Заключение

Настоящая работа является частью 2 (заключительной) статьи автора. В части 1 [8] были сформулированы проблемные вопросы и требования, связанные с упрощением процессов разработки и сопровождения ИП-приложений за счет создания средств автоматизации процессов программирования агентов при построении семантических сервисов. В настоящей работе предложены решения, направленные на достижение выделенных ранее требований на пути к созданию генератора программного кода агентов для объектно-ориентированных языков программирования на основе онтологий сервисов. За счет расширения онтологии OWL-S введено унифицированное онтологическое описание семантики процессов построения сервисов, позволяющее наделить их качествами семантических сервисов и сделать их пригодными для автоматизации программирования. Предложены алгоритмы автоматизации программирования агентов для реализации структур объектной модели данных и взаимодействия агентов при построении сервисов. Полученные результаты рассмотрены на представленных в части 1 сервисах для интеллектуальных окружений: сервис определения и анализа активности пользователей (интеллектуальный зал); сервис совместного обогащения информационного содержимого историческими данными (умный музей).

Исследование поддержано Минобрнауки России в рамках инициативного проекта № 2.5124.2017/8.9 базовой части государственного задания на 2017—2019 гг. Научные результаты получены при финансовой поддержке РФФИ, проект № 19-07-01027. Статья подготовлена в рамках реализации Программы развития опорного университета для Петрозаводского государственного университета на 2017—2021 гг.

Список литературы

1. **Qin Y., Sheng Q. Z., Falkner N. J.** et al. When things matter: A survey on data-centric internet of things // Journal of Network and Computer Applications. 2016. Vol. 64. P. 137—153.

2. **Dohr A., Modre-Opsrian R., Drobics M.** et al. The internet of things for ambient assisted living // 2010 Seventh International Conference on information technology: new generations. IEEE, 2010. P. 804—809.

3. **Korzun D. G., Balandin S. I., Kashevnik A. M.** et al. Smart spaces-based application development: M3 architecture, design principles, use cases, and evaluation // International Journal of Embedded and Real-Time Communication Systems (IJERTCS). 2017. Vol. 8, N. 2. P. 66—100.

4. **Honkola J., Laine H., Brown R., Tyrkkö O.** Smart-M3 information sharing platform // Proc. IEEE Symp. Computers and Communications, Riccione, Italy, 22—25 June 2010. Washington: IEEE Computer Society, 2010. P. 1041—1046.

5. **Viola F., D'Elia A., Korzun D.** et al. The M3 architecture for smart spaces: Overview of semantic information broker implementations // 2016 19th Conference of Open Innovations Association (FRUCT). IEEE, 2016. P. 264—272.

6. **Корзун Д. Ж.** Формализм сервисов и архитектурные абстракции для программных приложений интеллектуальных пространств // Программная инженерия. 2015. № 2. С. 3—12.

7. **Jara A. J., Olivieri A. C., Bocchi Y.** et al. Semantic web of things: an analysis of the application semantics for the iot moving towards the iot convergence // International Journal of Web and Grid Services. 2014. Vol. 10, N. 2—3. P. 244—272.

8. **Марченков С. А.** Автоматизация процессов программирования агентов на основе кодогенерации при построении семантических сервисов интеллектуальных пространств. Часть 1 // Программная инженерия. 2019. Т. 10, № 6. С. 257—264.

9. **Ломов А. А., Корзун Д. Ж.** Операция подписки для приложений в интеллектуальных пространствах платформы Smart-M3 // Труды СПИИРАН. 2012. Т. 4, № 23. С. 439—458.

10. **Martin D., Burstein M., Mcdermott D.** et al. Bringing semantics to web services with OWL-S // World Wide Web. 2007. Vol. 10, N. 3. P. 243—277.

11. **OWL-S:** Semantic Markup for Web Services. URL: https://www.w3.org/Submission/OWL-S (дата обращения: 04.09.2019).

12. **Sirin E., Parsia B., Grau B. C.** et al. Pellet: A practical owl-dl reasoner // Web Semantics: science, services and agents on the World Wide Web. 2007. Vol. 5, N. 2. P. 51—53.

13. **FOAF** Vocabulary Specification 0.99. URL: http://xmlns.com/foaf/spec (дата обращения 04.09.2019).

14. **Марченков С. А., Корзун Д. Ж.** Определение присутствия пользователей в интеллектуальном зале на основе отслеживания активности в беспроводной сети // Ученые записки Петрозаводского государственного университета. 2015. № 2. С. 114—119.

15. **Korzun D. G., Marchenkov S. A., Vdovenko A. S., Petrina O. B.** A semantic approach to designing information services for smart museums // International Journal of Embedded and Real-Time Communication Systems (IJERTCS). 2016. Vol. 7, N. 2. P. 15—34.

16. **Marchenkov S. A., Baganov D. E., Korzun D. G.** Smart-M3 CuteSIB Demo for a Wireless Router with OpenWrt-Based Firmware // Proc. 20th Conf. of Open Innovations Association FRUCT. 2017. P. 634—638.

17. **Batanov D. N., Vongdoiwang W.** Using ontologies to create object model for object-oriented software engineering // Ontologies. Springer, Boston, MA, 2007. P. 461—487.

18. **Siricharoen W. V.** Ontologies and object models in object oriented software engineering // IAENG International Journal of Computer Science. 2007. Vol. 33, N. 1. P. 19—24.

19. **Evermann J., Wand Y.** Ontology based object-oriented domain modelling: fundamental concepts // Requirements engineering. 2005. Vol. 10, N. 2. P. 146—160.

20. **Kulakov K., Marchenkov S., Tishkov S.** An Approach to Generating Ontology-Based Object Model for Smart-M3 platform // Proceedings of the 24th Conference of Open Innovations Association FRUCT. FRUCT Oy, 2019. P. 670—676.

Computer-Aided Programming of Software Agents Based on Code Generation in Constructing Semantic Services of Smart Spaces. Part 2

S. A. Marchenkov, marchenk@cs.petrsu.ru, Petrozavodsk State University, Petrozavodsk, 185910, Russian Federation

Corresponding author:

Marchenkov Sergei A., Junior Researcher, Petrozavodsk State University, Petrozavodsk, 185910, Russian Federation
E-mail: marchenk@cs.petrsu.ru

Received on September 05, 2019
Accepted on September 26, 2019

The work is the final second part of the paper, published with the same title. In the first part, problematic questions and requirements were formulated related to the simplification of the development and maintenance processes for smart space applications by creating tools for automating agent programming constructing in the construction of semantic services. This paper proposes solutions aimed at achieving the previously identified requirements on the way to creating an agent code generator for object-oriented programming languages based on service ontologies. By expanding the OWL-S ontology, a unified ontological description of the semantics for service construction processes is introduced. This ontology allows services to be endowed with the qualities of semantic services and makes them suitable for programming automation. Algorithms for automating agent programming to implement the structures of an object data model and agent interactions when programming services based on the generation of program code using a service ontology are proposed. The results are considered on the services presented in the previous part for smart environments: the presence detection and user activity service (smart room), the historical enrichment service (smart museum).

Keywords: *smart spaces, semantic services, ontological model, object model, code generation*

Acknowledgements: *The research was financially supported by the Ministry of Education and Science of Russia within project #2.5124.2017/8.9 of the basic part of state research assignment for 2017—2019. The reported study was funded by RFBR according to the research project # 19-07-01027. The article was prepared within the Government Program of Flagship University Development for Petrozavodsk State University in 2017—2021.*

For citation:

Marchenkov S. A. Computer-Aided Programming of Software Agents Based on Code Generation in Constructing Semantic Services of Smart Spaces. Part 2, *Programmnaya Ingeneria,* 2019, vol. 10, no. 11—12, pp. 440—450.

DOI: 10.17587/prin.10.440-450

References

1. **Qin Y., Sheng Q. Z., Falkner N. J., Dustdar S., Wang H., Vasilakos A. V.** When things matter: A survey on data-centric internet of things, *Journal of Network and Computer Applications*, 2016, vol. 64, pp. 137—153.

2. **Dohr A., Modre-Opsrian R., Drobics M., Hayn D., Schreier G.** The internet of things for ambient assisted living, *2010 seventh international conference on information technology: new generations*, IEEE, 2010, pp. 804—809.

3. **Korzun D. G., Balandin S. I., Kashevnik A. M., Smirnov A. V., Gurtov A. V.** Smart spaces-based application development: M3 architecture, design principles, use cases, and evaluation, *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, 2017, vol. 8, no. 2, pp. 66—100.

4. **Honkola J., Laine H., Brown R., Tyrkkö O.** Smart-M3 information sharing platform, *Proc. IEEE Symp. Computers and Communications*, Riccione, Italy, 22—25 June 2010, Washington: IEEE Computer Society, 2010, pp. 1041—1046.

5. **Viola F., D'Elia A., Korzun D., Galov I., Kashevnik A., Balandin S.** The M3 architecture for smart spaces: Overview of semantic information broker implementations, *2016 19th Conference of Open Innovations Association (FRUCT)*, IEEE, 2016, pp. 264—272.

6. **Korzun D. G.** Service Formalism and Architectural Abstractions for Smart Space Applications, *Programmnaya ingeneria*, 2015, no. 2, pp. 3—12 (in Russian).

7. **Jara A. J., Olivieri A. C., Bocchi Y., Jung M., Kastner W., Skarmeta A. F.** Semantic web of things: an analysis of the application semantics for the iot moving towards the iot convergence, *International Journal of Web and Grid Services*, 2014, vol. 10, no. 2—3, pp. 244—272.

8. **Marchenkov S. A.** Computer-aided programming of software agents based on code generation in constructing semantic services of smart spaces. Part 1, *Programmnaya ingeneria*, 2019, vol. 10, no. 6, pp. 257—264 (in Russian).

9. **Lomov A. A., Korzun D.** Subscription operation for applications in smart spaces of Smart-M3 platform, *Trudy SPIIRAN*, 2012, vol. 4, no. 23, pp. 439—458 (in Russian).

10. **Martin D., Burstein M., Mcdermott D., Mcilraith S., Paolucci M., Sycara K., McGuinness D. L., Sirin E., Srinivasan N.** Bringing semantics to web services with OWL-S, *World Wide Web*, 2007, vol. 10, no. 3, pp. 243—277.

11. **OWL-S:** Semantic Markup for Web Services, available at: https://www.w3.org/Submission/OWL-S (accessed 04.09.2019).

12. **Sirin E., Parsia B., Grau B. C., Kalyanpur A., Katz Y.** Pellet: A practical owl-dl reasoner, *Web Semantics: science, services and agents on the World Wide Web*, 2007, vol. 5, no. 2, pp. 51—53.

13. **FOAF** Vocabulary Specification 0.99, available at: http://xmlns.com/foaf/spec (accessed 04.09.2019).

14. **Marchenkov S. A., Korzun D. Zh.** Network activity tracking detection of user presence in smart room, *Uchenye zapiski Petrozavodskogo gosudarstvennogo universiteta*, 2015, no. 2, pp. 114—119 (in Russian).

15. **Korzun D. G., Marchenkov S. A., Vdovenko A. S., Petrina O. B.** A semantic approach to designing information services for smart museums, *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, 2016, vol. 7, no. 2, pp. 15—34.

16. **Marchenkov S. A., Baganov D. E., Korzun D. G.** Smart-M3 CuteSIB Demo for a Wireless Router with OpenWrt-Based Firmware, *Proc. 20th Conf. of Open Innovations Association FRUCT*, 2017, pp. 634—638.

17. **Batanov D. N., Vongdoiwang W.** Using ontologies to create object model for object-oriented software engineering, *Ontologies*, Springer, Boston, MA, 2007, pp. 461—487.

18. **Siricharoen W. V.** Ontologies and object models in object oriented software engineering, *IAENG International Journal of Computer Science*, 2007, vol. 33, no. 1, pp. 19—24.

19. **Evermann J., Wand Y.** Ontology based object-oriented domain modelling: fundamental concepts, *Requirements engineering*, 2005, vol. 10, no. 2, pp. 146—160.

20. **Kulakov K., Marchenkov S., Tishkov S.** An Approach to Generating Ontology-Based Object Model for Smart-M3 platform, *Proceedings of the 24th Conference of Open Innovations Association FRUCT*, FRUCT Oy, 2019, pp. 670—676.

УДК 004.023 DOI: 10.17587/prin.10.451-456

С. И. Родзин, канд. техн. наук, проф., srodzin@sfedu.ru, **О. Н. Родзина,** ст. препод., orodzina@sfedu.ru, Южный федеральный университет, Таганрог

Сравнение программных реализаций эволюционных вычислений для задач многомерной оптимизации*

Представлен эволюционный алгоритм, способный решать многомерные оптимизационные задачи с использованием иерархического мультипопуляционного подхода. Используются специальные операторы для поддержки разнообразия популяции решений, расширения области поиска решений за счет менее перспективных решений. Оценка эффективности предложенного алгоритма проводится на наборе многомерных функций Гриванка, Растригина, Розенброка, Швефеля. Показатели разработанного алгоритма сравнивают с показателями конкурирующих алгоритмов. Статистически значимые различия свидетельствуют в пользу масштабируемого эволюционного алгоритма для всех рассмотренных функций при возрастании размерности задачи.

Ключевые слова: *большие данные; масштабируемость; эволюционный алгоритм; многомерные задачи; функция Гриванка; функция Растригина; функция Розенброка; функция Швефеля*

Введение

Эволюционные вычисления представляют собой математические преобразования, позволяющие трансформировать входной поток информации в выходной поток по правилам, основанным на имитации механизмов эволюции, а также на статистическом подходе к исследованию ситуаций и итерационном приближении к искомому решению [1]. Эволюционные алгоритмы исследуют пространство поиска, синтезируя решения, являющиеся точками этого пространства, и запрашивая оценку их качества, или "приспособленность", которая используется для осуществления "естественного отбора" популяции решений. Тем самым эволюционные алгоритмы обучаются тому, какие области пространства поиска содержат наилучшие решения. В этом смысле эволюционные алгоритмы являются одной из ветвей машинного обучения.

При решении задач многомерной оптимизации, а также при анализе больших данных и в машинном обучении, где объем обучающих данных может быть очень велик, особенно важным является свойство масштабируемости алгоритмов, используемых при решении этих задач. Масштабируемость алгоритма предполагает, что прямо пропорционально увеличению объема обрабатываемых данных растут его вычислительные затраты и способность при этом выдать наилучшее по его настоящим возможностям решение в любое время вычисления, даже если процесс вычислений не завершен естественным остановом [2]. Масштабируемость также предполагает возможность проводить вычисления в пределах ограниченного объема памяти используемого компьютера.

Эволюционные алгоритмы недостаточно хорошо масштабируются. Особенно остро этот проблемный вопрос стоит при решении многомерных оптимизационных задач и задач обработки больших объемов данных. Актуальной является задача разработки масштабируемого эволюционного алгоритма, способного поддерживать разнообразие популяции решений и находить баланс между скоростью сходимости алгоритма и диверсификацией поиска в пространстве решений.

Многомерные задачи и эволюционные алгоритмы

Интенсивный рост объемов данных, доступных для использования при решении задач в таких областях, как биомедицина, инженерия, финансы и социальные науки, является результатом технологической революции. Источниками больших данных выступают данные с измерительных устройств, события от радиочастотных идентификаторов, потоки сообщений из социальных сетей, метеорологические данные, данные дистанционного зондирования Земли, потоки данных о местонахождении абонентов сетей сотовой связи, от устройств аудио- и видеорегистрации. Однако при применении традиционных технологий обработки больших данных при десятках тысяч объектов, извлеченных из документов и изображений, возникают известные пространственно-временные проблемы. Разреженность данных — еще одна особенность, связанная с методами анализа больших данных.

Как влияет рост размерности подлежащих анализу данных в задачах оптимизации на вычислительную производительность, вычислительные затраты, устойчивость работы эволюционных алгоритмов? Какие из существующих подходов используются

* Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-07-00570-а.

в эволюционных алгоритмах для получения ответов на такие вопросы?

В существующих эволюционных алгоритмах среднее значение функций приспособленности особей в популяции постепенно увеличивается, популяция теряет разнообразие, а поиск сходится к глобальному или локальному оптимуму. Поэтому в некоторых работах предлагается увеличить разнообразие популяций, генерируемых эволюционным алгоритмом [3]. Однако этого недостаточно. Более веская причина сходимости к субоптимальным решениям заключается в увеличении среднего значения функции приспособленности при генерации новых популяций решений. Даже решения, которые во многом определяются особями в популяции, с конкурентным значением функции приспособленности, но расположенные в отдаленных окрестностях поискового пространства, вследствие их редкого появления могут исчезнуть из популяции в результате дрейфа. Эта проблема еще более значима в случае многомерных задач с многоэкстремальными функциями в силу огромного объема пространства поиска [3].

В других работах [4] предложено контролировать число особей в локальных поисковых пространствах, избегая избыточности и одновременно контролируя среднее значение функции приспособленности в популяции особей. В работе [5] предложено генерировать несколько субпопуляций, контролируя миграцию особей из одной субпопуляции в другую. В работе [6] для решения проблемы масштабируемости предложено использовать метод дифференциальной эволюции, в работе [7] — метод кооперативной коэволюции. Однако эти подходы и предлагаемые в них эвристические алгоритмы с точки зрения масштабируемости требуют использования нереалистично больших популяций или субпопуляций, чтобы эффективно справляться с "проклятием размерности". Например, в работе [8] использовали популяцию из 350 000 особей. При этом увеличение размеров популяции не гарантировало улучшение результатов.

Коэволюционный подход к сокращению размеров популяции в генетическом алгоритме был предложен в работе [9]. Его идея состоит в декомпозиции многомерной задачи на низкоразмерные задачи с последующей их оптимизацией и комбинированием. Однако применение данного подхода к несепарабельным задачам оказалось затруднительным.

В настоящей работе предложен более простой подход к решению проблемы высокой размерности при разумной поддержке разнообразия популяции. Известно, что биоинспирированные алгоритмы имеют тенденции, во-первых, сходиться к локальному, а не к глобальному оптимуму для многомерных задач, во-вторых, с возрастанием размерности задачи увеличивать вероятность мутационных разрушений найденных наилучших решений, в-третьих, к потере разнообразия популяции решений. При этом в существующих на данный момент исследованиях недостаточное освещение получили вопросы, как бороться с преждевременной сходимостью биоинспирированных алгоритмов, поддерживать разнообразие популяции и находить баланс между скоростью сходимости алгоритма и диверсификацией поиска решений. Для решения указанных вопросов предлагается вначале определять тенденцию к локальной сходимости биоинспирированного алгоритма, а затем поддерживать разнообразие популяции путем замены избыточных особей из основной популяции на особи из субпопуляций из неисследованных областей пространства решений. При этом с увеличением размерности задачи не происходит существенного роста размеров популяции и поддерживается ее разнообразие.

Эволюционный масштабируемый алгоритм для решения многомерных оптимизационных задач

Идея предлагаемого эволюционного масштабируемого алгоритма (ЭМА) заключается в объединении преимуществ иерархической структуры популяции и когнитивного оператора мутации для поддержки разнообразия популяции а также в использовании перспективных областей поискового пространства.

Далее приведен псевдокод алгоритма ЭМА.

Procedure ЭМА
1: ***begin***
2: $t = 0$
3: Инициализация популяции $P(t)$
4: Оценка особей популяции $P(t)$
5: ***while*** (проверка условия останова)
6: ***begin***
7: $t := t + 1$
8: ***call procedure*** (построение иерархической структуры популяций)
9: ***call procedure*** (выполнение когнитивного оператора мутации)
10: Создание новой популяции с помощью механизма поддержки разнообразия популяции и когнитивного оператора мутации
11: Оценка особей популяции $P(t)$
14: ***end while***
15: ***for*** (каждая особь в популяции) ***do***
16: ***begin***
17: Миграция особей из основной популяции
18: ***end for***
19: ***for*** (каждая из n предварительно отобранных субпопуляций)
20: ***begin***
21: Популяция эволюционирует заданное количество раз
22: Миграция особей
23: ***end***
24: ***end***

Отличительными особенностями ЭМА являются построение иерархических субпопуляций и выполнение когнитивного оператора мутации. Иерархия популяций включает основную популяцию особей-решений и множество субпопуляций с менее подходящими особями.

