ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г.

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

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 (индексы: по каталогу агентства "Роспечать" — **22765**, по Объединенному каталогу "Пресса России" — **39795**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru/pi.html E-mail: prin@novtex.ru
Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

№ 4
April
2015

Published since September 2010

CONTENTS

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

УДК 004.89

Н. О. Дородных, аспирант, e-mail: tualatin32@mail.ru,
А. Ю. Юрин, канд. техн. наук, зав. лаб., e-mail: iskander@icc.ru,
Институт динамики систем и теории управления им. В. М. Матросова СО РАН (ИДСТУ СО РАН), г. Иркутск

Использование диаграмм классов UML для формирования продукционных баз знаний[1]

Дан анализ возможностей использования концептуальных моделей для автоматизированного формирования продукционных баз знаний. В качестве основных источников концептуальных моделей рассмотрены диаграммы классов UML как наиболее распространенный способ концептуализации знаний в процессе проектирования программного обеспечения в соответствии с объектным подходом. Описаны алгоритмы формирования элементов баз знаний CLIPS на основе анализа описания моделей, представленных в формате XML.

Ключевые слова: *база знаний, продукции, автоматизированное создание баз знаний, приобретение знаний, концептуальная модель, UML, CLIPS*

Введение

На настоящее время разработка новых методов и подходов к созданию прикладных программных систем искусственного интеллекта и, в частности, экспертных систем (ЭС) остается перспективной областью научных исследований. Ядром ЭС и систем, основанных на знаниях, является база знаний (БЗ), которая содержит как общие знания, так и информацию о частных случаях [1]. При этом сложность и трудоемкость процесса разработки ЭС обусловлены главным образом особенностью этапа разработки БЗ, который включает задачи идентификации (получения), концептуализации (структурирования) и формализации (представления) знаний, что считается традиционно "узким местом" проектирования ЭС [2].

Повышение эффективности получения (приобретения) знаний (*knowledge acquisition*) [2] из различных источников (баз данных, документов, концептуальных моделей и т. д.) является актуальной задачей. При этом особый интерес представляет повторное использование (в том числе и трансформация) моделей, построенных с использованием различных программных средств онтологического, когнитивного моделирования, CASE-средств (например, IBM Rational Rose и др.) [3]. Однако существующие в данной области решения имеют ряд недостатков, в частности, отсутствие возможности совместной распределенной и одновременной работы пользователей; отсутствие или ограниченность кодогенерации на языках представления знаний (ЯПЗ).

Целью работы, результаты которой представлены далее, являлось повышение эффективности процесса разработки продукционных БЗ путем автоматизированного анализа концептуальных моделей предметных областей, выполненных при помощи различных программных средств. Для достижения поставленной цели была осуществлена разработка алгоритмического и программного обеспечения [4], предназначенного для преобразования концептуальных моделей в онтологию предметной области (расширенную возможностью определения причинно-следственных отношений), а также для моделирования (модификации) продукций с использованием специальной нотации RVML (*Rule Visual Modeling Language*) [5] и их отображения (трансляции) в ЯПЗ CLIPS (*C Language Integrated Production System*) [6]. Использование онтологии с причинно-следственными отношениями позволяет унифицировать представление продукций, что обеспечивает возможность расширения набора доступных (поддерживаемых) в программном средстве ЯПЗ. Применение web-технологий в свою очередь обеспечивает возможность совместной и распределенной работы пользователей (экспертов, инженеров по знаниям и системных аналитиков) над проектами БЗ.

В качестве источников концептуальных моделей предлагается использовать диаграммы классов, построенные с использованием унифицированного языка моделирования UML (*Unified Modeling Language*) [7] и сохраненные в формате XML в соответствии со стандартом XMI (*XML Metadata Interchange*) [8].

[1] Работа выполнена при частичной поддержке гранта РФФИ 15-07-05641.

1. Постановка задачи

Анализ проблематики получения, структурирования и формализации знаний показал [1, 2], что в настоящее время наибольший интерес для исследователей представляют автоматизированные методы получения знаний. Это обстоятельство обусловлено главным образом наличием больших объемов накопленной информации, в том числе и в форме концептуальных моделей. При этом можно выделить основные группы программных средств, которые обеспечивают построение концептуальных моделей (рис. 1).

Результаты анализа данных систем моделирования позволяют выделить следующие их недостатки:

- практически отсутствуют системы (web-сервисы), обеспечивающие совместную, распределенную работу пользователей в сети Интернет;
- обеспечивая создание концептуальных моделей, а также генерацию различных отчетных документов в разных форматах, исследуемые системы либо не предусматривают возможность преобразования построенных моделей в структуры ЯПЗ, либо эта возможность ограничена (неполное преобразование или единственный ЯПЗ), что в свою очередь затрудняет возможности практического использования построенных моделей при разработке экспертных систем.

Существуют примеры успешного преодоления данных недостатков по отдельности, в частности, для обеспечения совместной распределенной работы над проектами возможно использование программных средств контроля конфигурации и целостности проекта, однако инструментальных средств, сочетающих данные возможности с функциями генерации программного кода баз, авторами статьи не обнаружено.

В целях преодоления отмеченных недостатков предлагается разработать алгоритмическое и программное обеспечение, которое не только позволяет генерировать БЗ на определенном языке программирования с использованием концептуальных моделей и технологии визуального моделирования, но и поддерживает возможность совместной и распределенной работы пользователей. Для апробации результатов такой разработки в качестве языка программирования выбран CLIPS.

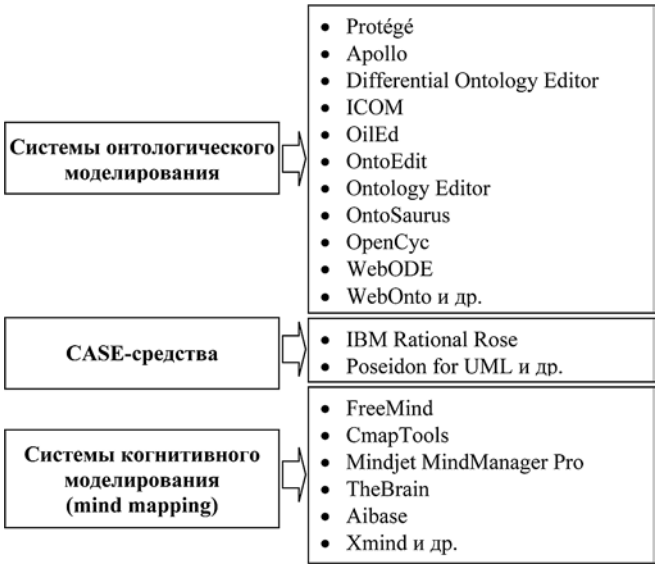

Рис. 1. Основные группы программных систем, обеспечивающих построение концептуальных моделей

Формализовать постановку задачи можно следующим образом. Необходимо определить оператор преобразования концептуальной модели T:

$$T: M \rightarrow Code^{CLIPS}, \quad (1)$$

где M — концептуальная модель; $Code^{CLIPS}$ — программный код на языке CLIPS.

2. Источники концептуальных моделей

Целью анализа программных систем, обеспечивающих построение концептуальных моделей, являлось в том числе выявление стандартов и языков, которые используются как при создании программного обеспечения (ПО), так и в процессе целенаправленного моделирования предметной области и построения ее онтологии (рис. 2).

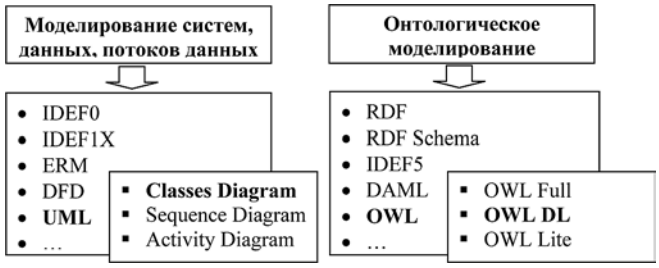

Рис. 2. Стандарты (нотации и языки) представления концептуальных моделей

На основе анализа был сделан вывод, что при моделировании программных систем (в процессе проектирования ПО) наиболее широко распространен язык UML [7]. Из языков описания онтологий особый интерес представляют языки, основанные на web-стандартах, в частности, OWL [9]. Поэтому в качестве основных источников концептуальных моделей могут быть рассмотрены диаграммы классов UML и онтологии OWL DL.

Далее рассмотрим подробнее использование диаграмм классов UML при автоматизированном формировании баз знаний и расширим выражение (1), доопределив концептуальную модель:

$$M = \left\langle M^{UML} \right\rangle. \quad (2)$$

3. Алгоритмическое обеспечение

Основной задачей алгоритмического обеспечения является преобразование концептуальных моделей в конструкции (элементы) ЯПЗ CLIPS, что может

Рис. 3. Процесс формирования кода БЗ на основе концептуальных моделей

быть представлено в виде последовательности действий (рис. 3). На этапах 1 и 2 такого преобразования средствами внешних программ пользователь строит концептуальную модель, которая представляется в формате XML. Этот формат является универсальным и наиболее распространенным способом интеграции программных систем и обеспечения обмена информацией между приложениями.