Далее приведен псевдокод процедуры построения иерархических субпопуляций.

1: ***begin***
/*Создание иерархической структуры популяций, включающих основную популяцию и множество суб-

популяций, число которых определяется примерно равным 7 % от размерности задачи *n**/

2: Определение плотности ρ основной популяции и субпопуляций в иерархии: $\rho_{Si} = N_{Si}/N_{POP}$, где N_{Si} — число особей в субпопуляции S_i; N_{POP} — общий размер популяции. Установление порогового значения α плотности ρ основной популяции и субпопуляций.

3: ***for*** (для каждой субпопуляции S_i иерархии популяций)

4: ***begin***

5: ***if*** ($\rho_{Si} > \alpha$) ***then*** (определение центра CS_i субпопуляции S_i)

7: ***end for***

/*поиск подмножества особей для перемещения в субпопуляцию S_i*/

8: ***for*** (для каждой субпопуляции S_i иерархии популяций)

9: ***begin***

10: ***while*** (число особей в субпопуляции S_i меньше порогового значения)

11: ***begin***

12: Поиск особей с функцией приспособленности, близкой к центру CS_i субпопуляции S_i

13: ***end***

14: Перемещение найденных особей в субпопуляцию S_i

15: ***end while***

16: ***end for***

17: ***end***

Согласно этой процедуре в процессе поиска оптимального решения отдельные наименее приспособленные особи перемещаются из основной популяции в иерархические субпопуляции, эволюционируют в них и имеют определенные шансы вернуться в основную популяцию. В многоуровневой иерархической структуре множество субпопуляций существует наряду с основной популяцией, а особи перемещаются между ними согласно их функциям приспособленности.

Псевдокод процедуры выполнения когнитивного оператора мутации представлен ниже.

1: ***begin***

2: Ввод списка иерархических субпопуляций S_i

3: ***for*** (для основной популяции иерархии популяций)

4: ***begin***

5: ***if*** (стандартное отклонение функции приспособленности для основной популяции S_i не больше значения стандартного отклонения функции приспособленности для всех субпопуляций) *AND* (плотность ρ_{Si} основной популяции не меньше плотности всех субпопуляций) ***then***

6: ***begin***

7: Случайный выбор *N* особей из других субпопуляций S_i и замена ими особей с наихудшей приспособленностью из основной популяции

8: ***end if***

9: ***else if*** (среднее значение функций приспособленности в основной популяции больше среднего значения функций приспособленности всех субпопуляций S_i) ***then***

10: ***begin***

11: Определение лучшей особи в субпопуляциях S_i, которая передается в основную популяцию

12: ***end else if***

13: **end for**

14: Применение стандартных операторов мутации и кроссинговера ко всей популяции $P(t)$

15: ***end***

Когнитивный оператор мутации поддерживает разнообразие популяции и применяется только к особям основной популяции. Решение о применении когнитивного оператора мутации принимается, если наблюдается сходимость к локальному оптимуму. Наименее приспособленные особи из основной популяции заменяются на перспективные особи из субпопуляций.

Результаты тестирования программной реализации эволюционного масштабируемого алгоритма на многомерных функциях-бенчмарках

Чтобы продемонстрировать эффективность, а также вычислительные характеристики предложенного алгоритма, создана программная среда на языке программирования C#. Отладку и тестирование проводили на ЭВМ типа IBM PC с процессором Core i7 с ОЗУ 8 Гбайт.

Для тестирования производительности ЭМА использовали набор многомерных (размерность от 20 до 1000) оптимизационных функций-бенчмарков.

Вначале была оценена эффективность ЭМА для задач размерности $n = 20$ переменных. Далее алгоритм тестировали на функциях размерности $n = 50$, $n = 100$ и т. д. переменных. Показатели ЭМА сравнивали с показателями тестирования программных реализаций следующих конкурирующих алгоритмов: стандартного эволюционного алгоритма (*SEA*); самоорганизующегося эволюционного алгоритма (*SOCEA*); клеточного эволюционного алгоритма (*CEA*); эволюционного алгоритма с управляемым разнообразием популяции (*DGEA*). Подробное описание этих алгоритмов представлено в работах [10—14].

В качестве тестовых функций использовали следующие многомерные функции-бенчмарки [15]:

- Гриванка

$$F_{gri}(\mathbf{X}) = \frac{1}{40000}\sum_{i=1}^{n}(x_i - 100)^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i - 100}{\sqrt{i}}\right) + 1;$$

- Растригина

$$F_{rtg}(\mathbf{X}) = \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right];$$

- Розенброка

$$F_{ros}(\mathbf{X}) = \sum_{i=1}^{n-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right];$$

- Швефеля

$$F_{sch}(\mathbf{X}) = 418,9829n - \sum_{i=1}^{n} x_i \sin\left(\sqrt{|x_i|}\right.$$

Здесь n — число переменных, от которых зависит функция.

Бенчмарки выбирали по следующим принципам:

- оптимизируемые функции должны быть непохожими друг на друга;
- функции должны вызывать затруднения у известных точных методов оптимизации;
- функции должны быть нелинейными, несепарабельными, масштабируемыми.

Все отмеченные выше функции удовлетворяют этим принципам. В качестве иллюстрации приведен трехмерный вид этих функций для $n = 2$ (рис. 1—4, см. третью сторону обложки).

Глобальный минимум этих функций: $F_{gri}(X = 0) = 0$, $F_{rtg}(X = 0) = 0$, $F_{ros}(X = 1) = 0$, $F_{sch}(X = 0) = 0$.

В экспериментах использовали следующие настройки: размер основной популяции $N = 250$, вероятность мутации $p_m = 0{,}05$, вероятность кроссинго-

Таблица 1

Значения ошибок на тестовых функциях для алгоритма ЭМА

n	Значение	Функция			
		$F_{gri}(\mathbf{X})$	$F_{rtg}(\mathbf{X})$	$F_{ros}(\mathbf{X})$	$F_{sch}(\mathbf{X})$
20	Лучшее	$4 \cdot 10^{-62}$	$1{,}01 \cdot 10^{-61}$	$0{,}5 \cdot 10^{-60}$	$1{,}05 \cdot 10^{-62}$
	Среднее	$4{,}91 \cdot 10^{-62}$	$1{,}95 \cdot 10^{-61}$	$1{,}11 \cdot 10^{-60}$	$1{,}15 \cdot 10^{-50}$
	Худшее	$8{,}11 \cdot 10^{-62}$	$3{,}00 \cdot 10^{-61}$	$1{,}92 \cdot 10^{-60}$	$2{,}29 \cdot 10^{-50}$
50	Лучшее	$4{,}73 \cdot 10^{-40}$	$1{,}1 \cdot 10^{-40}$	$1{,}11 \cdot 10^{-40}$	$1{,}03 \cdot 10^{-30}$
	Среднее	$4{,}81 \cdot 10^{-40}$	$1{,}1 \cdot 10^{-40}$	$1{,}14 \cdot 10^{-40}$	$1{,}04 \cdot 10^{-30}$
	Худшее	$5{,}1 \cdot 10^{-40}$	$1{,}3 \cdot 10^{-40}$	$1{,}2 \cdot 10^{-40}$	$1{,}19 \cdot 10^{-30}$
100	Лучшее	$8{,}92 \cdot 10^{-21}$	$1{,}21 \cdot 10^{-20}$	$1{,}2 \cdot 10^{-20}$	$1{,}09 \cdot 10^{-15}$
	Среднее	$8{,}93 \cdot 10^{-21}$	$1{,}21 \cdot 10^{-20}$	$1{,}2 \cdot 10^{-20}$	$1{,}09 \cdot 10^{-15}$
	Худшее	$8{,}95 \cdot 10^{-21}$	$1{,}22 \cdot 10^{-20}$	$1{,}21 \cdot 10^{-20}$	$1{,}68 \cdot 10^{-15}$

Таблица 2

Результаты сравнения алгоритмов SEA, SOCEA, CEA, DGEA и ЭМА

n	Функция	Алгоритм				
		SEA	SOCEA	CEA	DGEA	ЭМА
20	$F_{gri}(\mathbf{X})$	1,171	0,930	0,642	$7{,}88 \cdot 10^{-8}$	$4 \cdot 10^{-6}2$
	$F_{rtg}(\mathbf{X})$	11,12	2,875	1,250	$3{,}37 \cdot 10^{-8}$	$1{,}01 \cdot 10^{-61}$
	$F_{ros}(\mathbf{X})$	8292,32	406,490	149,056	8,127	$0{,}5 \cdot 10^{-60}$
	$F_{sch}(\mathbf{X})$	—	—	—	—	$1{,}5 \cdot 10^{-50}$
50	$F_{gri}(\mathbf{X})$	1,616	1,147	1,032	$1{,}19 \cdot 10^{-3}$	$1{,}11 \cdot 10^{-40}$
	$F_{rtg}(\mathbf{X})$	44,674	22,460	14,224	$1{,}97 \cdot 10^{-6}$	10^{-40}
	$F_{ros}(\mathbf{X})$	41425,7	4783,25	1160,08	59,789	10^{-40}
	$F_{sch}(\mathbf{X})$	—	—	—	—	$1{,}9 \cdot 10^{-30}$
100	$F_{gri}(\mathbf{X})$	2,250	1,629	1,179	$3{,}24 \cdot 10^{-3}$	$1{,}01 \cdot 10^{-21}$
	$F_{rtg}(\mathbf{X})$	106,212	86,364	58,380	$6{,}56 \cdot 10^{-5}$	$1{,}01 \cdot 10^{-20}$
	$F_{ros}(\mathbf{X})$	91251,3	30427,6	6053,87	880,324	10^{-20}
	$F_{sch}(\mathbf{X})$	—	—	—	—	10^{-15}

Примечание. Прочерк означает, что соответствующие данные отсутствуют

вера p_c = 0,9. Полученные результаты усреднялись по 30 независимым прогонам. Максимальное число поколений в каждом прогоне составляет 500, 1000 и 2000 для числа переменных n = 20, 50 и 100.

Эмпирические результаты, полученные алгоритмом ЭМА на четырех функциях-бенчмарках, представлены в табл. 1. В таблице указаны значения ошибок ($F(X)$ — $F(X)$*), где $F(X)$* — значение глобального минимума функции. Каждый столбец соответствует функциям Гриванка, Растригина, Розенброка и Швефеля.

Значения ошибок за 30 прогонов в порядке возрастания (лучшее, среднее, худшее) представлены для n = 20, 50, 100. Видно, что алгоритм ЭМА показывает устойчивые результаты при разной размерности в различных прогонах моделирования. Это показатель надежности работы алгоритма.

В табл. 2 представлены лучшие результаты (полученное минимальное значение функции) конкурирующих алгоритмов *SEA*, *SOCEA*, *CEA*, *DGEA* и алгоритма ЭМА.

Результаты свидетельствуют в пользу алгоритма ЭМА.

С помощью t-критерия Стьюдента (уровень значимости 0,05, достоверность 95 %) проведена проверка того, являются ли различия в результатах (значения оптимизируемой функции) для предлагаемого алгоритма ЭМА статистически значимыми по сравнению с конкурирующими алгоритмами.

Сравнение показало, что рассчитанные значения t-критерия превышают соответствующие критические значения, отмеченные в специальных таблицах. Следовательно, наблюдаемые различия являются статистически значимыми. Статистически значимые различия свидетельствуют в пользу алгоритма ЭМА для всех рассмотренных функций-бенчмарков, особенно с возрастанием размерности задачи.

Заключение

Разработан эволюционный масштабируемый алгоритм, использующий иерархический мультипопуляционный подход и специальные операторы для поддержки разнообразия популяции решений, расширения области поиска решений за счет менее перспективных решений. Этот алгоритм является достаточно перспективным при решении задач многомерной оптимизации со сложными мультимодальными пространствами решений и большими данными. Оценка эффективности предложенного алгоритма проводилась на наборе многомерных оптимизационных функций-бенчмарков Гриванка, Растригина, Розенброка, Швефеля. Показатели разработанного алгоритма сравнивали с показателями четырех конкурирующих алгоритмов, а именно — стандартного эволюционного алгоритма, саморганизующегося эволюционного алгоритма, клеточного эволюционного алгоритма, эволюционного алгоритма с управляемым разнообразием популяции. Эксперименты свидетельствуют в пользу алгоритма ЭМА. При этом различия в значениях оптимизируемых функций для алгоритма ЭМА являются статистически значимыми по сравнению с конкурирующими алгоритмами, особенно с возрастанием размерности задачи. На взгляд авторов, это объясняется возможностями ЭМА поддерживать разнообразие популяции и находить баланс между скоростью сходимости алгоритма и диверсификацией поиска.

Список литературы

1. **Родзин С. И., Курейчик В. В.** Состояние, проблемы и перспективы развития биоэвристик // Программные системы и вычислительные методы. 2016. № 2. С. 158—172.

2. **Курейчик В. В., Курейчик В. М., Родзин С. И.** Теория эволюционных вычислений. М.: Физматлит, 2012. 260 с.

3. **Črepinšek M., Liu S.-H., Mernik M.** Exploration and exploitation in evolutionary algorithms: a survey // ACM Computing Surveys. 2013. Vol. 45, N. 3. P. 35—44.

4. **Virginia Y., Analía A.** A deterministic crowding evolutionary algorithm to form learning teams in a collaborative learning context // Expert System Appl. 2012. Vol. 39, N. 10. P. 8584—8592.

5. **Jie Y., Nawwaf K., Grogono P.** Bi-objective multi population genetic algorithm for multimodal function optimization // IEEE Trans. Evol. Comput. 2010. Vol. 14, N. 1. P. 80—102.

6. **Hui W., Zhijian W., Shahryar R.** Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems // Soft Computing. 2011. Vol. 15, N. 11. P. 2127—2140.

7. **Xiaodong L., Yao X.** Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms // Proc. of the IEEE congress on evolutionary computation (CEC'09). 2009. P. 1546—1553.

8. **Koza J., Andre M, Bennet F., Keane M.** Genetic programming III: Darwinian invention and problem solving. Cambridge. MA: MIT Press, 1999. 253 p.

9. **García-Pedrajas N., Castillo J., Ortiz-Boyer D.** A cooperative coevolutionary algorithm for instance selection for instance-based learning // Machine Learning. 2010. Vol. 78, N. 3. P. 381—420.

10. **Родзин С. И., Курейчик В. В.** Теоретические вопросы и современные проблемы развития когнитивных биоинспирированных алгоритмов оптимизации // Кибернетика и программирование. 2017. № 3. С. 51—79.

11. **Bhattacharya M.** Exploiting landscape information to avoid premature convergence in evolutionary search // Proc. of the 2006 IEEE congress on evolutionary computation (CEC2006). IEEE Press, 2006. P. 2575—2579.

12. **Ursem R. K.** Diversity-guided evolutionary algorithms // Proceedings of parallel problem solving from nature VII (PPSN-2002), 2002. P. 462—71.

13. **Aranha C., Iba H.** Modelling cost into a genetic algorithm-based portfolio optimization system by seeding and objective sharing // Proc. of the IEEE congress on evolutionary computation, CEC2007, 2007. P. 196—203.

14. **Aranha C, Iba H.** The memetic tree-based genetic algorithm and its application to portfolio optimization // Memeting Computer. 2009. N. 1. P. 139—51.

15. **Сергиенко А. Б.** Тестовые функции для глобальной оптимизации. Красноярск: Изд-во СГАУ, 2015. 112 с.

Multidimensional Optimization Problems: Comparison among Software Implementations of Evolutionary Computations

S. I. Rodzin, srodzin@sfedu.ru, **O. N. Rodzina**, orodzina@sfedu.ru, Southern Federal University, Taganrog, 347900, Russian Federation

Corresponding author:

Rodzin Sergey I., Professor, Southern Federal University, Taganrog, 347900, Russian Federation
E-mail: srodzin@sfedu.ru

Received on July 21, 2019
Accepted on September 25, 2019

Evolutionary algorithms have particular advantages over traditional deterministic optimization methods. For example, class of algorithms better explores the solution space. Moreover, big data and multidimensional problems add additional complexity when we want to explore the solution search space. The paper presents an evolutionary algorithm capable of solving multidimensional optimization problems of high dimension. The algorithm uses a hierarchical multi-population approach, as well as special operators to support the diversity of the decision population and to expand the area of finding solutions with less promising solutions. The efficiency of the proposed algorithm is evaluated on a set of multidimensional optimization functions-benchmarks of Griewank, Rastrigin, Rosenbrock, and Schwefel. Moreover, the indicators of the developed algorithm are compared with the indicators of competing algorithms. We can note statistically significant differences. In its turn, this fact confirms the advantages of a scalable evolutionary algorithm for all considered benchmark functions especially with an increasing dimension of the problem. The authors believe that this is due to the ability of the algorithm to maintain the diversity of the population and to find a balance between the rate of convergence of the algorithm and the diversification of the search. To sum up, the algorithm is a quite promising way for solving multidimensional optimization problems with complex multimodal solution spaces and big data.

Keywords: *big data, scalability, evolutionary algorithm, multidimensional problems, Griewank function, Rastrigin function, Rosenbrockf unction, Schwefel function*

For citation:

Rodzin S. I., Rodzina O. N. Multidimensional Optimization Problems: Comparison among Software Implementations of Evolutionary Computations, *Programmnaya Ingeneria*, 2019, vol. 10, no. 11—12, pp. 451—456.

DOI: 10.17587/prin.10.451-456

References

1. **Rodzin S. I., Kureychik V. V.** Status, problems and prospects of bio heuristics development, *Programmnye sistemy i vychislitel'nye metody*, 2016, no. 2, pp. 158—172 (in Russian).

2. **Kureychik V. V., Kureychik V. M., Rodzin S. I.** *Theory of Evolutionary Computation*, Moscow, Fizmatlit, 2012, 260 p. (in Russian).

3. **Črepinšek M., Liu S.-H., Mernik M.** Exploration and exploitation in evolutionary algorithms: a survey, *ACM Computing Surveys*, 2013, vol. 45, no. 3, pp. 35—44.

4. **Virginia Y., Analía A.** A deterministic crowding evolutionary algorithm to form learning teams in a collaborative learning context, *Expert System Appl.,* 2012. vol. 39, no. 10, pp. 8584—8592.

5. **Jie Y., Nawwaf K., Grogono P.** Bi-objective multi population genetic algorithm for multimodal function optimization, *IEEE Trans. Evol. Comput*, 2010, vol. 14, no. 1, pp. 80—102.

6. **Hui W., Zhijian W., Shahryar R.** Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems, *Soft Computing*, 2011, vol. 15, no. 11. P. 2127—2140.

7. **Xiaodong L., Yao X.** Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms, *Proc. of the IEEE congress on evolutionary computation (CEC'09),* 2009, pp. 1546—1553.

8. **Koza J., Andre M., Bennet F., Keane M.** *Genetic programming III: Darwinian invention and problem solving,* Cambridge, MA, MIT Press, 1999, 253 p.

9. **García-Pedrajas N., Castillo J., Ortiz-Boyer D.** A cooperative coevolutionary algorithm for instance selection for instance-based learning, *Machine Learning,* 2010, vol. 78, no. 3, pp. 381—420.

10. **Rodzin S. I., Kureychik V. V.** Theoretical questions and contemporary problems of the development of cognitive bio-inspired optimization algorithms, *Kibernetika i programmirovanie,* 2017, no. 3, pp. 51—79 (in Russian).

11. **Bhattacharya M.** Exploiting landscape information to avoid premature convergence in evolutionary search, *Proc. of the 2006 IEEE congress on evolutionary computation (CEC2006)*, IEEE Press, 2006, pp. 2575—2579.