При представлении UML-моделей в виде XML-структур используется стандарт XMI (*XML Metadata Interchange*) [8], предназначенный главным образом для хранения UML-структур, а также любых других данных, метамодель которых задана с помощью MOF (*Meta Object Facility*), и обмена ими между различными инструментальными средствами разработки.

На этапе 3 процесса анализа UML-модели выделяются понятия предметной области и их отношения, далее (этап 4) на их основе формируется онтология, как универсальное представление продукций, независящее от используемого ЯПЗ. При помощи специальной графической нотации RVML [5] предоставляется возможность модификации, визуализации и проверки полученных продукций (этап 5). На этапе 6 происходит генерация кода базы знаний в формате CLIPS на основе онтологии.

Таким образом, уточним оператор преобразования концептуальной модели (1):

$$T = \langle T_{CM-XML}, T_{XML-ONT}, T_{ONT-Code} \rangle,$$

$$T_{CM-XML} : M_{CM} \to M_{XML},\ T_{XML-ONT} : M_{XML} \to \to M_{ONT},\ T_{ONT-Code} : M_{ONT} \to Code_{CLIPS}, \quad (3)$$

где T_{CM-XML} — оператор преобразования концептуальной модели в XML-структуру; $T_{XML-ONT}$ — оператор преобразования XML-структуры в онтологию; $T_{ONT-Code}$ — оператор преобразования онтологии в код на ЯПЗ CLIPS; M_{XML} — представление концептуальной модели в XML-формате; M_{ONT} — представление концептуальной модели в онтологии.

Вопросы, связанные с исследованием свойств предложенного алгоритмического обеспечения, в данной статье не рассмотрены. Отметим только конечный характер алгоритмов, что определяется конечным числом строк анализируемых файлов и обрабатываемых (извлеченных и преобразованных) элементов.

3.1. Описание концептуальной модели

Используя формулы (2) и (3), доопределим M_{XML} из выражения (3):

$$M_{XML} = M_{XML}^{UML}, \quad (4)$$

где M_{XML}^{UML} — модель UML (диаграмма классов) в формате XML.

Используя формулу (4), формализуем описание модели M_{XML}^{UML}:

$$M_{XML}^{UML} = \langle C^{UML}, DT^{UML}, R_A^{UML} \rangle,$$

где C^{UML} — описание классов; DT^{UML} — типы данных; R_A^{UML} — отношения между классами типа "ассоциация".

Уточним описание классов и отношений следующим образом:

- $C^{UML} = \{c_1^{UML} \dots c_m^{UML}\}, c_i^{UML} = \langle name_i, A_i^{UML} \rangle, i = \overline{1,m},$

где $name_i$ — имя i-го класса; A_i^{UML} — атрибуты i-го класса; m — число классов; в свою очередь, $A^{UML} = \{a_1^{UML} \dots a_k^{UML}\}$, где $a_j^{UML} = \langle name_j, type_j, value_j \rangle$, $j \in \overline{1,k}$, где $name_j$ — имя j-го атрибута; $type_j$ — тип j-го атрибута; $value_j$ — возможное значение, при этом $type_j \in DT^{UML}$; k — число атрибутов;

- $R_A^{UML} = \{r_{A_1}^{UML} \dots r_{A_n}^{UML}\}, r_l^{UML} = \langle name_l, r_{A-LHS_l}^{UML}, r_{A-RHS_l}^{UML} \rangle,$

$l \in \overline{1,n}$, где $name_l$ — имя отношения; $r_{A-LHS_l}^{UML} = \langle name_r, m_r, c^{UML} \rangle$ — левая часть связи; $name_r$ — наименование роли класса на определенной части связи, m_r — кратность (множественность) ассоциации; c^{UML} — ссылка на класс; $r_{A-RHS_l}^{UML} = \langle name_r, m_r, c^{UML} \rangle$ — правая часть связи, при этом $m_r \in M_r^{UML} = \{0; 0..1; 0..*; 1; 1..*\}$, n — число отношений.

3.2. Модель онтологии

Для унифицированного хранения и представления знаний разработана специальная модель онтологии. Разработанная модель позволяет абстрагироваться от особенностей описания продукций в различных языках программирования, используемых при реализации БЗ (например, CLIPS, JESS), и позволяет хранить знания в собственном независимом формате.

Формализуем описание модели онтологии, используя модель, предложенную в работе [10]:

$$M_{ONT} = \langle DT_{ONT}, CN_{ONT}, PN_{ONT}, Obj_{ONT}, R_{ONT}, E_{KRL} \rangle,$$

где DT_{ONT} — перечень базовых типов данных, при этом DT_{ONT} = {литерал, объект, коллекция}; CN_{ONT} — имена классов; PN_{ONT} — имена свойств классов; Obj_{ONT} — понятия (константы, объекты) предметной области; R_{ONT} — конечное множество отношений между концептами заданной предметной области; E_{KRL} — элементы (конструкции) языка программирования, соответствующие элементам модели онтологии, $E_{KRL} = \langle t_{KRL}, f_{KRL}, r_{KRL}, s_{KRL} \rangle$, где t_{KRL} — шаблон; f_{KRL} — факт; r_{KRL} — правило; s_{KRL} — слот (свойство), в данной работе $KRL = CLIPS$.

Уточним введенные понятия онтологии и расширим определение из работы [10] путем введения причинно-следственного отношения:

$R_{ONT} = \{R^{IS-A}_{ONT}, R^{P}_{ONT}, R^{C}_{ONT}\}$, где R^{IS-A}_{ONT} — отношение наследования между классами CN_{ONT}; R^{P}_{ONT} — отношение между классами и свойствами, pnR^{P}_{ONT} $\langle cn, cn\_bt \rangle$, где $pn \in PN_{ONT}$; $cn \in CN_{ONT}$; $cn\_bt \in CN_{ONT} \cup DT_{ONT}$, последнее означает, что свойство с именем pn_i характеризует класс cn_j, а значениями свойства могут быть элементы типа другого класса из CN_{ONT} или основного множества типов DT_{ONT}; R^{C}_{ONT} — причинно-следственные отношения между концептами. В свою очередь, $R^{C}_{ONT} = \{r^{C}_{ONT_1} ... r^{C}_{ONT_g}\}$, $r^{C}_{ONT_j} = \langle name_j, cn_{LHS_j}, cn_{RHS_j}, op_j \rangle$, $j \in \overline{1, g}$, где $name_j$ — имя отношения; cn_{LHS_j} — левый класс отношения (ссылка на класс); cn_{RHS_j} — правый класс отношения (ссылка на класс); op_j — оператор отношения, $op_j \in \{and, or, not\}$; g — число отношений.

3.3. Преобразование диаграмм классов UML

Далее кратко рассмотрим преобразование диаграммы классов UML в формате XML в онтологию продукций приложения:

$$T_{XML-ONT} : M^{UML}_{XML} \rightarrow M_{ONT}.$$

Следует отметить, что в данной работе в модели UML рассматриваются отношения только типа "ассоциация". По этой причине далее под отношением (связью) будем понимать именно ассоциацию.

Процесс преобразования представляет собой анализ структур XML-файла в целях выделения конструкций, описывающих элементы модели, а также поиска соответствия в таблице преобразований (табл. 1).

При этом в результате анализа формируются множества классов CN_{ONT}, свойств классов PN_{ONT}, объектов Obj_{ONT}, типов данных DT_{ONT} и отношений R_{ONT}

Таблица 1

Анализируемые элементы XMI UML

Элементы XMI UML	Описание
UML:Model	Модель диаграммы UML
UML:Class	Класс
UML:Attribute	Атрибут класса
UML:Expression	Значение атрибута
UML:DataType	Тип данных
UML:Association	Связь типа "ассоциация"
UML:AssociationEnd	Часть ассоциации (роль связи)
UML:Multiplicity	Мощность (кратность) связи
UML:MultiplicityRange	Интервал (диапазон) кратности связи

Таблица 2

Пример соответствия элементов онтологии и CLIPS

Элемент онтологии	Элемент CLIPS
Класс	(**deftemplate** <имя класса> "<описание класса>" <свойства>)
Свойство	(**slot**)
Значение свойства по умолчанию	(**default** "<значение>")
Причинно-следственное отношение	(**defrule** <имя отношения> "<описание>" <объекты> => <объекты>)
...	