12. **Ursem R. K.** Diversity-guided evolutionary algorithms, *Proceedings of parallel problem solving from nature VII (PPSN-2002)*, 2002, pp. 462—71.

13. **Aranha C., Iba H.** Modelling cost into a genetic algorithm-based portfolio optimization system by seeding and objective sharing, *Proc. of the IEEE congress on evolutionary computation, CEC2007*, 2007, pp. 196—203.

14. **Aranha C., Iba H.** The memetic tree-based genetic algorithm and its application to portfolio optimization, *Memeting Computer*, 2009, no. 1, pp. 139—51.

15. **Sergienko A. B.** *Test functions for global optimization*, Krasnoyarsk, Publishing house SSAU, 2015, 112 p. (in Russian).

УДК 004.825 DOI: 10.17587/prin.10.457-463

С. Д. Махортов, д-р физ.-мат. наук, зав. кафедрой, msd_exp@outlook.com,
Воронежский государственный университет

Алгебраическая модель интеллектуальной системы с нечеткими правилами*

Алгебраическая теория LP-структур предназначена для моделирования и оптимизации продукционных и подобных им в архитектурно-технологическом плане и по назначению систем в информатике. В предыдущих исследованиях автора были получены результаты, позволяющие обосновывать и автоматизировать решение ряда задач для логических систем продукционного типа: эквивалентные преобразования, устранение избыточности, верификация, ускорение обратного вывода. В настоящей работе представлена LP-структура, семантика которой охватывает нечеткие продукционные системы. В рамках расширенной модели возможности теории LP-структур оказываются доступными при построении и исследовании таких систем.

Ключевые слова: *интеллектуальная система, нечеткие продукции, LP-структура, логическое замыкание, эквивалентные преобразования*

Введение

Информационные технологии играют важнейшую роль в современном мире. Компьютерные системы становятся большими, критически значимыми. В результате возрастают значение и актуальность теоретического обоснования корректности и надежности процессов обработки информации в различных прикладных областях. Основу формального построения и исследования компьютерных систем, созданных на различных технологиях, предоставляют алгебраические структуры [1]. Это в полной мере относится к инженерии знаний, включая широко распространенные в информатике продукционные системы [2—4].

В последнее десятилетие автором и его учениками получен ряд результатов, связанных с логическими системами продукционного типа. Разработана основанная на алгебраических решетках теория LP-структур (*lattice production structures*) [5], которая обосновывает и эффективно решает задачи эквивалентных преобразований, верификации и минимизации баз знаний. Введен и исследован метод релевантного обратного вывода (LP-вывод) [6], существенно снижающий число обращений к внешним источникам информации. Впоследствии эта теория была расширена для моделирования распределенных продукционных систем [7].

Важной особенностью современных интеллектуальных систем является нечеткий характер знаний и рассуждений [8]. В связи с данным обстоятельством представляется актуальным распространение теории LP-структур на нечеткие продукционные системы. Первые шаги в этом направлении были сделаны в работах [9, 10]. Введены понятия, характеризующие нечеткость LP-структуры, и изучены некоторые полезные свойства нечеткого LP-вывода.

Исследования, результаты которых представлены в настоящей работе, посвящены развитию теории нечетких LP-структур для управления нечеткими базами знаний и моделирования нечеткого логического вывода. Введена терминология FLP-структур с нечетким логическим отношением (Fuzzy LP-структуры). Дано определение логического замыкания нечеткого бинарного отношения на решетке, представлена теорема о его существовании. Она позволяет ввести понятие эквивалентных FLP-структур, соответственно в приложениях — эквивалентных баз знаний. Доказана теорема об эквивалентных преобразованиях FLP-структуры. Ее прикладное значение — способ автоматизированных преобразований нечетких баз знаний и его обоснование. Наконец, введено определение и доказано утверждение о приведении нечеткой LP-структуры к каноническому виду. В приложениях такой формат соответствует множеству хорновских нечетких правил [5].

1. Базовая терминология LP-структур

Обозначения и определения, необходимые для дальнейшего изложения, подробно описаны в работе [5]. Для более полного понимания содержания настоящей работы напомним основные из них.

Бинарное отношение R на произвольном множестве F называется: *рефлексивным*, если для любого $a \in F$ справедливо $(a, a) \in R$; *транзитивным*, если для любых $a, b, c \in F$ из (a, b), $(b, c) \in R$ следует $(a, c) \in R$. Как известно [11], существует замыкание произвольного отношения относительно свойств рефлексивности и транзитивности (рефлексивно-транзитивное замыкание).

Обратная задача — нахождение транзитивной редукции: по данному R ищется минимальное отношение

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

R' такое, что его транзитивное замыкание совпадает с транзитивным замыканием R. Напомним, что для частично упорядоченных множеств различаются понятия минимального элемента (для него нет меньшего элемента) и наименьшего элемента (он меньше всех) [12]. В работе [11] представлен алгоритм построения транзитивной редукции ориентированных графов. Показано, что в случае ациклического графа эта задача имеет единственное решение и вычислительно эквивалентна построению транзитивного замыкания.

Базовые сведения о математических решетках содержатся, например, в работе [12]. Решеткой называется множество с частичным порядком $\leqslant$ ("не больше", "содержится"), где для любой пары элементов определены операции $\wedge$ (пересечение) — точная нижняя грань этой пары и $\vee$ (объединение) — ее точная верхняя грань.

Решетка $\mathbb{F}$ называется ограниченной, если она содержит общие нижнюю и верхнюю грани, а именно — такие два элемента O, I, что $O \leqslant a \leqslant I$ для любого $a \in \mathbb{F}$. Атомом ограниченной (снизу) решетки $\mathbb{F}$ называется минимальный элемент ее подмножества $\mathbb{F}\backslash\{O\}$. Решетка называется атомно-порожденной, если каждый ее элемент разложим в виде объединения атомов. Под *LP-структурой* подразумевается алгебраическая система, представляющая собой решетку $\mathbb{F}$, на которой задано дополнительное бинарное отношение R, обладающее некоторыми (продукционно-логическими) свойствами.

Введенное отношение R моделирует совокупность правил (продукций) интеллектуальной системы. Элементы решетки соответствуют предпосылкам и заключениям этих правил, содержащим элементарные факты базы знаний и выражения над ними.

Отношение R на решетке называется *продукционно-логическим*, если оно содержит отношение $\leqslant$, транзитивно и обладает другими свойствами, определяемыми конкретной логической моделью. Одно из таких свойств — *дистрибутивность*. Неформально дистрибутивность отношения означает возможность логического вывода по частям и объединения его результатов на основе решеточных операций $\wedge$ и $\vee$. Заданное на абстрактной решетке отношение R называется:

- $\wedge$-дистрибутивным, если из $(a, b_1), (a, b_2) \in R$ следует $(a, b_1 \wedge b_2) \in R$;
- $\vee$-дистрибутивным, если из $(a_1, b), (a_2, b) \in R$ следует $(a_1 \vee a_2, b) \in R$.

В общем случае отношение называется дистрибутивным при наличии обоих указанных свойств.

В настоящей работе в качестве основы LP-структур рассматриваются решетки с семантикой подмножеств — $\lambda(F)$ (множество конечных подмножеств универсума F). Поэтому вместо символов $\leqslant$, $\geqslant$, $\wedge$ и $\vee$ используются знаки теоретико-множественных операций $\subseteq$, $\supseteq$, $\cap$ и $\cup$, а элементы решетки обозначаются, как правило, большими буквами. Исключение составляют атомы, обозначаемые маленькими буквами.

В рассматриваемой модели под дистрибутивностью отношения подразумевается лишь одно из двух указанных выше свойств. В этой частной нотации $\cup$-дистрибутивность трактуется в следующем смысле: из $(A, B_1), (A, B_2) \in R$ следует $(A, B_1 \cup B_2) \in R$.

Для LP-структур актуальны следующие основные вопросы [5]: о логическом замыкании; об эквивалентных преобразованиях; о канонической форме; о логической редукции. Понятия логического замыкания и редукции представляют собой обобщения для транзитивного замыкания и редукции. Решение перечисленных вопросов открывает возможности обоснования, автоматизированного исследования и оптимизации множеств продукций в различных предметных областях.

Логическим замыканием произвольного бинарного отношения R называется наименьшее продукционно-логическое отношение, содержащее R. Два отношения R_1, R_2 называются (*логически*) *эквивалентными*, если их логические замыкания совпадают. Для таких отношений используется обозначение $R_1 \sim R_2$. *Эквивалентным преобразованием* данного отношения называется некоторая замена подмножества его пар, приводящая к эквивалентному отношению.

Ниже приведены доказанные ранее автором [5] теоремы о существовании логического замыкания произвольного бинарного отношения и о принципе локальности эквивалентных преобразований логических отношений. Представленные результаты открывают возможности для автоматических преобразований баз знаний.

Теорема 1.1. Для произвольного бинарного отношения R на решетке существует логическое замыкание.

Теорема 1.2. Пусть R_1, R_2, R_3, R_4 — отношения на общей решетке. Если при этом $R_1 \sim R_2$ и $R_3 \sim R_4$, то $R_1 \cup R_3 \sim R_2 \cup R_4$.

Отношение R на атомно-порожденной решетке $\mathbb{F}$ называется каноническим, если оно задано множеством пар вида (A, a), где $A \in \mathbb{F}$, a — атом в $\mathbb{F}$, т. е. неразложимый элемент.

Теорема 1.3. Для любого отношения на атомно-порожденной решетке существует эквивалентное ему каноническое отношение.

Поскольку LP-структура представляет собой алгебраическую модель базы знаний, напомним терминологию, связанную с простым видом логических систем продукционного типа — экспертными продукционными системами. Они манипулируют множествами фактов и правил (продукций). *Факт* представляет некоторое суждение о внешнем мире. Он задается триплетом вида "объект.атрибут = <значение>" (например, "термометр.температура = высокая"). В работе [13] для практической реализации продукционных систем триплет сводится к паре "параметр = значение", т. е. объект и атрибут агрегируются в единый параметр. В частности, "термометр.температура" и "термометр.изготовитель" считаются разными параметрами. В терминах настоящей работы в факт интегрируются параметр и его значение, при этом факт считается независимым элементом общего множества. В дальнейшем можно реализовать и более сложную конструкцию фактов, расширив описание LP-структуры.

Продукционная система содержит *рабочую память*. Это некоторое подмножество фактов, которые на текущий момент считаются выполненными. Такое

множество называется также *базой данных* продукционной системы.

Правило (*продукция*) состоит из предпосылки и заключения. *Предпосылка* обычно представляет собой выражение над фактами. Она может быть выполненной (истинной) или невыполненной (ложной) при текущем состоянии рабочей памяти. Если предпосылка верна, то правило может быть применено. *Заключение* — это действие, которое можно осуществить, если верна предпосылка (например, добавить к рабочей памяти некоторый новый факт). *Применение правила* состоит в выполнении действия заключения. В контексте настоящей работы рассматриваются приложения, в которых заключение, как и предпосылка, является выражением над фактами, и соответствующее заключению действие интерпретируется как "считать истинным". Применение правила означает модификацию рабочей памяти, обычно — запись в нее тех фактов, справедливость которых вытекает из истинности выражения в заключении правила. Совокупность правил называется *базой знаний*.

Прямым выводом в продукционной системе называется процесс применения правил к содержимому рабочей памяти (его исходное состояние задано в начале работы) и, соответственно, получение в результате новых фактов, которые считаются справедливыми. *Обратный вывод* — это противоположный процесс. В нем по некоторому набору результирующих фактов — *гипотезе*, путем анализа правил в направлении от заключения к предпосылке подтверждается или опровергается справедливость гипотезы при заданном исходном содержимом рабочей памяти. *Машина вывода* — программный модуль, который непосредственно реализует прямой или обратный вывод.

Предлагаемый подход к исследованию продукционных и подобных им систем основан на представлении множеств фактов и правил LP-структурой. Каждый элементарный факт отображается атомом решетки, предпосылка и заключение правила — соответствующими элементами решетки, а правила представляются парами бинарного отношения R.

2. Нечеткие LP-структуры

Моделирование нечетких знаний и рассуждений существенно приближает теорию интеллектуальных систем к решению практических задач [14]. Во многих современных информационных системах получение "четкого" (точного) решения зачастую невозможно вообще или за приемлемое время. В связи с отмеченным обстоятельством назрела актуальность распространения возможностей теории LP-структур на нечеткие продукционные системы.

В работах [9, 10] введены понятия, придающие LP-структуре нечеткость. Обоснованы некоторые полезные свойства нечеткого LP-вывода. Однако до сих пор не затронут ряд важных стандартных аспектов теории, таких как замыкание, эквивалентные преобразования и эквивалентная минимизация нечетких LP-структур. Основная задача исследования, представленного в настоящей работе, — устранение подобных пробелов.

Руководствуясь указанной целью, начнем с введения базовых понятий теории нечетких LP-структур. Связанные с ней основы нечетких множеств и отношений можно почерпнуть, например, в работе [15].

Как известно, нечеткое множество $A = (F, \mu_F)$ определяется функцией принадлежности μ_F: $F \to [0, 1]$ на некотором (обычном) множестве F. При этом значение $\mu_F(a)$ называется степенью принадлежности a к A. Соответственно, нечеткое бинарное отношение R на множестве F — это нечеткое множество упорядоченных пар элементов из F с заданной функцией принадлежности μ_R: $F \times F \to [0, 1]$.

Нечеткое бинарное отношение R на произвольном множестве F называется *рефлексивным*, если для любого $a \in F$ справедливо $\mu_R(a, a) = 1$.

В целях моделирования нечеткого логического вывода будем использовать композицию нечетких отношений в классической семантике, а именно — (max-min)-композицию. Для отношения R на множестве F композиция $R^2 = R \circ R$ определяется следующим образом:

$$\mu_{R^2}(a, c) = \max_b \left(\min \left(\mu_R(a, b), \mu_R(b, c) \right) \right),$$

где $a, b, c \in F$.

Нечеткое бинарное отношение R на множестве F называется *транзитивным*, если для любых a, b, $c \in F$ справедливо $\mu_R(a, c) \geqslant \min\left(\mu_R(a, b), \mu_R(b, c)\right)$. Нетрудно заметить, что свойство транзитивности для отношения R эквивалентно вложению $R^2 \subseteq R$ в смысле нечетких множеств. Существует также замыкание произвольного нечеткого отношения относительно свойств рефлексивности и транзитивности. Обзор алгоритмов его построения представлен в работе [16].

Пусть дана атомно-порожденная решетка $\mathbb{F}$, представляющая собой множество всех конечных подмножеств некоторого универсума F. На $\mathbb{F}$ рассматривается (дополнительное) нечеткое бинарное отношение R, содержащее $\supseteq$, а также обладающее транзитивностью и дистрибутивностью. Последнее свойство требует уточнения.

Определение 2.1. Нечеткое бинарное отношение $R = (\mathbb{F}, \mu_R)$ называется дистрибутивным, если для любых A, B_1, $B_2 \in \mathbb{F}$ справедливо $\mu_R(A, B_1 \cup B_2) \geqslant \geqslant \min\left(\mu_R(A, B_1), \mu_R(A, B_2)\right)$.

Отношение с указанными выше тремя свойствами будем называть продукционно-логическим или, для краткости, просто логическим.

Определение 2.2. Под нечеткой LP-структурой (FLP-структурой) подразумевается алгебраическая система, представляющая собой решетку $\mathbb{F}$, на которой задано продукционно-логическое отношение R.

Далее для введенного класса алгебраических систем рассматриваются станартные вопросы теории LP-структур о логическом замыкании, об эквивалентных преобразованиях, о канонической форме.

3. Логическое замыкание и преобразования нечетких LP-структур

Заданное на решетке нечеткое бинарное отношение моделирует совокупность нечетких продукций интеллектуальной системы. Оно, как правило, не яв-

ляется логическим, однако может быть рассмотрено его логическое замыкание. Это замыкание по сути содержит всевозможные логические выводы в продукционной системе.

Логическим замыканием нечеткого бинарного отношения R называется наименьшее логическое отношение, содержащее R. Заметим, что отношение включения $\supseteq$ является наименьшим логическим отношением.

Для выяснения вопроса о существовании логического замыкания произвольного отношения R введем отношение логической связи пар элементов решетки при заданном R.

Определение 3.1. Пусть задано нечеткое отношение R на решетке $\mathbb{F}$. Отношение $\bar{R}$ логической связи пар $A, B \in \mathbb{F}$ определяется функцией принадлежности $\bar{\mu}_R$, значение $\bar{\mu}_R(A, B)$ которой вычисляется как *максимальное* из следующих вариантов:

1) $\bar{\mu}_R(A, B) = \mu_R(A, B)$;
2) если $A \supseteq B$, то $\bar{\mu}_R(A, B) = 1$, иначе $= 0$;
3) $\bar{\mu}_R(A, B) = \min\left(\bar{\mu}_R(A, B_1), \bar{\mu}_R(A, B_2)\right)$ $\left(\forall B_1, B_2 : B_1 \cup B_2 = B\right)$;
4) $\bar{\mu}_R(A, B) = \min\left(\bar{\mu}_R(A, C), \bar{\mu}_R(C, B)\right)$ $(\forall C \in \mathbb{F})$.

Замечание 3.1. На основании определения 3.1 нетрудно заметить, что если $R_1 \subseteq R_2$, то $\bar{R}_1 \subseteq \bar{R}_2$.

Рекурсивное определение 3.1 по данному R задает новое нечеткое отношение $\bar{R}$ на решетке $\mathbb{F}$, которое содержит R, $\supseteq$, а также обладает некоторыми дополнительными свойствами. Условия 1—4 определения 3.1 будем также называть правилами вывода (логических связей).

При вычислении некоторой логической связи шагом вывода будем называть применение ровно одного правила, возможно, одновременно к некоторому конечному множеству элементов решетки, например:

- если

$$B_{1,t} \cup B_{2,t} = B_t,\ \bar{\mu}_R(A_t, B_{1,t}) > 0,\ \bar{\mu}_R\left(A_t, B_{2,t}\right) > 0,$$

то $\bar{\mu}_R(A_t, B_t) = \min\left(\bar{\mu}_R(A_t, B_{1,t}), \bar{\mu}_R(A_t, B_{2,t})\right)$, $t \in T$;

- если $\bar{\mu}_R(A_t, C_t) > 0,\ \bar{\mu}_R(C_t, B_t) > 0,$

то $\bar{\mu}_R(A_t, B_t) = \min\left(\bar{\mu}_R(A_t, C_t), \bar{\mu}_R(C_t, B_t)\right)$, $t \in T$.

Уровнем рекурсии при вычислении $\bar{\mu}_R(A, B)$ будем называть число шагов вывода, необходимое для нахождения этого значения. При этом учитываются лишь применения рекурсивных правил 3, 4. Для логической связи, основанной только на правиле 1 или 2, уровень рекурсии считается равным нулю.

Применительно к шагам вывода величины $\bar{\mu}_R(A, B)$ будем употреблять слова "начальный", "последний", а также "предыдущий", "следующий" и т. д. При этом имеется в виду продвижение в направлении прямого логического вывода, т. е. от пар исходного отношения $(R \cup \supseteq)$ к требуемой паре отношения $\bar{R}$. Заметим, что рекурсивное определение 3 сформулировано в рамках метода обратного вывода.

Лемма 3.1. Пусть R — логическое отношение на решетке $\mathbb{F}$ и $A, B \in \mathbb{F}$. Тогда справедливо $\bar{\mu}_R(A, B) = \mu_R(A, B)$, т. е. $\bar{R} = R$.

Доказательство. Проведем его с помощью индукции по m — уровню рекурсии в вычислении $\bar{\mu}_R(A, B)$. При $m = 0$ имеет место одно из условий 1, 2 определения 3.1. Случай 1 непосредственно означает справедливость доказываемого утверждения. Если же было применено условие 2, то и в этом случае $\mu_R(A, B) = 1 = \bar{\mu}_R(A, B)$, поскольку логическое отношение R содержит и $\supseteq$.

Предположим далее, что лемма верна для некоторого $m \geqslant 0$, и докажем ее утверждение при уровне рекурсии $m + 1$. В этом случае новые для рассмотрения варианты могут дать правила 3, 4.

Рассмотрим вариант, когда $\bar{\mu}_R(A, B)$ происходит из правила 3 определения 3.1. При этом $\bar{\mu}_R(A, B_1)$ и $\bar{\mu}_R(A, B_2)$ найдены с уровнем рекурсии $\leqslant m$, поэтому по предположению индукции $\bar{\mu}_R(A, B_1) = \mu_R(A, B_1)$, $\bar{\mu}_R(A, B_2) = \mu_R(A, B_2)$. Тогда, в силу дистрибутивности R, получим

$$\mu_R(A, B) \geqslant \min\left(\mu_R(A, B_1), \mu_R(A, B_2)\right) = \bar{\mu}_R(A, B).$$

Однако, поскольку отношение R содержится в $\bar{R}$, последнее неравенство может быть выполнено лишь при $\mu_R(A, B) = \bar{\mu}_R(A, B)$.

Наконец, пусть при вычислении $\bar{\mu}_R(A, B)$ последним было использовано правило 4. И в этом случае $\bar{\mu}_R(A, C)$ и $\bar{\mu}_R(C, B)$ имеют рекурсию уровня $\leqslant m$. Следовательно, по предположению индукции $\bar{\mu}_R(A, C) = \mu_R(A, C)$, $\bar{\mu}_R(C, B) = \mu_R(C, B)$. Отсюда, в силу транзитивности логического отношения R, имеем

$$\mu_R(A, B) \geqslant \min\left(\mu_R(A, C), \mu_R(C, B)\right) = \bar{\mu}_R(A, B).$$

Как и выше, поскольку R содержится в отношении логической связи $\bar{R}$, здесь возможно лишь $\mu_R(A, B) = \bar{\mu}_R(A, B)$.

Теорема 3.1. Для произвольного отношения R на решетке $\mathbb{F}$ логическое замыкание существует и совпадает с нечетким отношением логической связи $\bar{R}$, определяемым функцией принадлежности $\mu_{\bar{R}} = \bar{\mu}_R$.

Доказательство. Заметим вначале, что при произвольном R соответствующее ему отношение $\bar{R}$ является логическим. Действительно, в силу условия 2 определения 3.1 оно содержит $\supseteq$, из условия 3 следует его дистрибутивность, а правило 4 означает транзитивность данного отношения. Далее, в силу условия 1 определения 3.1, отношение $\bar{R}$ содержит R. Для доказательства теоремы осталось показать, что это — наименьшее из логических отношений, обладающих этим свойством.

Пусть R_2 — любое логическое отношение, содержащее R. Тогда (см. замечание 3.1) $R \subseteq \bar{R} \subseteq \bar{R}_2$. Вместе с тем по лемме 3.1 имеем $\bar{R}_2 = R_2$. Таким образом, построенное в определении 3.1 отношение $\bar{R}$ содержится в произвольно выбранном R_2 и является наименьшим логическим отношением, содержащим R. □

Понятие логического замыкания и доказанная теорема о его существовании позволяют рассмотреть вопросы эквивалентности нечетких LP-структур.

Определение 3.2. Два нечетких отношения R, P на общей решетке называются (*логически*) *эквивалентными* ($R \sim P$), если их логические замыкания

совпадают. Эквивалентным преобразованием нечеткого отношения R называется такая модификация его функции принадлежности ($\mu_R \to \mu_P$), что полученное в результате новое отношение P логически эквивалентно R.

Следствие 3.1. Пусть R — нечеткое отношение на решетке $\mathbb{F}$ с функцией принадлежности μ_R и $\{(A_t, B_t) \mid A_t, B_t \in \mathbb{F}; t \in T\}$ — некоторое множество пар. Определим новое отношение R' следующей функцией принадлежности:

$$\mu_{R'}(A, B) = \begin{cases} \bar{\mu}_R(A_t, B_t) & \text{при } A = A_t,\ B = B_t,\ t \in T \\ \mu_R(A, B) & \text{иначе} \end{cases}. \quad (1)$$

Тогда отношение R' эквивалентно R.

Доказательство. Из определения 3.1 имеем $R \subseteq \bar{R}$, т. е. $\mu_R(A, B) \leqslant \bar{\mu}_R(A, B)$ $(\forall A, B \in \mathbb{F})$. Отсюда по определению отношения R' (равенство (1)) следуют включения $R \subseteq R' \subseteq \bar{R}$. Применяя к этим включениям замечание 3.1 и учитывая, что логические замыкания обоих отношений R, $\bar{R}$ равны $\bar{R}$, получаем $\bar{R}' = \bar{R}$. □

Теорема 3.2. Пусть R_1, R_2, R_3, R_4 — нечеткие отношения на общей решетке $\mathbb{F}$. Если при этом $R_1 \sim R_2$ и $R_3 \sim R_4$, то $R_1 \cup R_3 \sim R_2 \cup R_4$.

Доказательство. Необходимо доказать равенство двух нечетких логических отношений $\overline{R_1 \cup R_3}$ и $\overline{R_2 \cup R_4}$. Выберем произвольную пару элементов $A, B \in \mathbb{F}$. Докажем, например, что справедливо $\bar{\mu}_{R_1 \cup R_3}(A, B) \leqslant \bar{\mu}_{R_2 \cup R_4}(A, B)$ (обратное неравенство доказывается симметрично). Для этого применим метод индукции по m — уровню рекурсии при вычислении $\bar{\mu}_{R_1 \cup R_3}(A, B)$.

При $m = 0$ значение $\bar{\mu}_{R_1 \cup R_3}(A, B)$ вычисляется по одному из правил 1, 2 определения 3.1. Если использовано правило 2, то в этом случае $\bar{\mu}_{R_1 \cup R_3}(A, B) = \bar{\mu}_{R_2 \cup R_4}(A, B) = 1$, поскольку логическое отношение $\overline{R_2 \cup R_4}$ содержит $\subseteq$. Если было применено правило 1, то по нему и по определению объединения нечетких множеств имеем

$$\bar{\mu}_{R_1 \cup R_3}(A, B) = \mu_{R_1 \cup R_3}(A, B) = \max\left(\mu_{R_1}(A, B), \mu_{R_3}(A, B)\right).$$

Пусть для определенности $\bar{\mu}_{R_1 \cup R_3}(A, B) = \mu_{R_1}(A, B)$ (вариант $\bar{\mu}_{R_1 \cup R_3}(A, B) = \mu_{R_3}(A, B)$ симметричен). В этом случае в силу условия эквивалентности $R_1 \sim R_2$ получим $\bar{\mu}_{R_1 \cup R_3}(A, B) \leqslant \bar{\mu}_{R_1}(A, B) = \bar{\mu}_{R_2}(A, B)$. Следовательно, (см. замечание 3.1), справедливо и $\bar{\mu}_{R_1 \cup R_3}(A, B) \leqslant \bar{\mu}_{R_2 \cup R_4}(A, B)$. Таким образом, при $m = 0$ утверждение теоремы доказано.

Предположим далее, что оно верно для некоторого $m \geqslant 0$, и докажем его при уровне рекурсии $m + 1$. В этом случае новые для рассмотрения варианты вычисления $\bar{\mu}_{R_1 \cup R_3}(A, B)$ могут дать правила 3, 4 определения 3.1.

Рассмотрим вариант, когда на последнем шаге вычисления $\bar{\mu}_{R_1 \cup R_3}(A, B)$ происходит из правила 3 определения 3.1 (с некоторым разложением $B = B_1 \cup B_2$). Поскольку $\bar{\mu}_{R_1 \cup R_3}(A, B_1)$, $\bar{\mu}_{R_1 \cup R_3}(A, B_2)$ найдены с уровнем рекурсии $\leqslant m$, то по предположению индукции

$$\bar{\mu}_{R_1 \cup R_3}(A, B_1) = \bar{\mu}_{R_3 \cup R_4}(A, B_1),$$
$$\bar{\mu}_{R_1 \cup R_3}(A, B_2) = \bar{\mu}_{R_2 \cup R_4}(A, B_2).$$

Тогда из правила 3 имеем

$$\bar{\mu}_{R_1 \cup R_3}(A, B) = \min\left(\bar{\mu}_{R_2 \cup R_4}(A, B_1), \bar{\mu}_{R_2 \cup R_4}(A, B_2)\right),$$

откуда в силу дистрибутивности $\overline{R_2 \cup R_4}$ получим требуемое неравенство $\bar{\mu}_{R_1 \cup R_3}(A, B) \leqslant \bar{\mu}_{R_2 \cup R_4}(A, B)$.

Наконец, пусть при вычислении $\bar{\mu}_{R_1 \cup R_3}(A, B)$ последним было использовано правило 4. В этом случае $\bar{\mu}_{R_1 \cup R_3}(A, C)$ и $\bar{\mu}_{R_1 \cup R_3}(C, B)$ имеют рекурсию уровня $\leqslant m$. Следовательно, по предположению индукции

$$\bar{\mu}_{R_1 \cup R_3}(A, C) = \bar{\mu}_{R_2 \cup R_4}(A, C),\ \bar{\mu}_{R_1 \cup R_3}(C, B) = \bar{\mu}_{R_2 \cup R_4}(C, B).$$

Поэтому из правила 4 имеем

$$\bar{\mu}_{R_1 \cup R_3}(A, B) = \min\left(\bar{\mu}_{R_2 \cup R_4}(A, C), \bar{\mu}_{R_2 \cup R_4}(C, B)\right),$$

откуда в силу транзитивности логического отношения $\overline{R_2 \cup R_4}$ получим $\bar{\mu}_{R_1 \cup R_3}(A, B) \leqslant \bar{\mu}_{R_2 \cup R_4}(A, B)$. □

Следствие 3.2. Пусть R_1, R_2, R_3 — нечеткие отношения на общей решетке $\mathbb{F}$. Если при этом $R_1 \sim R_2$, то $R_1 \cup R_3 \sim R_2 \cup R_3$.

Следствия 3.1 и 3.2 обосновывают принципы локально-эквивалентных преобразований FLP-структур. Полученные результаты могут быть использованы, в частности, для эквивалентного упрощения нечетких отношений на решетке.

Лемма 3.2. Пусть R — нечеткое отношение на решетке $\mathbb{F}$. Пусть также $A, B \in \mathbb{F}$, причем $B = \bigcup B_t$ — объединение элементов $B_t \in \mathbb{F}$ $(t \in T)$. Тогда отношение R', полученное из R представленной ниже модификацией функции принадлежности μ_R, эквивалентно R:

$$\mu_{R'}(X, Y) =$$
$$= \begin{cases} \max\left(\mu_R(A, B), \mu_R(A, B_t)\right) & \text{при } X = A,\ Y = B_t,\ t \in T; \\ 0 & \text{при } X = A,\ Y = B; \\ \mu_R(X, Y) & \text{иначе.} \end{cases}$$

Доказательство. Введем нечеткие отношения R_1, R_2, R_3, определяемые соответственно следующими функциями принадлежности:

$$\mu_1(X, Y) = \begin{cases} \mu_R(A, B) & \text{при } X = A,\ Y = B; \\ 0 & \text{иначе} \end{cases}$$

$$\mu_2(X, Y) = \begin{cases} \mu_R(A, B) & \text{при } X = A,\ Y = B_t,\ t \in T; \\ 0 & \text{иначе} \end{cases}$$

$$\mu_3(X, Y) = \begin{cases} 0 & \text{при } X = A,\ Y = B \\ \mu_R(X, Y) & \text{иначе.} \end{cases}$$

Нетрудно заметить, что отношения R_1, R_2, R_3 удовлетворяют условиям следствия 3.2, причем $R_1 \cup R_3 = R$ и $R_2 \cup R_3 = R'$, что и доказывает лемму. □

В качестве применения доказанного результата рассмотрим вопрос о приведении нечеткой LP-структуры к каноническому виду. Нечеткое отношение R на атомно-порожденной решетке $\mathbb{F}$ называется *каноническим*, если его функция принадлежности положительна лишь на парах вида (A, a), где $A \in \mathbb{F}$, a — атом в $\mathbb{F}$. Каноническое отношение в моделируемой продукционной системе соответствует множеству правил так называемого *хорновского* типа.

Теорема 3.3. Для произвольного нечеткого отношения R на атомно-порожденной решетке $\mathbb{F}$ существует эквивалентное ему каноническое отношение.

Доказательство легко следует из леммы 3.2. Применим ее к каждой паре A, $B \in \mathbb{F}$, для которой $\mu_R(A, B) > 0$ и B не является атомом в $\mathbb{F}$. При этом будем использовать представление $B = \bigcup b_t$ в виде атомов решетки. Тогда получим эквивалентное R каноническое отношение R'. □

Заключение

В настоящей работе введен и исследован новый класс алгебраических систем — нечеткие LP-структуры, семантика которых охватывает интеллектуальные системы с нечеткими правилами. Для данного класса рассмотрены стандартные вопросы теории LP-структур: о логическом замыкании, об эквивалентных преобразованиях, о канонической форме. Представленные теоретические результаты могут быть применены в задачах управления знаниями нечетких продукционных систем.

В качестве перспектив данного исследования можно указать постановку и решение задачи устранения избыточности в нечетких базах знаний, а также распространение использованного подхода на распределенные нечеткие системы.

Список литературы

1. **Бениаминов Е. М.** Алгебраические методы в теории баз данных и представлении знаний. М.: Научный мир, 2003. 184 с.
2. **Жожикашвили А. В., Стефанюк В. Л.** Алгебраическая теория продукционных систем // VIII нац. конф. по искусственному интеллекту с международным участием КИИ-2002: Тр. конференции. Т. 1. М.: Физматлит, 2002. С. 428—436.
3. **Maciol A.** An application of rule-based tool in attributive logic for business rules modeling // Expert Systems with Applications. 2008. Vol. 34, N. 3. P. 1825—1836.
4. **Дородных Н. О., Юрин А. Ю.** Использование диаграмм классов UML для формирования продукционных баз знаний // Программная инженерия. 2015. № 4. С. 3—9.
5. **Махортов С. Д.** Математические основы искусственного интеллекта: теория LP-структур для построения и исследования моделей знаний продукционного типа / Под ред. В. А. Васенина. М.: Изд-во МЦНМО, 2009. 304 с.
6. **Болотова С. Ю., Махортов С. Д.** Алгоритмы релевантного обратного вывода, основанные на решении продукционно-логических уравнений // Искусственный интеллект и принятие решений. 2011. № 2. С. 40—50.
7. **Махортов С. Д.** Алгебраическая модель распределенной логической системы продукционного типа // Программная инженерия. 2015. № 12. С. 32—38.
8. **Батыршин И. З., Недосекин А. О., Стецко А. А.** и др. Нечеткие гибридные системы: Теория и практика / Под ред. Н. Г. Ярушкиной. М.: Физматлит, 2007. 208 с.
9. **Махортов С. Д., Шмарин А. Н.** Оптимизация метода LP-вывода // Нейрокомпьютеры. Разработка, применение. 2013. № 9. С. 59—63.
10. **Махортов С. Д., Шмарин А. Н.** Нечеткий LP-вывод и его программная реализация // Программная инженерия. 2013. № 12. С. 34—38.
11. **Aho A. V., Garey M. R., Ullman J. D.** The transitive reduction of a directed graph // SIAM Journal of Computing. 1972. Vol. 1, N. 2. P. 131—137.
12. **Биркгоф Г.** Теория решеток: пер. с англ. М.: Наука, 1984. 568 с.
13. **Sowyer B., Foster D.** Programming Expert Systems in Pascal. John Wiley & Sons, Inc., 1986. 186 p.
14. **Чернов В. Г., Ганьшина С. И.** Экспертная система для ипотечного кредитования, основанная на нечетких продукционных правилах // Прикладная информатика. 2012. № 4 (40). С. 88—99.
15. **Рыжов А. П.** Элементы теории нечетких множеств и ее приложений. М.: Диалог-МГУ, 2003. 81 с.
16. **Garmendia L., Del Campo R. G., López V., Recasens J.** An Algorithm to Compute the Transitive Closure, a Transitive Approximation and a Transitive Opening of a Fuzzy Proximity // Mathware & Soft Computing. 2009. Vol. 16, N. 2. P. 175—191.

An Algebraic Model of the Intelligent System with Fuzzy Rules

S. D. Makhortov, msd_exp@outlook.com, Voronezh State University, Voronezh, 394018, Russian Federation

Corresponding author:

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

Received on August 16, 2019
Accepted on September 04, 2019

Information technologies play a crucial role in the modern world. Computer systems are becoming big, critically significant. As a result, importance and relevance of the theoretical justification of correctness and reliability of information processing in various application areas are increasing. The basis for formal construction and research of computer systems based on various technologies is provided by algebraic structures. This fully applies to knowledge engineering, including production systems widely used in computer science.

In the last decade, the author and his students have obtained a number of results related to logical systems of the production type. On the basis of algebraic lattices, there was developed a theory of LP structures (lattice production structures), which substantiates and effectively solves the problems of equivalent transformations, verification and minimization of knowledge bases. The method of backward inference (LP-inference), significantly reducing the number of appeals to external information sources, was introduced and studied. Subsequently, this theory was expanded for distributed production systems modeling.

An important feature of modern intelligent systems is the fuzzy nature of knowledge and reasoning. In this cir-

cumstance, it seems to be relevant to extend the theory of LP structures to fuzzy production systems. The first steps in this direction were taken in two previous works with the author's participation. The concepts characterizing the fuzziness of the LP structure were introduced, and some useful properties of the fuzzy LP-inference were studied.

The present work is devoted to the development of the theory of fuzzy LP structures for managing fuzzy knowledge bases and modeling fuzzy logical inference. The terminology of FLP structures (Fuzzy LP structures) with fuzzy logical relation is introduced. The definition of the logical closure of a fuzzy binary relation on a lattice is given; a theorem on its existence is presented. The theorem allows to introduce the concept of equivalent FLP structures and, respectively, in applications — of equivalent knowledge bases. The theorem on equivalent transformations of the FLP structure is proved. Its applied value is the method and its justification for automated transformations of fuzzy knowledge bases. Finally, a definition is introduced and a statement on the reduction of a fuzzy LP structure to a canonical form is proved. In applications, this format corresponds to a set of Horn fuzzy rules.

Keywords: *intelligent system, fuzzy productions, LP structure, logical closure, equivalent transformations*

For citation:

Makhortov S. D. An Algebraic Model of the Intelligent System with Fuzzy Rules, *Programmnaya Ingeneria,* 2019, vol. 10, no. 11—12, pp. 457—463.