онтологии продукций приложения. В частности, происходит преобразование ассоциаций в причинно-следственные отношения: $R_A^{UML} \to R_{ONT}^C$, при этом:

$$r_{A-LHS}^{UML} \to cn_{LHS}^{ONT},$$
$$r_{A-RHS}^{UML} \to cn_{RHS}^{ONT},$$
$$op = \begin{cases} and, \ if \ m_r \in \{1,1..n\} \\ or, \ if \ m_r \in \{0,0..n\} \end{cases}.$$

Таким образом, на формирование логического оператора влияет мощность (кратность) отношения правого элемента, если мощность не равна нулю, то назначается оператор *"and"*.

Необходимо отметить, что предложенная интерпретация отношения "ассоциация" не является общепринятой, поскольку отсутствует практика использования UML-моделей для формирования элементов БЗ.

На основе сформированной онтологии может быть получен программный код, например, на языке CLIPS:

$$T_{ONT-Code}: M_{ONT} \to Code_{CLIPS}.$$

Примеры соответствия элементов онтологии и CLIPS приведены в табл. 2.

4. Пример формирования продукций на основе концептуальных моделей

Рассмотрим пример автоматизированного формирования продукций на примере фрагмента диаграммы классов (рис. 4) [11].

В качестве источника UML-моделей, описывающих классы предметной области и их отношения, используется CASE-средство IBM Rational Rose. Анализируемый XML-файл с моделями получен с помощью модуля трансформации моделей XMI UML [8].

Графическому представлению модели (рис. 4) соответствует фрагмент XML-кода (рис. 5). При этом выделены ключевые конструкции (см. табл. 1), на основе которых осуществлялось извлечение необходимых элементов концептуальной модели.

Понятия онтологии могут быть представлены в нотации RVML [5] (рис. 6) для их дальнейшего уточнения и проверки.

На основе понятий онтологии генерируется код базы знаний в формате CLIPS (рис. 7).

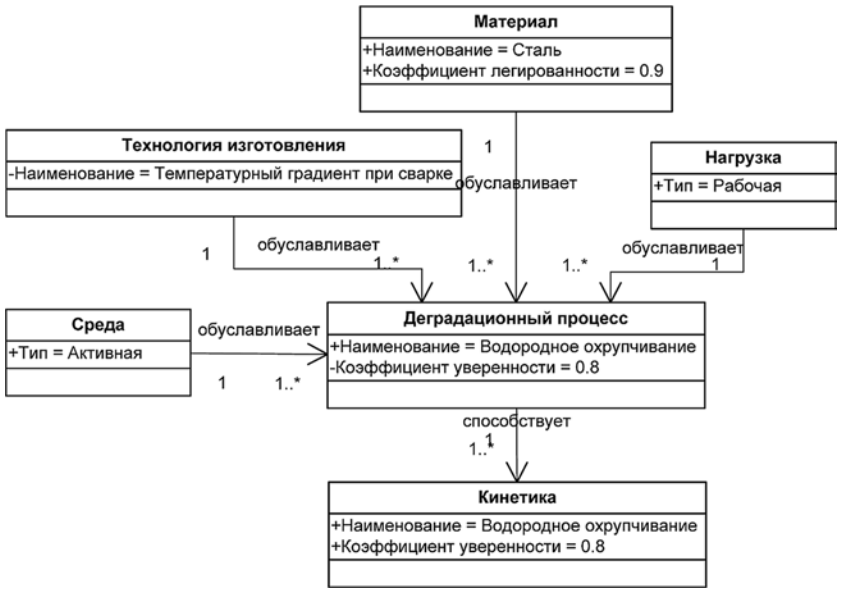

Рис. 4. Фрагмент анализируемой диаграммы классов UML

```
...
    <UML:Association xmi.id = 'G.4' name = 'обуславливает' ... >
        <UML:AssociationEnd xmi.id = 'G.5' type = 'S.350.1300.40.10' ... >
            <UML:Multiplicity>
                <UML:MultiplicityRange xmi.id = 'id.3510400.3' lower = '1' upper = '-1' />
            </UML:Multiplicity>
        </UML:AssociationEnd>
        <UML:AssociationEnd xmi.id = 'G.6' type = 'S.350.1300.40.1' ... >
            <UML:Multiplicity>
                <UML:MultiplicityRange xmi.id = 'id.3510400.4' lower = '1' upper = '1' />
            </UML:Multiplicity>
        </UML:AssociationEnd>
    </UML:Association>
...
    <UML:DataType xmi.id = 'G.16' name = 'String' ... />
    <UML:DataType xmi.id = 'G.17' name = 'Double' ... />
...
    <UML:Class xmi.id = 'S.350.1300.40.1' name = 'Материал' ... >
        <UML:Attribute xmi.id = 'S.350.1300.40.2' name = 'Наименование' type = 'G.16' ... >
            <UML:Expression language = '' >
              <UML:Expression.body>Сталь</UML:Expression.body>
            </UML:Expression>
        </UML:Attribute>
        <UML:Attribute xmi.id = 'S.350.1300.40.3' type = 'G.17' ... >
          <UML:ModelElement.name>Коэффициент легированности</UML:ModelElement.name>
            <UML:Expression language = '' body = '0.9' />
        </UML:Attribute>
    </UML:Class>
...
```

Рис. 5. Фрагмент анализируемой диаграммы классов UML в формате XML

Рис. 6. Пример полученной продукции в нотации RVML

```
;************************ Шаблоны ******************
(deftemplate material
   (slot naimenovanie (default "СТАЛЬ"))
   (slot koefficient_legirovannosti (default "0.9"))
   (slot cf (default "1"))
)
…
;************************ Правила ******************
(defrule pravilo_p_01 ""
   (declare (salience 1))
   (material
      (naimenovanie "СТАЛЬ")
      (koefficient_legirovannosti "0.9")
      (cf "1")
    )
…
=>
  (assert
       (degradacionnyy_process
            (naimenovanie "ВОДОРОДНОЕ ОХРУПЧИВАНИЕ")
            (cf "0.8")
        )
  )
)
…
```

Рис. 7. Фрагмент кода CLIPS, соответствующий RVML-модели

Заключение

Решение задач, связанных с получением (приобретением), структурированием и формализацией знаний, позволяет повысить эффективность процесса разработки БЗ интеллектуальных систем. В настоящей работе предложено автоматизировать анализ концептуальных моделей и автоматически формировать программный код на основе графических примитивов. Такой подход позволяет избежать ошибок программирования, которые, как правило, связаны с большими издержками на последующих этапах жизненного цикла программного продукта.

При решении задач исследования проведен анализ методов, подходов и систем, а также различных стандартов (языков) концептуального и онтологического моделирования. На основании этого анализа были выявлены основные источники концептуальных моделей: диаграммы классов UML и онтологии OWL DL. В качестве целевого ЯПЗ использован CLIPS. Подробно описан алгоритм преобразования концептуальных моделей (диаграмм классов UML) в программный код продукционной базы знаний.

На основе разработанного алгоритмического обеспечения реализован исследовательский прототип web-сервиса для автоматизированного формирования продукционных баз знаний CLIPS [4]. Основной областью применения такого сервиса является обеспечение распределенной коллективной работы специалистов-предметников при создании БЗ для продукционных экспертных систем. Сервис использовался при выполнении работ по хоздоговору с ОАО "ИркутскНИИхиммаш" на создание проблемно-ориентированного редактора продукционных БЗ в области оценки технического состояния и остаточного ресурса нефтехимического оборудования [12].

Безусловно, данный подход не позволяет исключить ошибки, обусловленные неточностью или неполнотой анализируемых концептуальных моделей. Однако автоматическая генерация программного кода на основе моделей позволяет использовать принцип быстрого прототипирования при реализации БЗ с последующей их проверкой и тестированием в сторонних средствах. По результатам тестирования можно внести необходимые изменения в модели (диаграммы классов UML или RVML) и провести повторную генерацию программного кода.

В дальнейшем планируется расширить функции программного обеспечения за счет подсистемы тестирования сгенерированных БЗ, использовать в качестве источников концептуальных моделей OWL DL, а также осуществить интеграцию сервиса со средством создания продукционных экспертных систем на основе порождающего программирования (модельно-управляемого подхода) [5].

Список литературы

1. **Люгер Дж. Ф.** Искусственный интеллект: стратегии и методы решения сложных проблем, 4-е изд.: пер. с англ. М.: Вильямс, 2003. 864 с.

2. **Гаврилова Т. А., Хорошевский В. Ф.** Базы знаний интеллектуальных систем. СПб.: Питер, 2000. 384 с.

3. **Система** визуального моделирования для разработки приложений "IBM Rational Rose". 2014. URL: http://www-03.ibm.com/software/products/ru/enterprise (дата обращения: 10.12.2014).

4. **Дородных Н. О., Юрин А. Ю.** Web-сервис для автоматизированного формирования продукционных баз знаний на основе концептуальных моделей // Программные продукты и системы. 2014. № 4. С. 103—107.

5. **Грищенко М. А., Юрин А. Ю., Павлов А. И.** Разработка экспертных систем на основе трансформации информационных моделей предметной области // Программные продукты и системы. 2013. № 3. С. 143—147.

6. **Частиков А. П., Гаврилова Т. А., Белов Д. Л.** Разработка экспертных систем. Среда CLIPS. СПб.: БХВ-Петербург, 2003. 608 с.

7. **Документация** спецификации Unified Modeling Language (UML). 2014. URL: http://www.omg.org/spec/UML (дата обращения: 10.12.2014).

8. **Документация** спецификации XML Metadata Interchange (XMI). 2014. URL: http://www.omg.org/spec/XMI (дата обращения: 10.12.2014).

9. **Документация** стандарта Web Ontology Language (OWL). 2014. URL: http://www.w3.org/TR/owl-semantics (дата обращения: 10.12.2014).

10. **Николайчук О. А.** Методы, модели и инструментальное средство для исследования надежности и безопасности сложных технических систем: Автореферат дис. ... д-ра техн. наук. М., 2011. 37 с.

11. **Берман А. Ф.** Информатика катастроф // Проблемы безопасности и чрезвычайных ситуаций. 2012. № 3. С. 17—37.

12. **Юрин А. Ю., Грищенко М. А.** Редактор баз знаний в формате CLIPS // Программные продукты и системы. 2012. № 4. С. 83—87.

N. O. Dorodnykh, Postgraduate Student, e-mail: tualatin32@mail.ru,
A. Yu. Yurin, Head of Laboratoty, e-mail: iskander@icc.ru, Institute for System Dynamics and Control Theory Siberian Branch of the Russian Academy of Sciences, Irkutsk

Using UML Class Diagrams for Design of Knowledge Bases of Rule-Base Expert Systems

The paper describes the approach for computer-aided design of rule-bases on the basis of conceptual models. The proposed approach consists of six steps: design of the conceptual model, representation of the conceptual model in the form of a XML-document, analysis (parsing) of the XML-document, generation ontology of rules, visualization and modification of rules, model-based generation of a program code. UML class diagrams are considered as the main source of conceptual models, because the analysis shown that it is the most common way of conceptualizing knowledge in software development. The original notation — Rule Visual Modeling Language (RVML) is proposed for visualization and modification of rules. The C Language Production System (CLIPS) is selected as a targeted programming language.

The main advantage of the approach is minimization of errors in program codes.

The approach does not eliminate errors caused by inaccuracy or incompleteness of analyzed conceptual models, however, the automatic model-based generation of program codes allows to apply the principle of rapid prototyping for the design of knowledge bases with the subsequent inspection and testing obtained codes with the aid of third-party tools.

The proposed algorithms are implemented in a prototype of a web-service. The main area of application of the web-service is providing distributed collaboration of experts and analytics in design of knowledge bases for rule-based expert systems.

Keywords: *knowledge base, rules, computer-aided, knowledge acquisition, conceptual model, ontology, UML, CLIPS*

References

1. **Luger G. F.** *Iskusstvennyj intellekt: strategii i metody reshenija slozhnyh problem*, 4-e izd. (Artificial Intelligence: Structures and Strategies for Complex Problem Solving, 4th ed). Translated from English. Moscow: Williams, 2003, 864 p. (in Russian).

2. **Gavrilova T.A., Khoroshevskii V.F.** *Bazy znanij intellektual'nyx sistem* (Knowledge bases of intelligent systems). Sankt-Petersburg: Piter, 2000, 384 p. (in Russian).

3. **Sistema** vizual'nogo modelirovanija dlja razrabotki prilozhenij "IBM Rational Rose" (Visual modeling software for application development "IBM Rational Rose"). 2014, available at: http://www-03.ibm.com/software/products/ru/enterprise (accessed: December 10, 2014). (in Russian).

4. **Dorodnykh N. O., Yurin A. Yu.** Web-servis dlya avtomatizirovannogo formirovaniya produktsionnykh baz znanii na osnove kontseptual'nykh modelei. *Programmnye produkty i sistemy*, 2014, no. 4, pp. 103—107 (in Russian).

5. **Grishchenko M. A., Yurin A. Yu., Pavlov A. I.** Razrabotka ekspertnykh sistem na osnove transformatsii informatsionnykh modelei predmetnoi oblasti. *Programmnye produkty i sistemy,* 2013, no. 3, pp. 143—147 (in Russian).

6. **Chastikov A. P., Gavrilova T. A., Belov D. L.** *Razrabotka ejspertnyx sistem. Sreda CLIPS* (Development of expert systems. CLIPS environment). Sankt-Petersburg: BHV-Petersburg, 2003, 608 p. (in Russian).

7. **Documentation** of specification of Unified Modeling Language (UML), available at: http://www.omg.org/spec/UML/ (accessed December 10, 2014).

8. **Documentation** of specification of XML Metadata Interchange (XMI), available at: http://www.omg.org/spec/XMI/ (accessed December 10, 2014).

9. **Documentation** of standard Web Ontology Language (OWL). 2014, available at: http://www.w3.org/TR/owl-semantics (accessed December 10, 2014).

10. **Nikolaychuk O. A.** *Metodi, modeli i instrumentalnoe sredstvo dlia issledovania nadezhnosti i bezopasnosti slozhnih tehnicheskih system:* avtoref. diss. d-ra tehn. nauk. (Methods, models and software for investigation reliability and safety of complex technical systems. Thesis abstract on scientific degree Dr. of Techn. Sciences). Moscow, 2011. 37 p. (in Russian).

11. **Berman A. F.** Informatika katastrof. *Problemyi bezopasnosti i chrezvyichaynyih situatsiy*, 2012, no. 3, pp. 17—37 (in Russian).

12. **Yurin A. Yu., Grishchenko M. A.** Redaktor baz znanii v formate CLIPS. *Programmnye produkty i sistemy,* 2012, no. 4, pp. 83—87 (in Russian).

УДК 004

В. А. Васенин, д-р физ.-мат. наук, проф., e-mail: vasenin@msu.ru,
А. А. Иткес, канд. физ.-мат. наук, науч. сотр., e-mail: itkes@imec.msu.ru,
К. А. Шапченко, канд. физ.-мат. наук, стар. науч. сотр., e-mail: shapchenko@iisi.msu.ru,
НИИ механики МГУ им. М. В. Ломоносова

О применении моделей разграничения доступа в социальных сетях к одному классу многопользовательских систем управления контентом

Изложены результаты исследования известных моделей логического разграничения доступа в социальных сетях на предмет возможности их использования в многопользовательских системах управления контентом с учетом задаваемых отношений вида "пользователь—пользователь", "пользователь—ресурс" и "ресурс—ресурс". Возможность практического использования исследуемых моделей оценивается в применении к информационно-аналитической системе "Наука—МГУ" ("ИСТИНА"), выполняющей функции по управлению информацией о результатах научной деятельности и использующей сложные правила разграничения доступа.

Ключевые слова: *разграничение доступа, правила разграничения доступа, модель логического разграничения доступа, система управления контентом, социальная сеть, наукометрическая информация*

Введение

В последние годы значительное распространение получили программные системы, позволяющие большому числу пользователей создавать и изменять содержимое информационных ресурсов, — многопользовательские системы управления контентом. В таких системах зачастую требуется ограничить ряд действий в зависимости от некоторых условий, например, наличия определенного атрибута у пользователя, выполняющего действие, или присутствия некоторой связи между объектами в системе. Совокупность таких ограничений составляет правила разграничения доступа в рассматриваемой системе. В качестве объектов доступа могут выступать данные, вычислительные ресурсы, сервисы и другие объекты системы.

В традиционных системах управления контентом для описания правил разграничения доступа к объектам системы используют, как правило, известные дискреционные, многоуровневые и ролевые модели. Такие модели хорошо зарекомендовали себя для применения в условиях, когда правила разграничения доступа являются достаточно простыми, а именно — есть определенная и редко изменяемая структура (например, иерархия), связывающая отдельные объекты или субъекты в системе, и нет большого числа динамически изменяемых во времени атрибутов.

Однако следует отметить, что в ряде многопользовательских систем управления контентом целесообразно использовать правила разграничения доступа, которые записываются в более общей по сравнению с традиционными моделями форме. Это обусловлено возможностью сокращения действий по настройке таких правил, что в некоторых случаях очень востребовано. Например, одно общее декларативное правило, связывающее сущности в системе некоторым сложным условием, может заменить большое число традиционных правил. В этом общем правиле могут быть учтены некоторые изменения, которые могут быть позже проведены в системе. К их числу относятся, например, добавление пользователей и их групп, изменение отношений между пользователями или объектами системы. В отличие от более общего подхода, подобные изменения при использовании традиционных моделей и механизмов, как правило, должны сопровождаться корректировкой правил разграничения доступа.