DOI: 10.17587/prin.10.457-463

References

1. **Beniaminov E. M.** *Algebraic methods in the theory of databases and knowledge representation*, Nauchnyj mir, 2003, 184 p. (in Russian).
2. **Zhozhikashvili A. V., Stefanjuk V. L.** Algebraic theory of production systems, *VIII nacional'naja konferencija po iskusstvennomu intellektu s mezhdunarodnym uchastiem KII-2002,* Kolomna, October 7—12, 2002, Trudy konferencii, Vol. 1, Fizmatlit, 2002, pp. 428—436 (in Russian).
3. **Maciol A.** An application of rule-based tool in attributive logic for business rules modeling, *Expert Systems with Applications*, 2008, vol. 34, no. 3, pp. 1825—1836.
4. **Dorodnykh N. O., Yurin A. Yu.** The Use of Diagrams of UML Classes for Production Knowledge Bases Formation, *Programmnaya Ingeneria*, 2015, no. 4, pp. 3—9 (in Russian).
5. **Makhortov S. D.** *Mathematical Foundations of Artificial Intelligence: The LP structures theory for the knowledge models of production type construction and research* / Eds. V. A. Vasenin, Moscow, MCCME, 2009, 304 p. (in Russian).
6. **Bolotova S. Yu., Makhortov S. D.** Algorithms of the relevant backward inference that is based on production-logic equations solving, *Iskusstvennyiy intellekt i prinyatie resheniy*, 2011, no. 2, pp. 40—50 (in Russian).
7. **Makhortov S. D.** The algebraic model of the distributed logical system of the production type*, Programmnaya Ingeneria*, 2015, no. 12, pp. 32—38 (in Russian).
8. **Batyrshin I. Z., Nedosekin A. O., Stecko A. A., Tarasov V. B., Yazenin A. V., Yarushkina N. G.** *The Fuzzy Gibrid Systems: Theory and Practice*, Moscow, Fizmatlit, 2007, 208 p. (in Russian).
9. **Makhortov S. D., Shmarin A. N.** Optimizing LP-inference method, *Nejrokomp'jutery. Razrabotka, primenenie*, 2013, no. 9, pp. 59—63 (in Russian).
10. **Makhortov S. D., Shmarin A. N.** Fuzzy LP-inference and its software implementation, *Programmnaya Ingeneria*, 2013, no. 12, pp. 34—38 (in Russian).
11. **Aho A. V., Garey M. R., Ulman J. D**. The transitive reduction of a directed graph, *SIAM Journal of Computing,* 1972, vol. 1, no. 2, pp. 131—137.
12. **Birkhoff G.** Lattice Theory, Rhode Island, 1995, 420 p.
13. **Sowyer B., Foster D**. *Programming Expert Systems in Pascal,* John Wiley & Sons, Inc., 1986, 186 p.
14. **Chernov V. G., Ganshina S. I.** Expert Mortgage Lending System Based on Fuzzy Production Rules, *Prikladnaya Informatika*, 2012, no. 4 (40), pp. 88—99 (in Russian).
15. **Ryzhov A. P.** *Elements of the Theory of Fuzzy Sets and of its Applications,* Moscow, Dialog-MGU, 2003, 81 p. (in Russian).
16. **Garmendia L., Del Campo R. G., López V., Recasens J.** An Algorithm to Compute the Transitive Closure, a Transitive Approximation and a Transitive Opening of a Fuzzy Proximity, *Mathware & Soft Computing*, 2009, vol. 16, no. 2, pp. 175—191.

ИНФОРМАЦИЯ

Продолжается подписка на журнал "Программная инженерия" на первое полугодие 2020 г.

Оформить подписку можно через подписные агентства
или непосредственно в редакции журнала.

Подписной индекс по каталогу

Пресса России — 22765

Сообщаем, что с 2020 г. возможна подписка
на электронную версию нашего журнала.

Подписку на электронную версию журнала можно оформить через
ООО "ИВИС": тел. (495) 777-65-57, 777-65-58;
e-mail: sales@ivis.ru и Агентство "Урал-Пресс http://ural-press.ru (индекс 013312)

Адрес редакции: 107076, Москва, Стромынский пер., д. 4,
Издательство "Новые технологии",
редакция журнала "Программная инженерия"

Тел.: (499) 269-53-97. Факс: (499) 269-55-10. E-mail: prin@novtex.ru

УДК 519.674 DOI: 10.17587/prin.10.464-470

А. П. Кудряшов, канд. техн. наук, мл. науч. сотр., kudryashovA@dvo.ru,
И. В. Соловьёв, аспирант, igorek.solovyev@mail.ru, Институт прикладной математики Дальневосточного отделения Российской академии наук, Владивосток

Выделение объектов на топографическом плане для реконструкции сцены городского пространства

Предложен метод реконструкции трехмерных городских сцен с использованием топографического плана. Применяется модифицированный волновой алгоритм для выделения контуров зданий на топографическом плане. Предложена технология текстурирования зданий на основании фотоизображений, полученных из открытой базы данных снимков. Показана гибкость метода реконструкции к исходным данным.

Ключевые слова: *трехмерная реконструкция, топографический план, географические координаты, текстурирование, городская обстановка*

Введение

Компьютерное моделирование на основе реконструкции реальных объектов окружающего мира является важной и актуальной задачей компьютерной графики и машинного зрения. Полученные таким образом объекты и сцены широко используются в научных, служебных и развлекательных целях. Одна из практических задач — реконструкция сцен городского пространства, которые позволяют ориентироваться на местности, помогают учитывать существующую композицию объектов при проектировании новых сооружений, а также для туристической и развлекательной сфер деятельности.

Городское пространство может состоять из множества классов объектов. Некоторые из них являются его обязательными составляющими. Другие дополняют его для придания сцене большей реалистичности. В зависимости от вида прикладной задачи состав может варьироваться, однако к общему списку классов объектов относятся здания, рельеф, деревья, кусты, светофоры, уличные фонари, автомобили, люди.

Уровень детализации сцены и отдельных ее объектов различается в зависимости от прикладной задачи. Так, например, здания могут быть смоделированы вплоть до воссоздания своей точной трехмерной формы или же могут представлять собой некоторый геометрический примитив. Текстуры зданий могут являться частью фотоизображения реального объекта или иметь некий условный вид для определенного типа строения. Во всех этих случаях требуется соблюдать баланс между детализацией сцены и временем ее реконструкции.

Ручное моделирование трехмерной сцены городского пространства с высокой степенью детализации требует огромных затрат времени и сил. Автоматизация реконструкции позволяет значительно сократить время, но имеет ряд недостатков.

К текущему моменту разработаны методы, направленные на решение задач реконструкции трехмерных сцен городского пространства. В качестве исходных данных некоторые из них используют фотоснимки, на которых алгоритмы выделяют геометрические примитивы [1, 2].

Другой подход — использование топографического плана, который содержит полезную информацию для моделирования сцены. За последние два десятилетия появился ряд как отечественных, так и иностранных решений, в рамках которых осуществляется реконструкция городской обстановки по топографическому плану. В работе [3] авторы предлагают метод распознавания контуров, однако этот метод не позволяет распознавать вложенную служебную информацию, что не дает точно идентифицировать здание и определить его тип и этажность. Автоматизированный алгоритм реконструкции городской сцены предлагают авторы работы [4]. Метод основан на использовании системы LIDAR для лазерного сканирования трехмерного пространства и на сопоставлении полученного облака точек с топографическим планом на основании элементов тензорного анализа. Метод не предусматривает автоматического текстурирования моделей, к тому же требует наличия сложного измерительного прибора.

В настоящей работе используется подход, при котором необходимо выделить все контуры на изображении и найти среди них здания, распознать внутри них текст, а также определить объекты, которые указаны с помощью условных обозначений: деревья, клумбы, уличные фонари, светофоры и т. д. Этот набор данных может быть преобразован в трехмерную модель сцены, в которой дополнительно может быть построен рельеф по географической привязке топографического плана. На здания наносятся текстуры, которые соответствуют либо их реальным прототипам, либо общим типам этих строений. Предлагаемый метод является развитием более ранних работ авторов [5—7].

Выделение контуров на топографическом плане

Основная информационная ценность топографического плана (рис. 1) — это расположение зданий с информацией о них и других объектов, таких как электросети, фонари, светофоры, дороги, деревья и т. д. Поэтому важно найти все объекты на сцене и определить их тип. Результат данного этапа — набор контуров. При этом каждый из них состоит из упорядоченной последовательности точек.

Методы, позволяющие выделить на изображении контуры (методы Робертса, Собеля [8], Превитта, Канни-детектора [9]), основаны на применении фильтров к изображению некоторым окном-матрицей и на нахождении отклика в каждой его точке. Сильной стороной данного подхода является его универсальность, так как он дает возможность обрабатывать любые изображения. При этом указанные алгоритмы нельзя применить для поиска зданий на топографическом плане в силу того, что они не могут получить контур, в котором все точки будут являться упорядоченной последовательностью.

Поэтому для выделения контуров предлагается использовать волновой алгоритма Ли [10], который изначальной был предназначен для поиска кратчайшего пути выхода из лабиринта. Если в некоторой точке топографического плана запустить волновой алгоритм, то он выделит все точки контура, внутри которого находилась исходная точка. После этого по данным точкам снова должен быть запущен волновой алгоритм, для того чтобы их упорядочить. Таким образом, во всех точках топографического плана выполняется данная последовательность действий. Это приведет к тому, что на изображении будут выделены все контуры и при этом они будут упорядочены. Подробно данный алгоритм описан в работе [5].

Классификация контуров

Выделенные на топографическом плане контуры могут принадлежать различным объектам. Часть из них может являться результатом выделения незначащей информации, например, изолинии рельефа, линии электропередач, а также их пересечения с дорогами или линиями контуров дадут ложные замкнутые контуры, которые нельзя классифицировать как отдельный объект. Такие контуры необходимо исключить из общего набора контуров. Остальные нужно рассортировать по видам, так как они могут быть контурами зданий, других объектов городской обстановки или контурами текстовой информации.

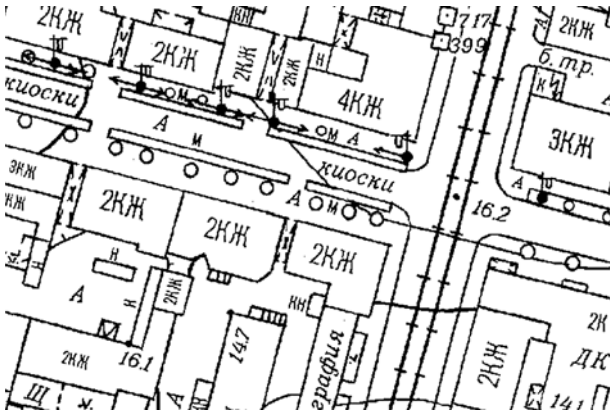

Рис. 1. Топографический план

На данном этапе определяются уникальные характеристики каждого класса объектов. Некоторые требования к каждому классу объектов можно поставить исходя из условий реального окружения. Перечислим признаки того, что объект на топографическом плане является зданием.

1. Форма объекта — это многоугольник. В реальности встречаются объекты иных форм (круглые, полукруглые и т. д.), однако всех их можно свести к форме многоугольника путем увеличения числа вершин.

2. Число сторон многоугольника больше трех и не больше какого-то определенного числа. В окружающем мире могут быть встречены здания с тремя сторонами. Однако такие объекты достаточно редки. А вот вероятность встретить на топографическом плане ложные объекты, которые в результате пересечения с другими объектами имели бы треугольную форму, достаточно высока. Поэтому будем пренебрегать данным частным случаем ради сокращения числа ложных распознаваний. Аналогичная ситуация со зданиями, у которых число стен (равно как и вершин многоугольника) слишком большое. На практике было выявлено, что число стен зданий обычно не превышает 20 штук.

3. Число прямых углов в многоугольнике значительно больше других. Если здание не имеет круглых форм, то в большинстве случаев его стороны перпендикулярны друг другу. Число острых и тупых углов значительно меньше прямых. На практике установлено, что здания должны содержать больше 80 % прямых углов.

4. Внутри объекта обязательно есть символы, которые обозначают его внутреннюю информацию.

Использование волнового алгоритма совместно с данной классификацией позволило выделять здания на топографическом плане с точность выше 90 %.

Дополнительные объекты сцены, такие как светофоры, деревья, клумбы, уличные фонари (рис. 2) и т. п., распознаются исходя из своих уникальных характеристик. Например, светофор — это контур, у которого внутри находятся три контура, центры масс которых лежат примерно на одной прямой.

При таком подходе дополнительные объекты сцены классифицируются с точностью более 80 %.

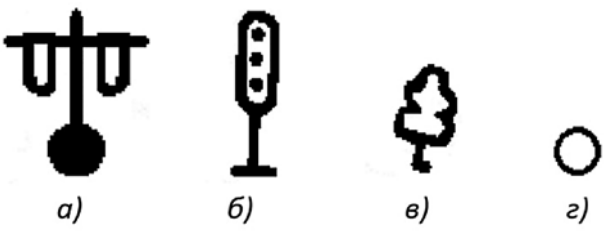

Рис. 2. Символы на топографическом плане:
а — фонарь; *б* — светофор; *в* — дерево; *г* — клумба

Распознавание внутренней информации зданий

После классификации всех выделенных контуров возникает необходимость определить дополнительную информацию, которая располагается внутри контуров зданий. Эта информация содержит данные о том, сколько этажей у здания, из чего оно сделано и жилое/нежилое ли оно. Число этажей здания напрямую влияет на его геометрическую форму на сцене. Остальные параметры внутренней информации требуются для определения внешнего вида здания, т. е. текстуры, которую оно должно иметь.

Внутренняя информация о зданиях состоит из условных символьных обозначений. Число этажей обозначается цифрами, а другая информация буквами. Выделение символов осуществляется также с помощью волнового алгоритма. В результате работы алгоритма образуются отдельные наборы выделенных точек, каждый из которых необходимо будет классифицировать как определенный символ. Каждый такой набор точек сравнивается с эталонным изображением символа и выбирается тот, у которого коэффициент корреляции максимален.

После распознавания символов всей надписи внутри здания требуется проверить ее на соответствие шаблонам, которые приняты для обозначения внутренней информации. Например, если была получена надпись "2ЖЖ", то она преобразуется в надпись "2КЖ" (двухэтажное кирпичное жилое здание). В результате будет получена информация о числе этажей и типе каждого здания. Такой подход позволяет верно определить данную информацию с точностью более чем 95 %.

Построение 3D-модели сцены

Реконструированная сцена должна содержать трехмерные модели всех выделенных на ней объектов. Топографический план не дает информации о точной геометрической трехмерной форме зданий, поэтому в качестве объемной модели здания будет использоваться прямая призма с основанием, обозначенным на топографическом плане. На основании полученных данных о числе этажей и условной высоте одного этажа будет определяться высота всей модели (рис. 3). Для большей визуальной реалистичности генерируется случайная крыша для каждого здания.

Другие объекты сцены, такие как деревья, кусты, светофоры и уличные фонари, обозначаются условными знаками. В связи с этим обстоятельством установить их реальные размеры и геометрическую форму не представляется возможным. Геометрическая форма таких объектов неизвестна, поэтому каждому их них ставится в соответствие определенная условная шаблонная трехмерная модель. Линейные размеры таких моделей могут быть заданы относительно высоты одного этажа здания.

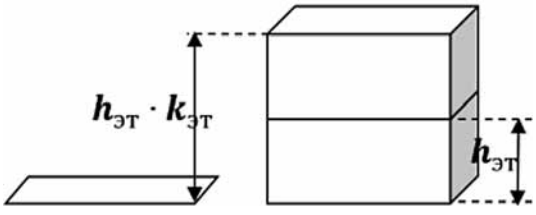

Рис. 3. Образование трехмерного каркаса на основе формы здания:
$h_{эт}$ — высота одного этажа; $k_{эт}$ — число этажей

Одним из главных компонентов сцены, которые придают ей максимальную информативность и реалистичность, является рельеф местности. На топографических планах, изображающих большую территорию местности, перепады уровня высот особенно заметны и значимы. Топографический план содержит информацию об уровне высот. На нем присутствуют геодезические линии и числа, обозначающие высоту над уровнем моря. Ввиду наличия большого числа объектов на топографическом плане, выделить эти данные проблематично. Однако для построения рельефа можно воспользоваться сторонними базами данных (Google Maps, GMTED2010, ASTER GDEM2, NextMap World 30 и др.). Поэтому топографический план может быть привязан к географическим координатам, что обеспечит наличие данных об уровне его высот. Информация о рельефе местности получается на основании двух заранее известных географических координат, которые соответствуют известным точкам на топографическом плане. На основании этих данных строится триангуляционная поверхность всей территории. Высота в каждой точке этой поверхности вычисляется путем обращения к сервису Google Maps Elevation API. В сервис передаются рассчитанные по двум исходным географическим точкам координаты каждой вершины поверхности.

Текстурирование сцены

Поскольку реконструируемые здания будут построены весьма условно, то для увеличения реалистичности сцены стоит использовать текстуру. Для это требуется, чтобы каждый объект сцены имел текстуру, максимально приближенную к реальному объекту.

Внешний вид здания имеет бóльшую информативность, чем его геометрическая форма, так как на зданиях, в большинстве случаев, присутствует дополнительная информация, которая отсутствует на топографическом плане (названия, адреса, цветовая гамма и т. п.). Эта информация служит дополнительным ориентиром на территории сцены. В качестве текстур здания могут служить фотографии его сторон, так как в подавляющем большинстве случаев они имеют прямоугольные формы. В условиях городской постройки не всегда есть возможность получить идеальные фотоснимки стен зданий, которые можно было бы использовать для текстурирования. На фотографиях необходимо выполнить коррекцию перспективы, выделив только те участки, которые относятся к стенам зданий. Каждый такой снимок должен быть обработан перед нанесением на модель здания.

Нанесение качественных текстур на сцены значительно увеличивает ее реалистичность. Предлагается проводить автоматическое текстурирование зданий фотореалистичными изображениями при наличии следующих данных:

- географическое положение и направление камеры;
- координаты углов стены на фотоснимке для коррекции перспективы;
- географические координаты всех объектов сцены.

Современные телефоны и планшеты позволяют делать фотоснимки с записью географических координат, определяемых с помощью систем GPS или ГЛОНАСС.

Нахождение углов стены на фотоснимке — довольно сложная и нетривиальная задача, поэтому выделение стены происходит вручную. Для этого разработано специальное программное обеспечение для мобильных устройств (рис. 4).

На основании выделенных четырех точек стены корректируется перспектива на фотоснимке (рис. 5).

Для каждого фотоснимка на модели необходимо определить здание и стену, на которую должна быть нанесена текстура. Каждый снимок содержит данные о географических координатах места и направления съемки. На основании географической привязки определяются координаты места съемки на топографическом плане.

Все фотоснимки переносятся в пространство топографического плана. Определение стены здания, к которой принадлежит фотоснимок текстуры, осуществляется с помощью данных о направлении съемки. На топографическом плане проводится луч по направлению съемки из точки, с которой она осуществлялась. По каждому зданию определяются стены, с которыми пересекся луч. Из всех этих стен выбирается та стена, точка пересечения с которой ближе остальных к точке съемки (рис. 6).

У стены на рис. 6 известны линейные размеры высоты и ширины, поэтому к ним приводится размер текстуры после коррекции перспективы.

Реалистичность любой трехмерной модели зависит не только от точности воссоздания ее геометрической формы, но и от качества нанесенных на нее текстур. Наилучший способ — это нанесение текстур, полученных из фотоизображений конкретных зданий, как было описано выше. Однако обработка больших топографических планов даже с несколькими десятками зданий требует серьезных трудозатрат. Поэтому предусмотрена открытая база данных стен зданий со всей технической информацией о месте съемки. При реконструкции сцены возможно обращение к этой базе данных за необходимой текстурой для заданного географического положения (рис. 7).

Помимо метода нанесения текстур, полученных из фотоизображений, предлагается метод автоматического текстурирования. Будут использоваться шаблонные текстуры (рис. 8), которые дают представление о типе здания. На основании распознанной служебной информации каждому зданию будет установлена та или иная текстура, если ранее для нее не была установлена текстура с фотоснимка.

Текстура здания должна состоять из нескольких однотипных шаблонных текстур (рис. 9). По вертикали шаблонная текстура будет повторяться число

Рис. 4. Мобильное приложение для получения текстур

Рис. 5. Текстура с исправленной перспективой

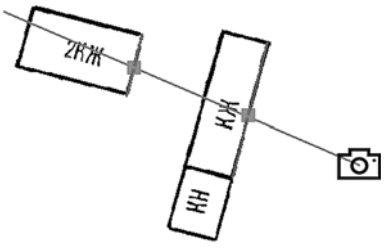

Рис. 6. Определение ближайшей стены

Рис. 7. Сцена с текстурой, полученной из фотоснимка

Рис. 8. Стандартные текстуры для жилого здания

Рис. 9. Здания со стандартными текстурами

раз, равное числу этажей у здания. Число повторений шаблонной текстуры по горизонтали вычисляется как

$$\left[\frac{\text{Высота текстуры}}{\text{Ширина текстуры}}\frac{\text{Длина стены}}{\text{Высота этажа}}\right]. \quad (1)$$

Модификации метода реконструкции

Предлагаемый метод реконструкции в качестве начальных данных использует современный топографический план. Однако метод может быть модифицирован для использования и других источников данных. Например, для реконструкции городского пространства применяли карты начала XX века (рис. 10).