Примером такой многопользовательской системы управления контентом является информационно-аналитическая система (ИАС) "Наука—МГУ" ("ИСТИНА") [1]. Эта система в контексте целей настоящей статьи является предметом исследования и

применения моделей, методов и программных механизмов логического разграничения доступа к ее ресурсам (объектам доступа). Такая система предоставляет возможности по формированию, анализу и модификации информации о результатах научной деятельности. Данная информация вносится в систему и корректируется ее пользователями. Механизмы разграничения доступа в этой системе позволяют снизить количество ошибочно введенных данных и предоставляют возможность ограничить доступ к определенной аналитической информации в зависимости от действующих связей между пользователями и объектами доступа в системе.

Информационно-аналитические системы, ориентированные на управление наукометрической информацией и подобные ИАС "Наука—МГУ" ("ИСТИНА"), являются относительно новым классом многопользовательских систем управления контентом. Вместе с тем определенным сходством с рассматриваемыми многопользовательскими системами управления контентом обладают социальные сети. Это обусловлено большим числом их пользователей, динамикой изменения состава пользователей и отношений между ними, наличием вносимого и изменяемого пользователями контента, а также правилами разграничения доступа, которые определяются в общей форме и зависят от динамически изменяемых отношений. Для социальных сетей в настоящее время разработано некоторое количество формальных моделей, описывающих правила разграничения доступа и принципы работы программных механизмов, которые реализуют такие правила. Такие модели анализируются в настоящей статье на предмет возможности их адаптации к многопользовательским системам управления контентом, в которых представляется целесообразным расширить способ задания правил разграничения доступа для более точного соответствия семантике таких правил, а также для сокращения административных действий, выполняемых по настройке правил разграничения доступа при определенных изменениях в процессе штатного функционирования системы.

Авторы рассматривают данную статью как первую в серии публикаций, освещающих результаты исследований, направленных на поиск моделей, методов и эффективных программных реализаций логического разграничения доступа к ресурсам многопользовательских систем управления наукометрическим контентом. Целью настоящей публикации является анализ состояния исследований в области построения и программной реализации моделей логического разграничения доступа к ресурсам социальных сетей и выявление свойств таких моделей, которые могут быть использованы в системах, аналогичных ИАС "Наука—МГУ" ("ИСТИНА").

1. Цели и задачи работы

В настоящей работе представлен краткий сравнительный анализ существующих моделей и механизмов логического разграничения доступа (ЛРД) к объектам доступа (ресурсам) в социальных сетях. Целью такого анализа является выявление возможностей их использования в системах управления наукометрической информацией.

При анализе моделей ЛРД, используемых в социальных сетях, особое внимание уделено свойствам, наиболее существенным для применения этих моделей в системе разграничения доступа к объектам ИАС "Наука—МГУ" ("ИСТИНА").

Выбор рассматриваемых моделей ЛРД в социальных сетях обусловлен результатами анализа литературных источников, который показал, что исследованиями моделей ЛРД для социальных сетей в мире занимаются в основном три авторских коллектива. Каждый из этих коллективов имеет несколько ключевых публикаций, описывающих достаточно похожие модели. В разд. 3 рассмотрены модели, предложенные П. Фонгом и его соавторами [2—4]; в разд. 4 рассмотрены модели, разработанные Б. Карминати, Е. Феррари и др. [5—7]; разд. 5 содержит анализ работ Ю. Ченга, Дж. Парка и Р. Сандху [8—11].

2. Логическое разграничение доступа в рассматриваемых системах

Примем следующие соглашения об обозначениях и терминологии. Будем считать, что во всех рассматриваемых моделях задано множество O объектов в информационной системе, в котором выделено подмножество пользователей $U \subset O$. В дальнейшем все объекты системы, не являющиеся пользователями, будем именовать ресурсами $Res = O \backslash U$ в соответствии с терминологией, принятой в работах [8, 9]. Поскольку в исследуемых моделях в большей степени представляет интерес возможность доступа пользователя к ресурсу, то под объектом доступа будем подразумевать ресурс, если не указано иное.

Большинство рассматриваемых моделей разграничения доступа так или иначе оперирует понятием социального графа, однако в разных работах этот термин может употребляться в двух разных смыслах. В первом случае (например, в работах [2, 8]) социальным графом называется нагруженный граф, вершинами которого являются пользователи системы. Две вершины связаны каким-либо ребром в том и только том случае, если они связаны каким-либо отношением, с каждым ребром ассоциировано множество отношений, связывающих указанных пользователей. Во втором случае (например, в работе [9]) вершинами графа является множество всех объектов системы,

а ребрами, как и в первом случае, являются связывающие их отношения. В дальнейшем социальный граф, построенный на множестве всех объектов системы, будем именовать полным социальным графом, а построенный на множестве пользователей — усеченным социальным графом.

Рассмотрим подробнее сходство и отличия социальных сетей и многопользовательских систем управления наукометрическим контентом на примере ИАС "Наука—МГУ" ("ИСТИНА") в части логического разграничения доступа.

Целевая система "Наука—МГУ" ("ИСТИНА") во многом организована сходным образом с социальными сетями. По аналогии с социальными сетями в ней имеется большое число пользователей и добавление новых пользователей возможно в автоматическом режиме, без участия администратора системы. Каждый пользователь системы может добавлять объекты, связанные с другими объектами, и при этом могут изменяться права доступа к объектам других пользователей. Так, например, при добавлении одним из пользователей ИАС "Наука—МГУ" ("ИСТИНА") в систему новой публикации, имеющей других соавторов, эти соавторы также получают некоторые права доступа к информации об этой новой публикации и, возможно, к информации о журнале, в котором она опубликована. Меняются также права доступа к объектам для пользователей, ответственных за сопровождение информации по отдельным подразделениям, в которых работают авторы новой публикации.

Традиционные модели ЛРД ограниченно подходят для управления доступом к подобным системам, так как они требуют выделения в системе статического (изменяемого только администратором системы) множества атрибутов безопасности как для пользователей, так и для объектов. Однако в системах, аналогичных ИАС "Наука—МГУ" ("ИСТИНА"), при добавлении в систему любого пользователя он должен получить права доступа, не совпадающие ни с одним из существующих пользователей, а связанные с ним объекты получают новые атрибуты безопасности, соответствующие типу взаимосвязи с новым пользователем.

Для разграничения доступа к объектам подобных систем целесообразно использовать модели, предусматривающие принятие решения о предоставлении пользователю доступа к объекту в зависимости от структуры последовательностей пользователей или объектов доступа, связывающих пользователя с целевым объектом определенной последовательностью отношений. К числу подобных правил для традиционных социальных сетей можно отнести, например, правило "видеть фотографию может владелец, его друзья или друзья друзей", а для систем управления наукометрическим контентом — правило "изменять название статьи может один из авторов, либо ответственный по подразделению, в котором работает один из авторов".

Следует особо отметить, что ИАС "Наука—МГУ" ("ИСТИНА") по сравнению с социальными сетями имеет и свои особенности. В классических социальных сетях большинство объектов имеют пользователя-владельца, хотя могут быть связаны различными отношениями и с другими пользователями. Для многих объектов в ИАС "Наука—МГУ" ("ИСТИНА") не существует одного пользователя-владельца, который должен иметь больше прав доступа к этому объекту, чем остальные пользователи. Например, если публикация имеет несколько авторов, то с точки зрения прав доступа к этой публикации все авторы должны быть равноправны. В то же самое время многие модели разграничения доступа в социальных сетях, например, представленные в работах [2, 3, 5, 8], требуют наличия у каждого объекта системы единственного пользователя-владельца и способны регламентировать доступ других пользователей к объекту только на основании отношений между пользователем, запросившим доступ, и владельцем объекта. Другие модели [9, 10] допускают разные виды отношений между пользователями и объектами системы, отличные от отношения "пользователь является владельцем объекта". Заметим, что тем не менее эти модели также предполагают наличие у объекта пользователя-владельца, который имеет право изменять права доступа к объекту других пользователей.

В работах [8, 10] представлен краткий сравнительный анализ нескольких моделей, рассматриваемых в настоящей работе, в том числе представленных в публикациях [2—7]. Предложены следующие аспекты оценки выразительных свойств моделей ЛРД:

1) допускаются разные виды отношений;

2) отношения не обязаны быть симметричными;

3) рассматриваются отношения между пользователями;

4) рассматриваются отношения между пользователями и ресурсами;

5) пользователь может управлять доступом к "своим" объектам;

6) проводится разграничение доступа как к ресурсам, так и к пользователям;

7) возможность ассоциировать с пользователем запрет на доступ к некоторым ресурсам;

8) максимальная длина пути в социальном графе, на основании существования которого принимается решение о доступе;

9) выразительные возможности описания пути в социальном графе, на основании существования которого принимается решение о доступе;

10) учет более сложных, чем существование пути, топологических свойств социального графа, например, требование существования не менее определенного числа общих друзей у пользователя, запро-

сившего доступ, и пользователя-владельца объекта доступа;

11) учет дополнительных атрибутов пользователей;

12) учет дополнительных атрибутов отношений.

Среди перечисленных аспектов оценки выразительности моделей выполнение пунктов 1—4 и 7 является необходимым для разграничения доступа к объектам ИАС "Наука—МГУ" ("ИСТИНА") в силу особенностей внутреннего устройства системы и предъявляемых к ней функциональных требований. Так, например, необходимость выполнения пунктов 1—4 обусловлена тем, что в системе существует множество видов отношений между пользователями и ресурсами системы. Большинство отношений при этом не является симметричными, например, отношение авторства, которое связывает сотрудника с публикацией. Выполнение пунктов 5 и 6 при этом является желательным, но не обязательным в ИАС "Наука—МГУ" ("ИСТИНА").

Пункты 8 и 9 классифицируют модели ЛРД по максимальной длине путей в социальном графе, которые могут использоваться при принятии решения о предоставлении пользователю доступа к объекту, и по требованиям к структуре этого пути. В силу особенностей использования ИАС "Наука—МГУ" ("ИСТИНА") для разграничения доступа к ее объектам необходимо использовать правила, которым соответствуют достаточно длинные пути в социальном графе, состоящие из многих отношений. Например, пользователю *u* может предоставляться доступ на изменение к публикации *p*, если существуют такой сотрудник *a* и такие подразделения s_1 и s_3, что

- пользователь *u* является ответственным по подразделению s_1;
- подразделение s_3 совпадает с подразделением s_1, либо входит в подразделение s_1, либо существует такое подразделение s_2, входящее в s_1, что s_3 входит в s_2;
- сотрудник *a* входит в подразделение s_3;
- сотрудник *a* является автором публикации *p*.

Таким образом, модель ЛРД для ИАС "Наука—МГУ" ("ИСТИНА") должна предусматривать возможность предоставлять пользователю доступ к объекту в зависимости от того, связан ли этот пользователь с этим объектом путями достаточно большой длины и достаточно сложной структуры.

Следует также заметить, что некоторые из отношений между объектами ИАС "Наука—МГУ" ("ИСТИНА") по своей сути являются транзитивными, например, отмеченное выше отношение вложенности подразделений. В связи с этим фактом разумным требованием к модели ЛРД для этой системы является возможность использования правил доступа пользователя к объекту на основании существования в социальном графе пути, включающего цепочку вложенных подразделений произвольной длины. Отметим, что проверка пути на соответствие правилу, содержащему длинную последовательность повторений одного отношения, в ряде случаев может быть реализована с высоким уровнем производительности за счет использования специализированных возможностей современных реляционных СУБД по выполнению иерархических и рекурсивных запросов.

Выполнение пунктов 10—12 необходимо для задания сложных правил разграничения доступа. При этом для более полного соответствия задаваемых правил семантике разграничения доступа в ИАС "Наука—МГУ" ("ИСТИНА") целесообразно в перечень аспектов оценки моделей ЛРД добавить следующие пункты:

- учет отношений вида "ресурс—ресурс";
- учет дополнительных атрибутов ресурсов.

Учет отношений вида "ресурс—ресурс" необходим в силу ряда причин, одна из которых состоит в том, что в системе разделены объекты "пользователь" и "сотрудник". По этой причине почти все объекты, так или иначе связанные с пользователем (в первую очередь — публикации), связаны с ним не непосредственно, а посредством объекта "сотрудник". В других случаях пользователь должен получать доступ к объекту на основании более длинной цепочки ресурсов, связывающих пользователя с целевым ресурсом. Например, пользователь, назначенный ответственным по подразделению, должен иметь ряд прав доступа к публикациями, авторы которых работают в этом подразделении. По этой причине без отношений между ресурсами системы представление этих связей невозможно.

При разграничении доступа к объектам ИАС "Наука—МГУ" ("ИСТИНА") также может быть необходим учет дополнительных атрибутов вершин и ребер, входящих в путь, соединяющий пользователя с объектом доступа. Например, возможна ситуация, когда пользователю необходимо предоставить доступ к публикации в случае, если этот пользователь является ответственным по подразделению, в котором была выполнена эта публикация. Одним из возможных способов решения этой задачи было бы предоставление пользователю доступа к статье на основании наличия пути в социальном графе вида "пользователь является ответственным по подразделению" — "в подразделении работает сотрудник" — "сотрудник является автором статьи". В таком случае доступ разрешается, если существует такое подразделение и такой сотрудник, что пользователь связан с подразделением отношением администрирования (является ответственным), сотрудник связан с подразделением отношением "работает в" и связан со статьей отношением авторства. Данное правило однако может дать не тот результат, который был необходим, в случае, если во время написания статьи сотрудник работал в другом подразделении. В этом случае пользователь получает доступ к статье,

не являясь ответственным по подразделению, в котором была написана статья. Корректным вариантом решения данной задачи является ассоциирование с каждым ребром социального графа некоторого набора дат. С каждым ребром, связывающим сотрудника с подразделением отношением "работает в", связывается дата его поступления на работу и дата увольнения, если оно было. С каждым ребром, связывающим сотрудника с публикацией отношением авторства, ассоциируется дата сдачи публикации в печать. Пользователь может получить доступ к статье на основании наличия в социальном графе пути вида "пользователь является ответственным по подразделению" — "в подразделении работает сотрудник" — "сотрудник является автором статьи", соединяющим этого пользователя с этой статьей только при условии, если дата написания публикации, соответствующая отношению "сотрудник является автором статьи", лежит в диапазоне дат, соответствующих отношению "в подразделении работает сотрудник".

Таким образом, разграничение доступа к объектам ИАС "Наука—МГУ" ("ИСТИНА") требует использования в правилах предоставления доступа пользователя к объекту не только топологических свойств социального графа, но и дополнительных свойств вершин и ребер, входящих в путь от пользователя к объекту доступа.

С учетом перечисленных выше особенностей ИАС "Наука—МГУ" ("ИСТИНА") и требований, предъявляемых к ее системе разграничения доступа, рассмотрим более детально механизмы ЛРД в социальных сетях и оценим возможности их использования в целевой системе.

3. Модели Фонга

В настоящем разделе рассмотрены модели логического разграничения доступа, представленные в работах [2—4].

Первой из рассматриваемых моделей логического разграничения доступа к ресурсам социальных сетей является модель сохранения приватности Фонга—Анвара—Жао [2]. Эта модель рассматривает все ресурсы системы как элементы пользовательского профиля. В модели задано множество пользователей, множество типов объектов доступа, каждый ресурс системы однозначно идентифицируется его типом и пользователем-владельцем.

При использовании данной модели система представлена усеченным социальным графом, представляющим собой неориентированный ненагруженный граф, вершинами которого являются пользователи системы, а ребрами — отношения дружбы, связывающие пользователей. Другие отношения в модели не рассматриваются, а отношение дружбы предполагается симметричным. При этом единственное отношение между пользователями и ресурсами системы — "пользователь является владельцем ресурса", причем каждый ресурс в системе имеет только одного пользователя-владельца. Отношения, связывающие ресурсы системы, не рассматриваются.

Расширением модели Фонга—Анвара—Жао является модель Фонга, представленная в работе [3]. В отличие от предыдущей модели данная модель позволяет рассматривать несколько различных отношений между пользователями системы, причем такие отношения могут не быть симметричными. Модель Фонга оперирует также рядом других понятий, определяющих разные аспекты функционирования системы. Вместе с тем данная модель, как и модель Фонга—Анвара—Жао, не учитывает возможных отношений между ресурсами системы, а единственным отношением, которое может связывать пользователя и ресурс системы, является отношение владения.

В работе [4] предлагаются специальные языки, которые могут использоваться для описания моделей, аналогичных модели Фонга.

В первом из предложенных в работе [4] языков каждая операция с объектом может быть выполнена в случае, если истинна некоторая соответствующая ей формула. Это логическая формула, которая может быть истинной или ложной в зависимости от существования в социальном графе пути, обладающего определенными свойствами и соединяющего пользователя u, запросившего доступ, и пользователя v, являющегося владельцем объекта доступа. В языке, на котором записывается эта формула, существуют выделенные символы *true*, *false* и a. Эти символы означают соответственно: отношение, связывающее любую пару пользователей; отношение, не связывающее никакие пары пользователей; отношение, связывающее пользователя с собой. Истинность формулы определяется для пользователей u и v с учетом следующих правил.

- Формула a истинна, если $u = v$.
- Формула $\neg \varphi$ истинна в том и только том случае, если φ ложна.
- Формулы $\varphi \wedge \psi$ и $\varphi \vee \psi$ имеют соответственно смысл логической конъюнкции или дизъюнкции исходных формул φ и ψ.
- Для отношения r и формулы φ формула $r\ \varphi$ истинна в том и только том случае, если существует такой пользователь u', связанный с пользователем u отношением r, что формула φ истинна для пользователей u' и v.