Их основное отличие от современных топографических планов состоит в отсутствии информации об этажности зданий, их типах, о дополнительных объектах и т. п. Тем не менее на подобных картах есть информация о форме зданий, что позволяет получить их примерную трехмерную форму.

На карте выделяются контуры. Признаки, по которым определяется принадлежность к зданию, остаются прежними. При этом распознавание внутренней информации не происходит, потому что она не содержит данных об этажности или типе. Число этажей зданий задается реконструктором. На сцену наносятся стандартные шаблоны текстур. Результат реконструкции сцены по карте представлен на рис. 11.

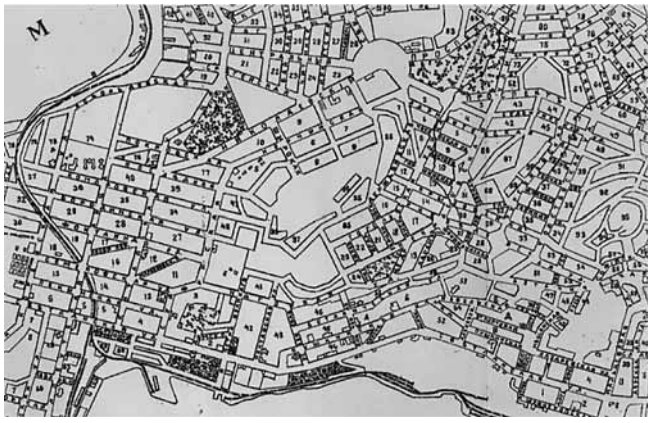

Рис. 10. Карта г. Владивостока, 1924 г.

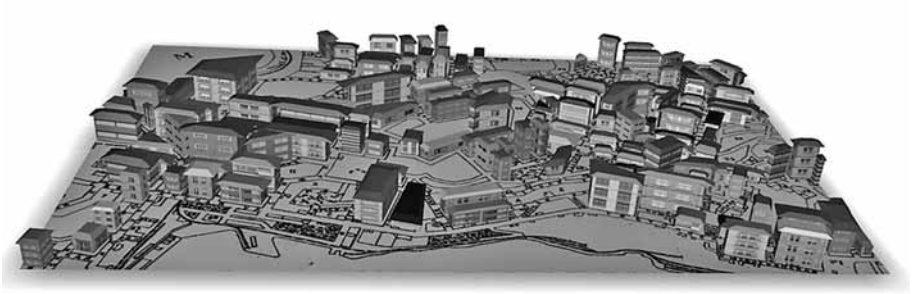

Рис. 11. Реконструированная по карте сцена

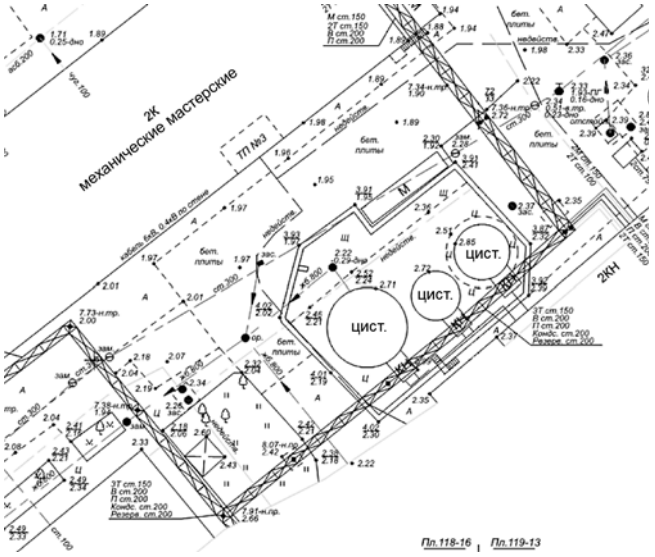

Рис. 12. Топографический план с цистернами

Рис. 13. Результат реконструкции сцены с цистернами

Метод реконструкции был также модифицирован на этапе выделения дополнительных объектов для того, чтобы определить на топографическом плане цистерны (рис. 12).

Выделенные контуры классифицировались как цистерны, если внутри них была надпись "цист." и отношение квадрата периметра контура к его площади было примерно равно $4/\pi \approx 1{,}27$. К цистернам была применена стандартная текстура (рис. 13).

Все результаты были получены с помощью разработанной программной системы (рис. 14), которая использует предложенный метод реконструкции. Система состоит из десктопного (рис. 15) и мобильного приложений.

Реконструкция (рис. 16) топографического плана с размерами 7874×7874 пикселей (1 км^2 реальной территории) был реконструирован менее чем за 6 мин. При этом из 539 зданий верно был распознан 91 %, а дополнительных объектов был распознан 81 % из 1415 штук.

Рис. 14. Схема программной системы для реконструкции

Рис. 15. Вид программного обеспечения для реконструкции по топографическому плану

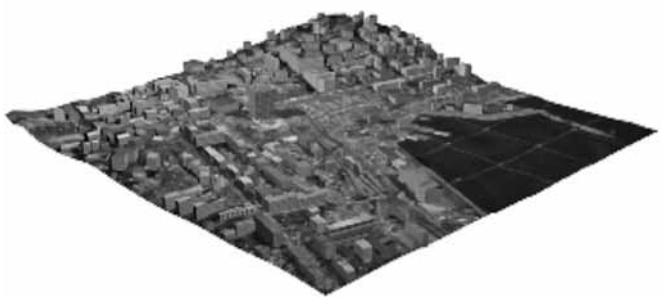

Рис. 16. Результат реконструкции сцены

Заключение

Полученные результаты подтверждают точность, надежность и гибкость метода как в поиске контуров зданий на изображениях, так и в выделении текста внутри них. Точность реконструкции была проверена на реальных топографических планах г. Владивостока, общая площадь которых составила более 6 км^2. Эти топографические планы были ранее оцифрованы, в результате чего на них присутствовали мелкие шумы и неровности краев. При этом сверка результата и исходных данных показала, что количество верно распознанных зданий (в том числе и внутренней информации) составило более 90 %.

Благодаря гибкой настройке алгоритма существует возможность расширять диапазон выделяемых объектов путем

увеличения алфавита допустимых символов на топографическом плане и параметров контура вокруг них.

Простота и эффективность предложенного метода делает его приемлемым инструментом для реконструкции трехмерных сцен реальной городской обстановки. Он может быть альтернативой методам, которые используют более дорогие и сложные технологии для получения аналогичных результатов. Полная автоматизация всех этапов реконструкции позволяет осуществлять обработку сцен огромных размеров, экономя при этом значительное количество времени. Основное время занимает работа оператора при подготовке и обработке фотоснимков городского пространства.

Список литературы

1. **Vezhnevets V., Konushin A., Ignatenko A.** Interactive image-based urban modelling // The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2007. P. 63—68.
2. **Knopp J., Prasad M., Gool L. V.** Scene cut: Class-specific object detection and segmentation in 3D scenes // Proceedings of international conference on 3D imaging, modeling, processing, visualization and transmission. 2011. P. 180—187.
3. **Чернов А. В., Чупшев Н. В.** Автоматическое распознавание контуров зданий на картографических изображениях // Компьютерная оптика. 2007. № 31. С. 101—103.
4. **Lin B. C., You R. J.** Tensor-based quality prediction for building model reconstruction from LIDAR data and topographic map // International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2012. Vol. XXXIX-B7. P. 437—441. DOI:10.5194/isprsarchives-XXXIX-B7-437-2012.
5. **Кудряшов А. П., Соловьёв И. В.** Распознавание контуров зданий на топографическом плане для реконструкций городских сцен // Вестник компьютерных и информационных технологий. 2015. № 2. С. 3—8.
6. **Кудряшов А. П., Соловьёв И. В.** Реконструкция трехмерной модели городского пространства на основе топографического плана // Вестник Амурского государственного университета. 2016. № 73. С. 58—66.
7. **Кудряшов А. П., Соловьёв И. В.** Реконструкция городской обстановки с учетом рельефа местности с использованием топографического плана и сервисов Google Maps // Информационные технологии. 2017. Том 23, № 5. С. 382—387.
8. **Harville M., Culbertson W. B., Sobel I., Gelb D., Fitzhugh A., Tanguay D.** Practical Methods for Geometric and Photometric Correction of Tiled Projector // Conference of Computer Vision and Pattern Recognition Workshops (CVPRW). 2006. IEEE, 2006. P. 5.
9. **Canny J.** A computational approach to edge detection // IEEE Transactions on Pattern Analysis and Machine Intelligence. 1986. Vol. 8, N. 6. P. 679—698.
10. **Lee C. Y.** An Algorithm for Path Connections and Its Applications // IRE Transactions on Electronic Computers. 1961. Vol. 10, N. 2. P. 364—365.

Detection of Objects on the Topographical Map for the Reconstruction of the Scene of Urban Space

A. P. Kudryashov, kudryashovA@dvo.ru, **I. V. Solovyev**, igorek.solovyev@mail.ru, Institute of Applied Mathematics of RAS, Vladivostok, 690041, RussianFederation

Corresponding author:

Kudryashov Aleksey P., Ph. D., Junior Researcher, Institute of Applied Mathematics of RAS, Vladivostok, 690041, Russian Federation
E-mail: Alkud1981@mail.ru

Received on July 11, 2019
Accepted on September 04, 2019

One of the tasks of computer graphics is the reconstruction of urban scenes. There are several methods aimed at solving this problem. One of these methods is reconstruction using a topographical map. The article proposes a method for the reconstruction of three-dimensional urban scenes using a topographical map. A modified wave algorithm is used to highlight the contours of buildings on a topographical map. The technology of texturing buildings based on photographs obtained from an open database of images is proposed. The flexibility of the reconstruction method to the initial data is shown. An information system is presented that is used to solve both the entire reconstruction process and for individual local tasks. The reconstruction of real topographical maps of Vladivostok of 1: 2000 scale is given.

Keywords: *three-dimensional reconstruction, topographical map, geographic coordinates, texturing, urban setting*

For citation:

Kudryashov A. P., Solovyev I. V. Detection of Objects on the Topographical Map for the Reconstruction of the Scene of Urban Space, *Programmnaya Ingeneria,* 2019, vol. 10, no. 11—12, pp. 464—470.

DOI: 10.17587/prin.10.464-470

References

1. **Vezhnevets V., Konushin A., Ignatenko A.** Interactive image-based urban modeling, *PIA-2007*, 2007, pp. 63—68.
2. **Knopp J., Prasad M., Gool L. V.** Scene cut: Class-specific object detection and segmentation in 3D scenes, *Proceedings of international conference on 3D imaging, modeling, processing, visualization and transmission*, 2011, pp. 180—187.
3. **Chernov A. V., Chupshev N. V.** Automatic recognition of the contours of buildings on cartographic images, *Komp'juternaja optika*, 2007, no. 31, pp. 101—103 (in Russian).
4. **Lin B. C., You R. J.** Tensor-based quality prediction for building model reconstruction from LIDAR data and topographic map, *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2012, vol. XXXIX-B7, pp. 437—441.
5. **Kudryashov A. P., Solovyov I. V.** Topographic recognition of the contours of buildings for reconstruction of urban scenes, *Vestnik komp'yuternykh i informatsionnykh tekhnologiy*, 2015, no. 2, pp. 3—8 (in Russian).
6. **Kudryashov A. P., Solovyov I. V.** Reconstruction of a three-dimensional model of urban space based on a topographic plan, *Vestnik Amurskogo gosudarstvennogo universiteta*, 2016, no. 73, pp. 58—66 (in Russian).
7. **Kudryashov A. P., Solovyov I. V.** Reconstruction of the urban environment taking into account the topography using the topographic plan and Google Maps services, *Informatsionnye tekhnologii*, 2017, vol. 23, no. 5, pp. 382—387 (in Russian).
8. **Harville M., Culbertson W. B., Sobel I., Gelb D., Fitzhugh A., Tanguay D.** Practical Methods for Geometric and Photometric Correction of Tiled Projector, *CVPR Workshops*, 2006, p. 5.
9. **Canny J.** A computational approach to edge detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986, vol. 8, no. 6, pp. 679—698.
10. **Lee C. Y.** An Algorithm for Path Connections and Its Applications, *IRE Transactions on Electronic Computers*, 1961, vol. 10, no. 2, pp. 364—365.

УДК 004.021+004.82+004.89 DOI: 10.17587/prin.10.471-479

А. О. Корней, аспирант, korney.alena@yandex.ru,
Е. Н. Крючкова, канд. физ.-мат. наук, доц., проф. кафедры, kruchkova_elena@mail.ru,
Алтайский государственный технический университет им. И. И. Ползунова, г. Барнаул

Применение адаптируемых обобщенных словарей в задачах аспектно-ориентированного анализа тональности

Рассмотрены вопросы адаптации систем анализа тональности к предметным областям. Предложена система анализа тональности коротких текстов на основе тоновых словарей. В систему внедрен семантический граф русского языка, представляющий собой источник обобщенных знаний. Граф используется для расширения тонового лексикона. Изучена устойчивость системы к изменению предметной области. Предложен метод адаптации лексикона и графа к новым предметным областям, не требующий полного переобучения системы.

Ключевые слова: *анализ тональности, аспектно-ориентированный анализ тональности, тоновый лексикон, семантический граф русского языка, математическая модель оценки тональности, метод оценки тональности, категоризация текстов, адаптация словарей к предметной области, извлечение знаний*

Введение

В современном мире неотъемлемой частью повседневной жизни стал Интернет. Множество факторов приводит к смещению социального фокуса в сторону онлайн-коммуникации. Все более активно пользователи общаются на интернет-площадках, делятся личным мнением о различных товарах, услугах, событиях. Публикуемые отзывы, рецензии, мнения содержат в себе колоссальный объем информации. Исходя из пользовательских предпочтений, представители бизнеса могут принимать взвешенные стратегические решения, связанные с развитием услуги или продукта. Потребители, в свою очередь, тщательно изучают информацию и все чаще на ее основе делают окончательный выбор при покупке товаров, при выборе отелей, ресторанов, авиакомпаний и т. д. Публикуемые различными СМИ тексты могут влиять на репутацию крупных некоммерческих организаций, государственных предприятий и учреждений [1].

Для автоматического извлечения субъективной информации из текстов используется анализ тональности. Ранние работы, например [2, 3], в основном посвящены определению тональности всего документа в терминах бинарной либо тернарной шкалы.

Такой подход непригоден для работы с отзывами. Это связано с тем, что каждый продукт или услуга — совокупность отдельных параметров, аспектов, каждый из которых может быть оценен по-своему. Например, в предложении *"Экран великолепный, но батарея садится очень быстро"* автор выражает позитивное мнение об экране, но плохо отзывается об аккумуляторе (батарее). В работе [4] были представлены одни из первых систем аспектно-ориентированного анализа тональности (АОАТ). Подобные системы осуществляют поиск в тексте отдельных аспектов исследуемого продукта, а затем извлекают мнения относительно этих аспектов.

В АОАТ принято выделять следующие подзадачи: выделение аспектных терминов *(aspect term extraction,* ATE*)*; определение тональности аспектов *(aspect term polarity,* ATP*)*; выявление аспектных категорий (*aspect category detection,* ACD); определение тональности категорий *(aspect category polarity,* ACP*)*. В некоторых работах [5, 6] выделение категорий и определение их полярности рассматриваются как единая подзадача.

Извлечение аспектных терминов — одна из наиболее хорошо формализуемых подзадач АОАТ. На настоящее время распространены следующие подходы: на основе машинного обучения с учителем [7, 8]; на основе частотности [9]; на основе методов тематического моделирования [10, 11] и др. В работах [12, 13] отмечено, что наивысшие результаты показывают методы, основанные на семантической близости или оперирующие кластерами слов, вместо индивидуальных понятий.

Для случая, когда аспектный термин не содержится в тексте явно, но категория упоминается через тематически связанные слова, не подходят методы, которые требуют явного присутствия термина в тексте. Задача определения категорий в общем случае сводится к *multi-label*-классификации предложений. В работе [14] представлен подход к определению категорий, основанный на z-score [15]. В работе [16] рассмотрено применение машинного обучения для категоризации. Большую роль играет определение семантического сходства, близости слов, а также ча-

стотности совместного употребления слов в пределах аспектной категории.

Важная цель АОАТ — оценить тональность фрагмента текста, контекстно связанного с категорией/аспектом. Для выделения контекста используются различные методики — от контекстных окон [6] до CRF [18] и нейронных сетей [19]. В случае, когда никакой аспектный термин не присутствует в тексте явно, под контекстом мнения подразумевают предложение целиком.

Помимо непосредственного анализа мнений, предпринимаются попытки извлекать из отзывов связанные с аспектами предложения, подсказки, советы [20]. Появление подобных задач указывает на то, что в современном мире значение имеют не только статистические показатели, но и содержимое, смысл пользовательских отзывов [21].

Тоновые словари в аспектно-ориентированном анализе

Построение качественных АОАТ-систем невозможно без привязки к конкретной предметной области. На настоящее время не существует методов построения универсальных систем ни на базе словарей, ни на базе машинного обучения. В связи с этим обстоятельством остро стоит вопрос об эффективной адаптации уже существующих систем к новым предметным областям. Такая адаптация может существенно снизить затраты на разработку, как временные, так и финансовые.

Основные идеи при построении гибких адаптируемых систем сводятся к одному из следующих двух приемов:

1) отделить обработку базовой информации от информации, обладающей спецификой для предметной области;

2) сформировать эффективное представление данных, которое позволило бы "проецировать" новый домен на уже известный.

Работы [22, 23] описывают различные пути адаптации — от простого применения тезаурусов до сложных подходов к классификации. Большое число работ посвящено эффективному формированию тоновых лексиконов [24, 25], так как лексиконы демонстрируют большую гибкость и адаптируемость в сравнении с системами на основе машинного обучения.

Авторы настоящей работы поставили перед собой задачу построить адаптируемую систему анализа тональности на основе обобщенных тоновых словарей и семантического графа русского языка. Обобщенные тоновые словари построены на основе статистического анализа корпуса русскоязычных твитов, а семантический граф — на основе общелингвистических словарей русского языка. Таким образом, каркас системы содержит независимые от предметной области объективные и субъективные данные о словах и выражениях русского языка.

Целями исследования, результаты которого представлены в настоящей работе, являются:

- оценка границ применимости обобщенных тоновых словарей для задач АОАТ;
- изучение возможностей использования обобщенных знаний о мире для уточнения контекста;
- разработка методов адаптации словарей к предметной области.

Анализ тональности на основе обобщенных словарей

В настоящей работе рассматривается система, в основе которой лежат обобщенные тоновые словари. Представленные словари могут быть с небольшими трудозатратами адаптированы для применения в отдельных предметных областях.

Для повышения гибкости системы авторами было принято решение отойти от классических дискретных моделей оценки тональности. Одним из начальных этапов усложнения бинарной модели является перевод ее из дискретной формы в непрерывную, где каждая лексическая единица может иметь тональность в отрезке $[-1; 1]$.

Статистический анализ обучающей выборки

В рамках настоящей работы для построения тоновых словарей была использована обучающая выборка [26], размеченная в терминах бинарной шкалы. Корпус состоит из 114,911 положительных и 111,923 отрицательных записей. Каждая запись представляет собой twitter-пост (твит) длиной как минимум 40 символов, содержащий только один тип эмоций — положительный или отрицательный.

Качественный и количественный анализ лексики обучающей выборки (рис. 1, см. четвертую сторону обложки) показал, что с точки зрения частеречевой принадлежности лексикон выборки с небольшими отклонениями совпадает с лексиконом G из работы [27], автоматически извлеченным авторами из общелингвистических словарей. Это позволяет оценить как надежные полученные статистические данные обработки twitter-поста.