Например, если отношение *parent* связывает родителя и ребенка, то формула *parent parent a* означает, что доступ к объекту должен быть предоставлен родителям родителей пользователя-владельца. Формула *parent parent a* $\vee$ *parent a* $\vee$ *a* означает, что доступ к объекту должен быть предоставлен владельцу, его родителям или родителям родителей.

Второй язык, рассмотренный в работе [4], является расширением первого. В дополнение к операторам первого языка в формулах допускается использование следующих конструкций.

- Для формул φ и ψ формула $\varphi \otimes \psi$ истинна в том и только том случае, если в социальном графе существуют пути, соответствующие формулам φ и ψ, и при этом эти пути не пересекаются за исключением того, что имеют общие начало и конец.
- Для формулы φ и символа (переменной) p формула $@p\varphi$ истинна в том и только том случае, если формула φ (содержащая в себе символ p) была бы верна, считая символ p указывающим на владельца объекта.

Данный язык позволяет предоставлять пользователям доступ к объектам, основываясь на более сложных топологических свойствах социального графа. Например, в работе [4] доказано, что во втором представленном языке возможно разрешить доступ к объекту для пользователей, имеющих не менее n общих друзей с владельцем объекта при любом n, что невозможно в первом языке.

Заметим однако, что использование языков сложного вида для задания пользователем правил предоставления доступа к "своим" объектам может оказаться источником ошибок при задании этих правил. Отмеченный факт обусловлен тем обстоятельством, что пользователь фактически должен заниматься программированием, в то время как он, возможно, не имеет соответствующих навыков.

Все модели разграничения доступа, представленные в работах [2—4], оперируют усеченным социальным графом, рассматривающим только отношения между пользователями системы. Одновременно с этим для разграничения доступа к объектам ИАС "Наука—МГУ" ("ИСТИНА") необходим также учет отношений между ресурсами системы, что не позволяет использовать рассмотренные модели в явном виде.

4. Модели Карминати—Феррари

В настоящем разделе рассмотрены модели ЛРД, представленные в работах [5—7].

Модель Карминати—Феррари—Перего [5] рассматривает усеченный социальный граф, вершинами которого являются пользователи системы, а ребрами — отношения между пользователями. Каждое ребро при этом имеет дополнительный атрибут — уровень доверия между пользователями. Возможность доступа пользователя к объекту определяется набором правил вида *u*, *relation*, *length*, *level*, которые могут быть связаны операциями конъюнкции или дизъюнкции. Каждое из перечисленных правил разрешает пользователю доступ к объекту, если этот пользователь связан с пользователем *u* (владельцем объекта) путем длины не более *length*, каждое из ребер которого соответствует отношению *relation* с уровнем доверия *level*. Таким образом, каждое из условий доступа может использоваться для разрешения или запрета доступа пользователю, связанному с владельцем ресурса произвольным числом повторений одного и того же отношения, например, "друзьям друзей друзей ... друзей". При этом модель рассматривает отношения только между пользователями системы, но не рассматривает отношения между ее ресурсами.

Модель ЛРД, представленная в работах [6, 7], во многом аналогична модели Карминати—Феррари-Перего. Однако в дополнение к возможностям предыдущей модели данная модель предусматривает возможность связывания различными отношениями пользователей и ресурсов системы. Так, например, с фотографией помимо пользователя-владельца может быть ассоциирован список пользователей, изображенных на фотографии. В то же время в данной модели не рассматриваются отношения вида "ресурс—ресурс".

Модели ЛРД, представленные в работах [5—7], обладают тем же важным недостатком, что и ранее рассмотренные модели Фонга. Этот недостаток заключается в том, что данные модели не позволяют учитывать отношения между ресурсами системы. Данный факт ограничивает использование подобных моделей в явном виде в ИАС "Наука—МГУ" ("ИСТИНА").

5. Модели Ченга, Парка и Сандху

В настоящем разделе рассмотрены модели логического разграничения доступа к объектам социальных сетей, представленные в работах [8—11].

Модель ЛРД, представленная в работе [8], является самой простой с позиций выразительности из рассматриваемых в данном разделе. Эта модель получила название *User-to-User Relationship-based Access Control* (UURAC), так как рассматривает разные виды отношений между пользователями системы. Однако следует отметить, что эта модель не рассматривает отношений между ресурсами системы, а единственным отношением, которое может связывать пользователя и ресурс, является отношение "владелец". При этом каждому ресурсу системы соответствует только один пользователь-владелец.

При использовании модели UURAC в системе определяется множество пользователей U, множество возможных отношений между пользователями $\Sigma = \{\sigma_1, ..., \sigma_n, \sigma_1^{-1}, ..., \sigma_n^{-1}\}$ и социальный граф $G = (U, E, \Sigma)$, где $E \subset U \times \Sigma \times U$. Множество вершин социального графа является множеством пользователей, а множество E — множеством ребер, при этом с каждой парой пользователей ассоциировано множество связывающих их отношений. В рассматриваемой модели для каждого отношения $\sigma \in \Sigma$ выделено обратное отношение, т. е. такое отношение $\sigma^{-1} \in \Sigma$, что пользователь u связан с пользователем v

отношением σ в том и только том случае, если пользователь *v* связан с пользователем *u* отношением σ^{-1}.

С каждым объектом в системе ассоциирован пользователь-владелец, который имеет право определять множество пользователей, имеющих право того или иного доступа к этому объекту. Для этого владелец задает некоторые правила, описывающие пути в социальном графе, и пользователь получает доступ к объекту, если в социальном графе есть путь, соединяющий этого пользователя с владельцем объекта и удовлетворяющий заданному правилу. Например, пользователь может предоставить доступ к объекту своим друзьям и друзьям своих друзей. Кроме того, модель UURAC позволяет определять возможные пути отношений между пользователем и владельцем объекта с помощью более сложных правил, учитывающих несколько различных отношений, например, пользователь может разрешить доступ к объекту друзьям друзей коллег своих родителей. В общем случае для задания правил разграничения доступа к объектам предлагается использовать специальный язык, похожий по синтаксису на регулярные выражения.

В работе [8] представлены два алгоритма проверки разрешения на доступ пользователя к объекту, основанные на известных алгоритмах обхода графов — поиске в ширину и поиске в глубину. Экспериментальные оценки производительности, полученные позже в работе [10], показали, что использование алгоритма поиска в глубину намного предпочтительнее использования алгоритма поиска в ширину.

Расширением модели UURAC является модель ЛРД *User-to-Resource Relationship-Based Access Control* (URRAC), представленная в работе [9]. Основным отличием данной модели является возможность предоставлять пользователям доступ к объектам с учетом отношений не только между пользователями, но и между ресурсами системы. При использовании модели URRAC в системе определяется множество пользователей *U*, множество ресурсов *R*, множество отношений $\Sigma = \Sigma_{uu} \cup \Sigma_{ur} \cup \Sigma_{rr}$ и социальный граф $G = (V, E, \Sigma)$, где $V = U \cup R$, $E \subset V \times \Sigma \times V$. Множеством вершин социального графа является множество, состоящее из пользователей и ресурсов системы. Каждому ребру в социальном графе соответствует множество отношений, связывающих вершины, которые соединены этим ребром. Множество Σ_{uu} — это множество отношений, которые могут соединять двух пользователей системы; Σ_{ur} — это множество отношений, которые могут соединять пользователя и ресурс; Σ_{rr} — это множество отношений, которые соединяют между собой ресурсы.

Также как и при использовании модели UURAC, пользователь получает доступ к некоторому ресурсу, если в социальном графе есть путь, удовлетворяющий определенному правилу, по синтаксису напоминающему регулярное выражение. Однако в отличие от случая модели UURAC путь, влияющий на предоставление доступа, может проходить как через пользователей, так и через ресурсы системы.

Другой отличительной особенностью модели URRAC является использование в ней множества правил разрешения конфликтов в случае, если изменять атрибуты безопасности объекта могут пользователи, связанные с ним разными цепочками отношений. Данные правила могут быть следующих трех видов: конъюнктивное (доступ разрешается, если он разрешен всеми пользователями, задавшими правила доступа к объекту); дизъюнктивное (доступ разрешается, если он разрешен хотя бы одним пользователем); приоритетное (правила, заданные одним пользователем, имеют приоритет над правилами, заданными другим пользователем). Заметим, что из перечисленных способов разрешения конфликтов применение приоритетов правил в ИАС "Наука—МГУ" ("ИСТИНА") может быть существенно затруднено тем обстоятельством, что многие объекты могут не иметь единственного пользователя-владельца. Например, зарегистрированная в системе публикация может иметь несколько авторов, которые должны быть равноправны с точки зрения управления ее атрибутами.