Математическая модель оценки тональности

После проверки структуры тонового лексикона были вычислены числовые характеристики отдельных слов. В словарь включались канонические формы слов русского языка без учета отрицаний, кроме стоп-слов и служебных частей речи. Пусть f_{pw} — частота вхождения слова w в корпус позитивных twitter-постов, а f_{nw} — частота вхождения слова w в корпус негативных twitter-постов. Рассмотрим следующие величины:

- тональность слова w: $s_w = \dfrac{f_{pw} - f_{nw}}{f_{pw} + f_{nw}}$,
- нормированная частота появления слова w в словаре W: $o_w = \dfrac{f_{pw} + f_{nw}}{\max\limits_{l \in W}\left(f_{pl} + f_{nl}\right)}$.

Величина o_w характеризует степень принадлежности слова w к числу общеупотребимых слов. Каж-

дому слову поставлена в соответствие двумерная оценка $t_w = (s_w, o_w)$. Экспериментально установлен порог доверия статистическим данным на уровне десяти вхождений в корпус положительных или отрицательных twitter-постов. При первичном анализе отфильтрованных данных на основе здравого смысла ожидалось, что большим значениям o_w соответствуют малые значения s_w и наоборот. Это соответствует природе естественного языка — слова, употребляемые очень часто, как правило, не имеют ярко выраженной эмоциональной окраски и формируют "нейтральный" каркас языка. Рис. 2 (см. четвертую сторону обложки) демонстрирует графическое представление оценок отдельных слов, а также распределение словарных единиц по шкале эмоциональной окраски, которые вполне соответствуют теоретическим предположениям.

Включение в словари простых семантических конструкций

В терминах бинарной шкалы частица "не", как правило, изменяет эмоциональную окраску слова или группы слов с позитивной на негативную и наоборот. Однако при работе с непрерывной моделью оценка такого утверждения не является однозначно верной. Это значит, что, имея в словаре слово w с оценкой $t_w = (s_w, o_w)$, нет оснований принять $t_{(\text{НЕ})w} = (-s_w, o_w)$ в качестве оценки тональности конструкции "(НЕ) *w*". Кроме отрицаний, в более тонкой обработке и интерпретации нуждаются устойчивые речевые обороты, а также связные смысловые конструкции, в составе которых более одного значимого слова.

Для повышения точности при оценке тональности текстов авторами был построен тоновый лексикон на основе данных поверхностного семантического анализа (синтаксико-семантическое представление предложения в виде семантической сети). В качестве инструментария для обработки твитов был использован семантический парсер RML[1], а размер обучающей выборки составил 28 000 позитивных и 28 000 негативных твитов.

В состав словаря вошли перечисленные далее лексико-семантические единицы.

- Униграммы — слова значимых частей речи, за исключением стоп-слов, приведенные к канонической форме. В словарь включались агрегированные отрицания, полученные от семантического парсера, в виде *"(НЕ)<слово>"*.
- Биграммы. Рассматривались сочетания значимых слов, за исключением стоп-слов, связанные семантическими отношениями SUB, OBJ, CONTEN и PROPERT в выходном графе парсера RML. Данные типы связей отобраны на основе экспериментов как наиболее часто встречающиеся и осмысленные.

Статистические показатели для униграмм и биграмм подсчитывались независимо, затем подвергались фильтрации. В табл. 1 и 2 приведены данные выборочного анализа содержимого словарей.

[1] Сайт рабочей группы "Автоматическая обработка текста" http://aot.ru

Таблица 1

Сравнение эмоциональной окраски слов и их отрицаний

Каноническая форма слова	s_w	$s_{(\text{НЕ})w}$
выспаться	0,296	−0,736
хотеть	−0,273	−0,636
уметь	0,148	−0,511
разговаривать	0,341	−0,294
бояться	−0,394	0,394

Таблица 2

Сравнение эмоциональной окраски униграм и биграмм

Слово и его оценка	Список биграмм (с оценками)	Характер изменения оценки
бояться (−0,394)	я бояться (−0,632) бояться что (−0,667)	*Усиление*
любить (0,471)	сильно любить (0,833) я любить (0,515) любить ты (0,6)	*Усиление*
решить (0,208)	я решить (0,435)	*Усиление*
жить (−0,053)	я жить (−0,25)	*Уточнение*
год (0,039)	новый год (0,112) этот год (−0,219) каждый год (0,222)	*Уточнение*

Значения для эмоциональной окраски, полученные на уровне поверхностного семантического анализа, оцениваются как более адекватные и достоверные. На основе полученных данных был сделан вывод, что двумерная оценка $t = (s, o)$ пригодна для вычисления тональности коротких текстовых фрагментов.

Семантический граф как источник базовых знаний о мире

Интернет-язык обладает своей спецификой, к которой относятся: отсутствие слов высокого стиля, общая простота конструкций, обилие ошибок и жаргонных терминов. В связи с этим обстоятельством убедительные статистические данные можно собрать для ограниченного лексикона. Одним из способов расширения словарного запаса является использование словарей синонимов, тезаурусов и более сложных лексических ресурсов, основанных на семантических связях. Широко известны такие проекты, как WordNetAffect, SenticNet и др. Существующие ресурсы имеют ряд положительных характеристик. Однако лежащие в их основе математические модели не вполне подходят для решения задач, поставленных авторами. Например, *синсеты* (от англ. *synset* — набор синонимов) дочерних проектов WordNet, связанные семантическими отношениями, не позволяют отражать нюансы для отдельных пар слов. Пространство эмоций *Hourglass*

of Emotions [28] по сложности многократно превосходит используемые авторами модели.

В связи с тем, что одним из ключевых требований к системе является простота реализации, было принято решение использовать разработанный авторами лексикон *G*. Лексикон представляет собой ориентированный взвешенный граф $G = (W, E)$, вершинами которого являются канонические формы русскоязычных слов, а дугами — направленные взвешенные связи трех типов — ассоциации ($Assoc(w_a, w_b)$), синонимии ($Syn(w_a, w_b)$) и определения ($Def(w_a, w_b)$). Связи представляют собой нечеткие отношения. Присутствие дуги в графе означает наличие семантической связи между словами в общеупотребительном контексте. Выводы о связи между словами строятся на основании словарных статей, которые можно считать достоверными и адекватными. Тип связи зависит от положения слова в дереве семантического разбора и типа словарной статьи (синонимы обычно приводятся списком и не требуют семантического разбора). Граф позволяет оценивать меру семантической близости слов в русском языке. В случае, когда слова связаны напрямую, в качестве меры близости принимается максимальный вес из имеющихся отношений. Если же прямая связь отсутствует, необходимо рассчитать максимальное произведение близостей слов на пути между заданными понятиями по следующей формуле:

$$Dict(w_a, w_b) =$$

$$= \begin{cases} \max\left(Assoc(w_a, w_b),\ Def(w_a, w_b),\ Syn(w_a, w_b)\right), \\ \text{если } (w_a, w_b) \in E \\ \max_{P_k} \prod\limits_{(w_i, w_j) \in P_k} Dict(w_i, w_j), \\ \text{где } P_k \text{ — пусть в графе между } w_a, w_b, \\ \text{если } (w_a, w_b) \notin E. \end{cases}$$

При работе с лингвистическими словарями входные данные не так остро нуждаются в семантической или статистической верификации. Причина в том, что словарные статьи имеют малый размер и строгую формальную структуру, а также содержат данные, которые обязательно так или иначе связаны с определяемым словом. Таким образом, при обработке словарей отношениям синонимии, ассоциации и определения присваиваются базовые веса независимо от частотности (частотность имеет значение при работе с предметной областью и используется совместно с базовым весом). Структура графа и аппарат для вычисления близости позволяют изменять пороговые значения и базовые веса отношений между словами и формировать для слов семантические окрестности различного наполнения. Граф содержит обобщенные знания о мире и на протяжении последних лет зарекомендовал себя как достоверный и надежный источник информации в задачах поиска и классификации [29, 30].

Вычисление тональности коротких текстов

При вычислении итоговой тональности короткого текста различные слова и биграммы, входящие в его состав, должны учитываться с различным весом. По итогам анализа статистических данных были сформулированы следующие требования к весовой функции:

- слова с "нейтральной" эмоциональной окраской (s близко к нулю), либо с минимальным числом вхождений имеют минимальный вес;
- чем чаще употребляется слово (биграмма) и чем "ярче" эмоциональная окраска, тем выше вес;
- весовая функция должна быть положительной для любых (s, o).

Характер расположения оценок на плоскости в сочетании с перечисленными требованиями позволяют строить весовую функцию не для всего лексикона, а покрывать лишь заданный процент словарного запаса, отбросив наиболее нейтральные и часто употребляемые слова как незначимые.

По результатам экспериментов была выбрана симметричная весовая функция, основанная на кривой Гаусса, ключевыми параметрами которой являются фактор роста g, функция роста $\varphi(g, d)$ и степень покрытия тонового лексикона T (которому соответствует пороговое значение o_T).

Рассмотрим порядок вычисления веса отдельного слова. Пусть имеется слово w с известной оценкой $t_w = (s_w, o_w)$, задано значение T и соответствующие ему векторы $(1, o_T)$ и $(-1, o_T)$. Тогда R — расстояние от t_w до соответствующего по знаку s_w вектора, а d — расстояние от центра координат до точки пересечения направляющего вектора с нормалью, опущенной из точки t_w. Значение r, влияющее на поведение гауссианы, вычисляется на основе R в каждой точке. При этом значение σ, также влияющее на ширину и высоту колокола гауссианы, задается однократно и остается неизменным для всей поверхности. Тогда вес слова можно рассчитать по формуле $M_w = \dfrac{\varphi(g, d)}{\sigma\sqrt{2\pi}} e^{-\frac{r^2}{2\sigma^2}}$.

Оценка тональности короткого текста складывается из оценок отдельных лексических единиц в его составе. В первую очередь запрос адресуется тоновым словарям (на уровне семантики — словарю биграмм, затем униграмм). При отсутствии данных в тоновых словарях вычисляется неявная оценка тональности по семантическому графу по формуле

$$\bar{t}_l = \frac{\sum\limits_{w \in N} \bar{t}_w M_w}{\sum\limits_{w \in N} M_w},$$

где N — пересечение тоновых словарей с семантической окрестностью слова l; M_w — вес, а $\bar{t}_w = (s_w, o_w)$ — двумерная оценка тональности слова w из окрестности.

Итоговая тональность текста T вычисляется по формуле

$$S_T = \frac{\sum\limits_{w \in T} s_w M_w}{\sum\limits_{w \in T} M_w}.$$

Результаты экспериментов

Эксперименты по определению полярности проводили для 200 позитивных и 200 негативных твитов, случайно выбранных из имеющегося набора. Под полярностью в рамках эксперимента следует понимать знак вычисленного значения s_w. Согласно математической модели знаку "минус" соответствует негативная тональность, знаку "плюс" — позитивная тональность. Для построения словарей семантического уровня было проанализировано 56 000 твитов, на лексическом уровне — существенно больше, около 220 000, что составляет почти весь имеющийся набор данных. Тем не менее эксперименты, результаты которых описаны в табл. 3, показывают, что использование семантической информации даже при меньших исходных данных дает существенный прирост точности классификации.

Таблица 3

Сравнение точности классификации при использовании различных словарей

Словари	Точность, %	
	Для постов с позитивной окраской	Для постов с негативной окраской
Лексические	69	60
Семантические	**73**	**69**

Определение границ устойчивости словарей к предметной области

В задачах аспектно-ориентированного анализа тональности огромное значение имеет принадлежность текста к той или иной предметной области (домену). Предметная область во многом определяет семантику и эмоциональную окраску отдельных слов и устойчивых выражений. Однако некоторые конструкции оказываются устойчивыми к изменению предметной области (например, оценочные прилагательные "хорошо", "ужасно", "восхитительно" и им подобные сохраняют смысл и тональность в большинстве доменов). Авторами были проведены эксперименты с целью определить сильные и слабые стороны обобщенных словарей, сформировать эффективную стратегию последующей адаптации. В ходе экспериментов решались задачи определения *категории аспекта* и вычисления *тональности категории.*

Категоризация текстов в условиях реальных данных

Для исследования применимости обобщенных тоновых словарей в задачах аспектно-ориентированного анализа тональности авторами был выбран набор русскоязычных отзывов о ресторанах, подготовленный и представленный в рамках семинара *SemEval-2016, Task 5, Subtask2 (Text-levelABSA)*[2]. Обозначенный набор содержит 300 отзывов о ресторанах на русском языке в обучающей выборке и 103 отзыва — в тестовой выборке. Средняя длина отзыва превышает 10 предложений. Каждому отзыву поставлен в соответствие набор меток вида *<category = "Entity#Attribute", polarity = "polarity_class">*. Список сущностей включает в себя шесть возможных значений, список атрибутов — пять. Несмотря на существование 30 возможных комбинаций, реально в наборе представлены лишь 12 из них. Для наиболее редких категорий доля релевантных отзывов не превышает 7...10 %. Дисбаланс, связанный с частотой упоминания категорий, вероятнее всего, отражает специфику реальных данных. Тональность представлена тремя реальными классами — *positive, negative, neutral* и особым классом *conflict.* Последний класс соответствует ситуации, в которой в отношении категории высказано и позитивное, и негативное мнение.

Для категоризации использовались словари, каждому слову ставилась в соответствие метрика *z-score* [14]. Однако неоднородность данных не позволяет сравнивать абсолютные значения *z-score* из различных категорий в случае неоднозначности. Вместо этого для категоризации текстов использовали *z-score rating*, оценивающий, какой процент лексикона категории имеет метрику ниже, чем данное слово. Эксперименты проводили на уровне документов (отзывов). Набор меток, вычисленных системой, сравнивали с реальными данными. Пусть m — набор вычисленных меток; n — набор реальных меток, тогда $q = m \cap n$, $P = \frac{|q|}{|m|}$, $R = \frac{|q|}{|n|}$ для каждого документа.

С применением *z-score rating* удалось добиться средней точности 66 % при пороговом значении метрики 97,5 %, т. е. лишь 2,5 % лексикона категорий использовалось для разметки тестовых данных. В среднем 4 из 6 категорий, присущих отзыву, выявляются верно. Однако показатели точности для отдельных категорий имеют существенный разброс, который напрямую связан с частотой выхождения категории в корпус отзывов. Более подробно результаты представлены в табл. 4, а также в работе [31].

В представленных экспериментах для категоризации текстов использовались лишь те обладающие спецификой знания, которые удалось извлечь из обучающего

Таблица 4

Показатели точности метода категоризации в зависимости от порога *z-score rating*

Порог *z-score rating*	Точность, %	Полнота, %	F-мера, %
0,875	43,5	99,6	60,5
0,900	44,6	99,6	61,6
0,925	48,4	99,5	65,2
0,950	51,3	97,3	67,2
0,975	**66,4**	**90,0**	**76,4**

[2] SemEval-2016 Task 5: http://alt.qcri.org/semeval2016/task5/

набора отзывов. Однако в реальной ситуации человек воспринимает тексты иначе, через призму здравого смысла и обобщенных знаний. Поэтому человеку намного проще ориентироваться в аспектных категориях даже в новой предметной области. Таким образом, имеет смысл включать в состав систем источники общих знаний наравне с источниками знаний, имеющих специфику. Пример значительного улучшения качества АОАТ за счет обобщенных знаний показан в работе [32].

Определение тональности текстов

Для эксперимента был выбран тестовый датасет SemEval — 2016 из 103 отзывов. Каждый отзыв в обязательном порядке имеет метку категории *restaurant#general*, которая по сути является маркером принадлежности домену. Из датасета были удалены все отзывы с нейтральной и конфликтной тональностью по данной категории, и затем оставшиеся 84 (69 положительных, 15 отрицательных) отзыва были проанализированы:

- на уровне текста без разбиения на предложения (взвешивание неупорядоченного набора биграмм и униграмм);
- с отдельным анализом каждого предложения.

Результаты оказались практически идентичными. При использовании обобщенного тонового лексикона оценка смещается (в данном случае — к позитиву):

- верно определены все 69 позитивных отзывов, ложно положительно — 12;
- верно определены только 3 из 15 отрицательных отзывов.

Смещение оценки в сторону позитивных значений при любом алгоритме оценки свидетельствует о потере некоторого количества контекстно-зависимой лексики, которая вне домена не несет негативной эмоциональной окраски. Таким образом, подтверждена необходимость адаптации словарей к домену путем включения в них информации, характеризующей специфику.

Метод адаптации к предметной области

В современных системах обработки естественно-языковых текстов хорошо зарекомендовали себя подходы на основе инструментальных средств дистрибутивной семантики. К их числу относятся, например, GloVe, модели Skip-gram и CBOW в составе TensorFlow, русскоязычный сервис RusVectrs [33]. В последнее время появляется все больше исследований, связанных с применением векторных представлений в классических задачах анализа тональности и в АОАТ-системах [8, 34].

Семантический граф G по своему содержанию отвечает основополагающим идеям, на которых строятся инструментальные средства дистрибутивной семантики. Математическая модель позволяет адаптировать граф к различным предметным областям, при этом базовые знания о них сохраняются. Структура связей, предложенная авторами при создании графа, позволяет проводить поиск синонимов. Глубина поиска при этом может быть выбрана в зависимости от условий задачи.

Анализ структуры графа, сравнение информационной выдачи с популярными инструментами дистрибутивной семантики дают основания для вывода о том, что предложенный граф может использоваться в АОАТ-системах как источник семантической информации. Основная идея семантической адаптации состоит в итеративном увеличении семантических весов домен-специфичных связей в графе. Связи могут быть извлечены из соответствующих текстов, а изменение весов приведет к изменению извлекаемой информации в сторону увеличения релевантности. Кроме того, такой подход потенциально пригоден для формирования тематических кластеров и расширения словарей аспектных терминов на основе синонимии.

Структура словарей, предложенных авторами, не противоречит идее постоянного дообучения в режиме реального времени, что позволит непрерывно повышать качество обработки текстов. Обобщенная

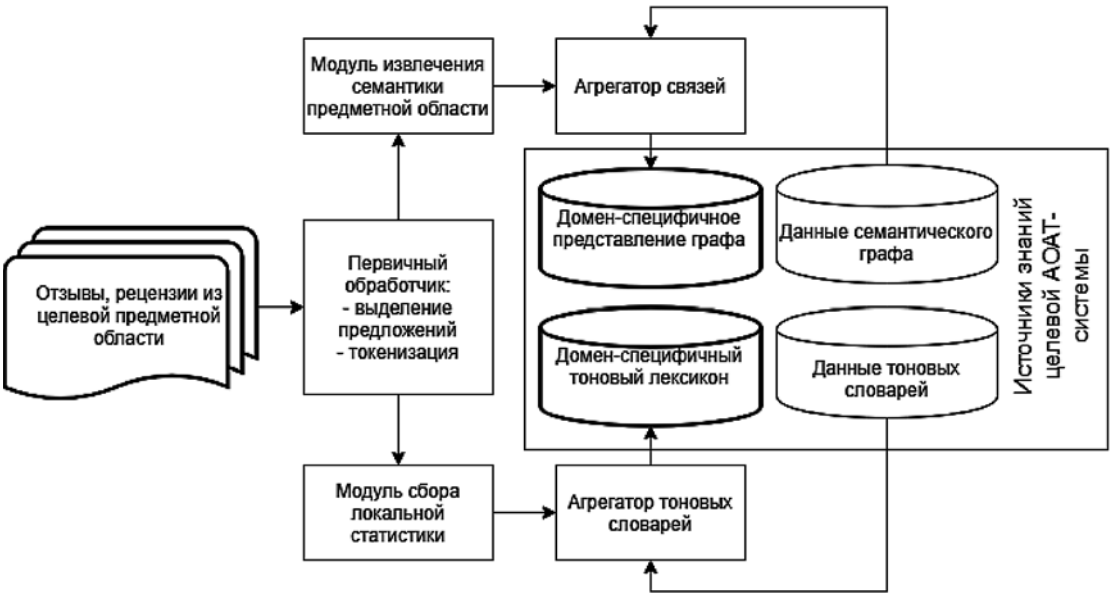

Рис. 3. Схема работы системы адаптации лексиконов

схема предлагаемой системы адаптации словарей представлена на рис. 3.

Методика адаптации предполагает точечное, локальное дополнение существующих базовых словарей. При работе с семантическим графом информация о частоте совместного употребления слов в предметной области хранится отдельно от основных знаний системы о мире. Комбинация частоты, характеризующей специфику, и базового общелингвистического веса позволяет рассчитывать вес связи, характерный для предметной области, и при этом в случае недостатка информации в текстах, обладающих спецификой, извлекать обобщенные данные из базового графа. При совместной работе с базовым и обладающими спецификой тоновыми словарями возможно оценить устойчивость конструкций к изменению предметной области.

Заключение

Проблемные вопросы, возникающие при построении систем анализа тональности, в большинстве своем связаны с природой естественного языка и его разнообразием. В последние годы исследователи, занимаясь различными вопросами в данной области, приходят к общим выводам — их можно частично или полностью преодолеть, если задействовать семантический анализ и источника знаний, в том числе обобщенных. Это справедливо и для категоризации, и для выявления сарказма, и для обработки отрицаний, и для раскрытия контекстных связей, и для верного определения тональности.

Метод, предложенный в данной работе, учитывает важность семантических подходов и инструментальных средств. Использование семантического графа в сочетании с тоновыми словарями имеет ряд преимуществ. К их числу относятся:

- единый согласованный источник информации для семантического и тонового анализа (гетерогенные настраиваемые связи позволяют извлекать нужные данные без изменения структуры графа);
- возможности синхронной адаптации;
- возможность постоянного дообучения системы на основе анализируемых текстов.

Основные недостатки предложенной системы связаны с необходимостью поверхностного семантического анализа текстов, который является наиболее затратным по времени процессом.

Список литературы

1. **Васенин В. А., Роганов В. А., Дзабраев М. Д.** Методы автоматизированного анализа тональности текстов в средствах массовой информации // Программная инженерия. 2016. Т. 7, № 8. С. 360—372.
2. **Turney P.** Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews // Proceedings of the Association for Computational Linguistics, 2002. P. 417—424.
3. **Pang B., Lee L., Vaithyanathan S.** Thumbs up? Sentiment Classification using Machine Learning Techniques // Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2002. P. 79—86.
4. **Thet T. T., Na J. C., Khoo C. S. G.** Aspect-based sentiment analysis of movie reviews on discussion boards // Journal of Information Science archive. 2010. Vol. 36, No. 6. P. 823—848.
5. **Xue W. W., Li T. E.** Aspect Based Sentiment Analysis with Gated Convolutional Networks // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers). Melbourne, Australia, 2018. P. 2514—2523.
6. **Mashkin D.** Extracting Aspects, Category and Sentiment of Aspects in Russian User Reviews in Restaurants Domain // Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialogue 2016". Moscow, 2016, 8 p. URL: http://www.dialog-21.ru/media/3453/mashkindo.pdf
7. **Ivanov V., Tutubalina E., Mingazov N., Alimova I.** Extracting Aspects, Sentiment and Categories of Aspects in User Reviews about Restaurants and Cars // Proceedings of the 21st International Conference on Computational Linguistics Dialog-2015, 2015. Vol. 2. P. 46—57.
8. **Blinov P. D., Kotelnikov E. V.** Semantic Similarity for Aspect-Based Sentiment Analysis // Proceedings of the 21st International Conference on Computational Linguistics Dialog-2015, 2015. Vol. 2. P. 36—45.
9. **Popescu A. M., Nguyen B., Etzioni O.** OPINE: Extracting product features and opinions from reviews // Proceedings of HLT/EMNLP on interactive demonstrations, 2005. P. 32—33.
10. **Mukherjee A., Liu B.** Aspect extraction through semi-supervised modeling // Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, 2012. Vol. 1. P. 339—348.
11. **Titov I., McDonald R.** Modeling online reviews with multi-grain topic models // Proceedings of the 17th international conference on World Wide Web. ACM, 2008. P. 111—120.
12. **Андрианов И. А., Майоров В. Д., Турдаков Д. Ю.** Современные методы аспектно-ориентированного анализа эмоциональной окраски // Труды ИСП РАН. 2016. Т. 27, № 5. С. 5—22. DOI: 10.15514/ISPRAS-2015-27(5)-1.
13. **Машкин Д. О., Котельников Е. В.** Извлечение аспектных терминов на основе условных случайных полей и векторных представлений слов // Труды ИСП РАН. 2016. Т. 28, № 6. С. 223—240.
14. **Hamdan H., Bellot P., Bechet F.** Super-vised Methods for Aspect-Based Sentiment Analysis // Proceedings of the Eighth International Workshop on Semantic Evaluation (SemEval 2014), 2014. P. 596—600.
15. **Zubareva O., Savoy J.** Opinion Detection by Combining Machine Learning and Linguistic tools // Proceedings of NTCIR-8 Workshop Meeting, Tokyo, Japan, June 15—18, 2010. P. 221—227.
16. **Schouten K., van der Weijde O., Frasincar F., Dekker R.** Supervised and Unsupervised Aspect Category Detection for Sentiment Analysis with Co-occurrence Data // IEEE Transactions on Cybernetics. 2018. Vol. 48, Issue 4. P. 1263—1275. DOI:10.1109/tcyb.2017.2688801.
17. **San Vicente I. N., Saralegi X., Agerri R.** Elixa: A modular and flexible ABSA platform // Proceedings of the 9th International Workshop on Semantic Evaluation. Association for Computational Linguistics, Denver, Colorado (June 2015), 2015. P. 748—752.
18. **Toh Z., Wang W.** DLIREC: Aspect Term Extraction and Term Polarity Classification System // Proceedings of the 8th International Workshop on Semantic Evaluation, 2014. P. 235—240.
19. **Jebbara S., Cimiano P.** Aspect-Based Sentiment Analysis Using a Two-Step Neural Network Architecture // Proceedings of the twenty-second European conference on artificial intelligence, 2016. Vol. 641. P. 153—167.
20. **SemEval 2019** Task 9 — SubTask A — Suggestion Mining from Online Reviews and Forums, URL: https://competitions.codalab.org/competitions/19955.
21. **Negi S., Asooja K., Mehrotra S., Buitelaar P.** A Study of Suggestions in Opinionated Texts and their Automatic Detection // Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics, Berlin, Germany, 2016. P. 170—178.
22. **Bollegala D., Weir D., Carroll J.** Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification // Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, Oregon, 2011. P. 132—141.
23. **Franco-Salvador M., Cruz F. L., Troyano J. A., Rosso P.** Cross-domain polarity classification using a knowledge-enhanced meta-classifier // KnowledgeBased Systems. 2015. Vol. 86. P. 46—56.
24. **Dubatovka A., Kurochkin Yu., Mikhailova E.** Automatic Generation of the Domain-Specific Sentiment Russian Dictionaries // Computational Linguistics and Intellectual Technologies: Proceedings of the Annual International Conference "Dialogue". 2016. Issue 15. P. 146—158.

25. **Labille K., Gauch S., Alfarhood S.** Creating Domain-Specific Sentiment Lexicons via Text Mining // In Proceedings of WISDOM 2017: 6th KDD Workshop on Issues of Sentiment Discovery and Opinion Mining, At Halifax, Nova Scotia, Canada. URL: https://sentic.net/wisdom2017labille.pdf.
26. **Рубцова Ю. В.** Построение корпуса текстов для настройки тонового классификатора // Программные продукты и системы. 2015. № 1 (109). С. 72—78.
27. **Крайванова В. А., Крючкова Е. Н., Кротова А. О.** Математическая модель естественного языка в задачах нечеткого ассоциативного поиска // Материалы XIV Международной конференции "Речь и компьютер" (SPECOM'2011), 2011. С. 402—406.
28. **Cambria E., Livingstone A., Hissain A.** The hourglass of emotions // Proceedings of the 2011 International conference on Cognitive Behavioural Systems, February 21—26, 2011, Dresden, Germany, 2011. P. 144—157.
29. **Казаков М. Г., Крючкова Е. Н.** Классификация сложных изображений на основе семантического графа понятий // Прикладная информатика. 2014. № 6 (54). С. 79—89.
30. **Савченко В.** Алгоритм семантического поиска в больших текстовых коллекциях // Supplementary Proceedings of the 3rd International Conference on Analysis of Images, Social Networks and Texts (AIST'2014), 2014. P. 161—166.
31. **Корней А. О., Крючкова Е. Н.** Проблемы эффективности аспектного анализа в условиях несбалансированной обучающей выборки // Высокопроизводительные вычислительные системы и технологии. 2019. Т. 3, № 1, С. 161—165.
32. **Ma Y., Peng H., Cambria E.** Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM // AAAI, 2018. P. 5876—5883.
33. **Kutuzov A., Kuzmenko E.** WebVectors: A Toolkit for Building Web Interfaces for Vector Semantic Models // Analysis of Images, Social Networks and Texts. AIST 2016. Communications in Computer and Information Science, 2016. P. 155—161.
34. **Alghunaim A., Mohtarami M., Cyphers S., Glass J.** A Vector Space Approach for Aspect Based Sentiment Analysis // Proceedings of NAACL-HLT 2015, Denver, Colorado, May 31 — June 5, 2015. P. 116—122.

Application of Adaptable Generalized Lexicons in Aspect-Based Sentiment Analysis

A. O. Korney, korney.alena@yandex.ru, **E. N. Kryuchkova**, kruchkova_elena@mail.ru,
Polzunov Altai State Technical University, Barnaul, 656038, Russian Federation

Corresponding author:

Korney Alena O., korney.alena@yandex.ru, Graduate Student, Polzunov Altai State Technical University, Barnaul, 656038, Russian Federation
E-mail: korney.alena@yandex.ru

Received on July 21, 2019
Accepted on August 09, 2019

Social networks, microblogging sites, online shops, forums and other resources are now becoming more and more popular. People use Internet as a platform for their self-expression. Everyone can share their experiences and give a product feedback. People read public reviews to choose hotels, restaurants, smartphones, weekend movies, service providers etc. Aspect-based sentiment analysis (ABSA) systems allow extracting user opinions about different product/service aspects and features. ABSA includes four main tasks: Aspect term extraction (ATE), Aspect term polarity (ATP), Aspect category detection (ACD) and Aspect category polarity (ACP). All of these tasks are domain-sensitive. It means that ABSA-system must be built for a particular domain or adapted to it to achieve high quality of classification. Lexicon-based systems are considered as more adaptable then machine-learning or hybrid systems. There are many adaptation techniques: with or without using of core lexicon; based on graphs, semantic networks, domain-sensitive thesauri etc. Some approaches use the concept of different level of domain-sensitivity to reduce the amount of adaptation procedures, but this idea requires a source of common-sense knowledge integrated to the system.

This paper is devoted to the problem of domain adaptation in ABSA-tasks in Russian. It represents an approach based on generalized lexicons. Presented approach includes a statistical-based algorithm of sentiment lexicon extraction from short informal text corpuses. Mathematical model of sentiment mark is introduced. Semantic graph is used to extend basic sentiment lexicon. The graph contains the data extracted from common dictionaries, which is domain-independent, reliable and full enough to use presented graph as a source of common-sense knowledge. Graph structure allows adapting the data to a particular domain with keeping important common-sense knowledge in background. Semantic adaptation can also elevate the quality of explicit sentiment calculation based on graph.

Keywords: *sentiment analysis, aspect-based sentiment analysis, sentiment lexicon; semantic graph; sentiment classification model; sentiment calculation method; text classification; lexicon adaptation to domain; knowledge extraction;*

For citation:

Korney A. O., Kryuchkova E. N. Application of Adaptable Generalized Lexicons in Aspect-Based Sentiment Analysis, *Programmnaya Ingeneria*, 2019, vol. 10, no. 11—12, pp. 471—479.

DOI: 10.17587/prin.10.471-479

References

1. **Vasenin V. A., Roganov V. A., Dzabraev M. D.** Methods of Automated Sentiment Analysis of Texts Published by Mass Media, *Programmnaya Ingeneria*, 2016, vol. 7, no 8, pp. 360—372 (in Russian).
2. **Turney P.** Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews, *In Proceedings of the Association for Computational Linguistics*, 2002, pp. 417—424.
3. **Pang B., Lee L., Vaithyanathan S.** Thumbs up? Sentiment Classification using Machine Learning Techniques, *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002, pp. 79—86.
4. **Thet T. T., Na J. C., Khoo C. S. G.** Aspect-based sentiment analysis of movie reviews on discussion boards, *Journal of Information Science archive*, 2010, vol. 36, no. 6, pp. 823—848.
5. **Xue W. W., Li T. E.** Aspect Based Sentiment Analysis with Gated Convolutional Networks, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, Melbourne, Australia, 2018, pp. 2514—2523.
6. **Mashkin D.** Extracting Aspects, Category and Sentiment of Aspects in Russian User Reviews in Restaurants Domain, *Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialogue 2016"*, Moscow, 2016, 8 p., available at: http://www.dialog-21.ru/media/3453/mashkindo.pdf.
7. **Ivanov V., Tutubalina E., Mingazov N., Alimova I.** Extracting Aspects, Sentiment and Categories of Aspects in User Reviews about Restaurants and Cars, *Proceedings of the 21st International Conference on Computational Linguistics (Dialog-2015)*, 2015, vol. 2, pp. 46—57.
8. **Blinov P. D., Kotelnikov E. V.** Semantic Similarity for Aspect-Based Sentiment Analysis, *Proceedings of the 21st International Conference on Computational Linguistics Dialog-2015*, 2015, vol. 2, pp. 36—45.
9. **Popescu A. M., Nguyen B., Etzioni O.** OPINE: Extracting product features and opinions from reviews, *Proceedings of HLT/EMNLP on interactive demonstrations*, 2005, pp. 32—33.
10. **Mukherjee A., Liu B.** Aspect extraction through semi-supervised modeling, *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, 2012, vol. 1, pp. 339—348.
11. **Titov I., McDonald R.** Modeling online reviews with multi-grain topic models, *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 111—120.
12. **Andrianov I. A., Mayorov V. D., Turdakov D. Yu.** Modern methods of aspect-based sentiment analysis, *Trudy ISP RAN,* 2015, vol. 27, issue 5, pp. 5—22. DOI: 10.15514/ISPRAS-2015-27(5)-1 (in Russian).
13. **Mashkin D. O., Kotelnikov E. V.** Aspect Terms Extraction Based on Conditional Random Fields and Vector Representations of Words, *Proceedings of ISP RAS*, Volume 28, Issue 6, 2016, pp. 223—240 (in Russian).
14. **Hamdan H., Bellot P., Bechet F.** Super-vised Methods for Aspect-Based Sentiment Analysis, *Proceedings of the Eighth International Workshop on Semantic Evaluation* (SemEval 2014), pp. 596—600.
15. **Zubareva O., Savoy J.** Opinion Detection by Combining Machine Learning and Linguistic tools, *Proceedings of NTCIR-8 Workshop Meeting*, Tokyo, Japan, June 15—18, 2010, pp. 221—227.
16. **Schouten K., van der Weijde O., Frasincar F., Dekker R.** Supervised and Unsupervised Aspect Category Detection for Sentiment Analysis with Co-occurrence Data, *IEEE Transactions on Cybernetics*, 2018, vol. 48, issue 4, pp. 1263—1275. DOI: 10.1109/tcyb.2017.2688801.
17. **San Vicente I. N., Saralegi X., Agerri R.** Elixa: A modular and flexible ABSA platform, *Proceedings of the 9th International Workshop on Semantic Evaluation* Association for Computational Linguistics, Denver, Colorado (June 2015), 2015, pp. 748—752.
18. **Toh Z., Wang W.** DLIREC: Aspect Term Extraction and Term Polarity Classification System, *Proceedings of the 8th International Workshop on Semantic Evaluation*, 2014, pp. 235—240.
19. **Jebbara S., Cimiano P.** Aspect-Based Sentiment Analysis Using a Two-Step Neural Network Architecture, *Proceedings of the twenty-second European conference on artificial intelligence*, 2016, vol. 641, pp. 153—167.
20. **SemEval 2019** Task 9 — SubTask A — Suggestion Mining from Online Reviews and Forums, available at: https://competitions.codalab.org/competitions/19955.
21. **Negi S., Asooja K., Mehrotra S., Buitelaar P.** A Study of Suggestions in Opinionated Texts and their Automatic Detection, *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, Berlin, Germany, 2016, pp. 170—178.
22. **Bollegala D., Weir D., Carroll J.** Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification, *In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011, Portland, Oregon, 2011, pp. 132—141.
23. **Franco-Salvador M., Cruz F. L., Troyano J. A., Rosso P.** Cross-domain polarity classification using a knowledge-enhanced meta-classifier, *Knowledge Based Systems*, 2015, vol. 86, pp. 46—56.
24. **Dubatovka A., Kurochkin Yu., Mikhailova E.** Automatic Generation of the Domain-Specific Sentiment Russian Dictionaries, *Computational Linguistics and Intellectual Technologies: Proceedings of the Annual International Conference "Dialogue" (2016)*, 2016, issue 15, pp. 146—158.
25. **Labille K., Gauch S., Alfarhood S.** Creating Domain-Specific Sentiment Lexicons via Text Mining, *Proceedings of WISDOM 2017: 6th KDD Workshop on Issues of Sentiment Discovery and Opinion Mining*, At Halifax, Nova Scotia, Canada. available at: https://sentic.net/wisdom2017labille.pdf.
26. **Rubtsova Y. V.** Building of text corpus for sentiment classifier training, *Programmnye produkty i sistemy*, 2015, no. 1 (109), pp. 72—78 (in Russian).
27. **Krotova A., Krayvanova V., Kryuchova E.** Mathematical model of natural language for fuzzy associative search tasks, *SPECOM 2011 Proceedings*, Kazan, 2011, pp. 402—406 (in Russian).
28. **Cambria E., Livingstone A., Hissain A.** The hourglass of emotions, *Proceedings of the 2011 international conference on Cognitive Behavioral Systems*, February 21—26, 2011, Dresden, Germany, 2011, pp. 144—157.
29. **Kazakov M., Kruchkova E.** Classication of complex images based on semantic graph, *Prikladnaya informatika*, 2014, vol. 6 (54), pp. 79—89 (in Russian).
30. **Savchenko V.** Semantic Search Algorithms in Large Text Collections, *Supplementary Proceedings of AIST 2014*, 2014, pp. 161—166 (in Russian).
31. **Korney A., Kruchkova E.** Problems of aspect analysis under conditions of unbalanced training sample, *Vysokoproizvoditel'nye vychislitel'nye sistemy i tekhnologii*, 2019, vol. 3, no. 1, pp. 161—165 (in Russian).
32. **Ma Y., Peng H., Cambria E.** Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM, *AAAI*, 2018, pp. 5876—5883.
33. **Kutuzov A., Kuzmenko E.** WebVectors: A Toolkit for Building Web Interfaces for Vector Semantic Models, *Analysis of Images, Social Networks and Texts. AIST 2016. Communications in Computer and Information Science*, 2016, pp. 155—161.
34. **Alghunaim A., Mohtarami M., Cyphers S., Glass J.** A Vector Space Approach for Aspect Based Sentiment Analysis, *Proceedings of NAACL-HLT 2015*, Denver, Colorado, May 31 — June 5, 2015, pp. 116—122.

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4
Технический редактор *Е. М. Патрушева*. Корректор *З. В. Наумова*

Сдано в набор 07.10.2019 г. Подписано в печать 25.11.2019 г. Формат 60×88 1/8. Заказ PI11-1219
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**

Указатель статей, опубликованных в журнале "Программная инженерия" в 2019 г.