В работе [9] не представлено данных о возможной производительности программной реализации модели URRAC. Однако заметим, что при использовании для реализации данной модели алгоритмов, аналогичных представленным для модели UURAC в работе [8], их производительность будет существенно ниже, чем для модели UURAC, в силу того, что в полном социальном графе возможная длина пути между вершинами, как правило, значительно больше, чем в усеченном социальном графе.

Другим расширением рассмотренной ранее модели UURAC является модель ЛРД *Attribute-Aware User-to-User Relationship-Based Access Control* ($UURAC_A$), представленная в работе [11]. Данная модель основана на модели UURAC, однако имеет возможность учитывать дополнительные атрибуты пользователей системы и отношений между пользователями. При этом модель позволяет ассоциировать с пользователями атрибуты произвольного вида. Вторым улучшением модели $UURAC_A$ по сравнению с моделью UURAC является наличие возможности предоставлять пользователю доступ к объекту на основании более общих топологических свойств социального графа, чем существование пути определенного вида. Например, модель $UURAC_A$ позволяет предоставить доступ к объектам пользователям, имеющим не менее пяти общих друзей с владельцем объекта, в то время как модель UURAC имеет возможность принимать решение о доступе только на основании наличия или отсутствия у двух пользователей общих друзей.

Вместе с тем модель $UURAC_A$ в отличие от модели URRAC оперирует только усеченным социальным

графом и не позволяет ассоциировать атрибуты с ресурсами и учитывать отношения вида "пользователь—ресурс" и "ресурс—ресурс".

Таким образом, из перечисленных моделей Ченга—Парка—Сандху только модель URRAC имеет возможность учета отношений между ресурсами системы, которая необходима для управления доступом к объектам ИАС "Наука—МГУ" ("ИСТИНА"). Однако модель URRAC не обладает другим свойством, необходимым для управления доступом к объектам ИАС "Наука—МГУ" ("ИСТИНА"): эта модель не предусматривает учета дополнительных атрибутов отношений. Одновременно с этим модель $UURAC_A$, предполагающая учет дополнительных атрибутов отношений между пользователями системы, не рассматривает отношений между ее ресурсами.

Все три рассмотренные модели предполагают, что пользователь должен иметь возможность задавать достаточно сложные правила для управления доступом к связанным с ним объектам на специальном языке. Такой подход имеет два недостатка. Первый из них заключается в том, что пользователь, не обладающий навыками программирования и познаниями в математической логике и, возможно, теории графов, имеет высокую вероятность допустить ошибку при вводе подобных правил. Второй недостаток заключается в том, что злоумышленник, имеющий возможность формулировать сколь угодно сложные правила для разграничения доступа к объектам системы, может подвергнуть систему атаке на отказ в обслуживании, введя специально сформированные правила, проверка которых потребует значительных затрат вычислительных ресурсов. Кроме того, сложная структура правил проверки доступа с высокой вероятностью приведет к недопустимым для использования в реальных социальных сетях затратам вычислительных ресурсов даже для проверки относительно просто сформулированных правил разграничения доступа при использовании модели URRAC. Отмеченный факт имеет место в связи с тем, что при использовании этой модели по сравнению с моделью UURAC значительно возрастают основные параметры социального графа, от которых зависит производительность используемого в представленных моделях алгоритма поиска в глубину.

Таким образом, при разработке моделей ЛРД к объектам ИАС "Наука—МГУ" ("ИСТИНА") целесообразно значительно ограничить возможность изменения пользователями атрибутов безопасности объектов. Одним из вариантов такого ограничения является выбор из конечного множества вариантов. Например, представляется целесообразным дать возможность пользователю, который является автором статьи, выбирать круг пользователей, имеющих право скачать полный текст статьи, из следующих вариантов: всем пользователям разрешено скачать текст статьи; пользователям, работающим в одном подразделении с одним из авторов, разрешено скачать текст статьи; только авторам разрешено скачать текст статьи. Заметим, что при этом множество пользователей, которым может быть предоставлено право того или иного доступа к объекту, может быть описано конечным множеством фиксированных последовательностей отношений (возможно, включая выделенное фиктивное отношение, связывающее любой объект с любым пользователем).

Заключение

Проведенный анализ моделей ЛРД в социальных сетях позволил оценить возможность их применения к системам, подобным ИАС "Наука—МГУ" ("ИСТИНА"), и выделить целесообразные к использованию в таких системах выразительные инструменты для описания правил разграничения доступа. В частности, результаты такого анализа позволили выполнить следующее уточнение требований к модели ЛРД в ИАС "Наука—МГУ" ("ИСТИНА").

- Модель должна предоставлять пользователю тот или иной доступ к объекту в зависимости от взаиморасположения пользователя и объекта доступа в полном социальном графе, т. е. графе, вершинами которого являются объекты системы, а ребрами — отношения между объектами.
- Модель должна учитывать топологические свойства полного социального графа, т. е. отношения не только между пользователями системы, но также отношения вида "пользователь—ресурс" и "ресурс—ресурс". Все три вида отношений ("пользователь—пользователь", "пользователь—ресурс" и "ресурс—ресурс") должны допускать изменение администратором системы, а не быть фиксированно заданными в модели (как, например, в модели Фонга—Анвара—Жао).
- Модель должна предусматривать возможность указывать права доступа пользователя к объекту на основании не только топологических свойств социального графа, но и дополнительных атрибутов ребер и вершин, входящих в пути между пользователем и объектом доступа.
- Модель должна предусматривать возможность для пользователя изменять некоторые атрибуты безопасности связанных с этим пользователем объектов. При этом пользователь не должен иметь возможности формулировать сколь угодно сложные правила предоставления доступа. Система должна предоставлять пользователю возможность выбирать из задаваемого администратором списка наборов правил.
- Модель должна предусматривать возможность использования в системе транзитивных отношений, т. е., если отношение r определено как транзитивное, и согласно записям в используемой базе данных объект o_1 связан отношением r с объектом o_2, а объект o_2 связан с объектом o_3, то механизм принятия ре-

шения о доступе пользователя к объектам должен считать объекты o_1 и o_3 связанными отношением r, даже если в базе данных они не связаны никаким отношением.

Кроме того, разрабатываемая модель ЛРД для ИАС "Наука—МГУ" ("ИСТИНА") должна учитывать ряд свойств, отражающих специфику этой системы и отличающих ее от социальных сетей. В частности, важной особенностью ИАС "Наука—МГУ" ("ИСТИНА") оказалось то, что большинству объектов в ней не может быть поставлен в соответствие единственный пользователь-владелец. Это обстоятельство затрудняет создание правил разрешения конфликтов доступа, которые возникают в случаях, если два пользователя, имеющие право управления доступом других пользователей к объекту, пытаются назначить ему противоречивые метки доступа.

В настоящее время работы по совершенствованию модели ЛРД к объектам ИАС "Наука—МГУ" ("ИСТИНА") проводятся с учетом отмеченных свойств, отражающих специфику этой системы, а также перспективных механизмов других моделей ЛРД для социальных сетей, краткий анализ которых представлен в настоящей статье.

Список литературы

1. **Садовничий В. А., Афонин С. А., Бахтин А. В. и др.** Интеллектуальная система тематического исследования научно-технической информации ("ИСТИНА"). М.: Изд-во МГУ, 2014. 262 с.

2. **Fong P., Anwar M., Zhao Z.** A privacy preservation model for Facebook-style social network systems // ESORICS 2009: Proceedings of 14th European Symphosium on Research in Computer Security. Saint-Malo, France. 21—23 September, 2009. Springer, 2009. P. 303—320.

3. **Fong P.** Relationship-Based Access Control: protection model and policy language // Proceedings of the first ACM conference on Data and application security and privacy, CODASPY'11. New York, USA, 2011. ACM, 2011. P. 191—202.

4. **Fong P., Siahaan I.** Relationship-Based Access Control Policies and Their Policy Languages// Proceedings of the 16th ACM Symposium on Access Control Models And Technologies, SACMAT'11. New York, USA, 2011. ACM, 2011. P. 51—60.

5. **Carminati B., Ferrari E., Perego A.** Enforcing Access Control in Web-Based Social Networks // ACM Trans. Information System Security. 2009. Vol. 13, N. 1. P. 1—38.

6. **Carminati B., Ferrari E., Heatherly R., Kantarcioglu M., Thuraisingham B.** A semantic web based framework for social network access control // Proceedings of the 14th ACM symphosium on Access control models and technologies, SACMAT'09. New York, USA, 2009. ACM, 2009. P. 177—186.

7. **Carminati B., Ferrari E., Heatherly R., Kantarcioglu M., Thuraisingham B.** Semantic web-based social network access control // Computers and Security. 2011. Vol. 30, N. 2—3: Special Issue on Access Control Methods and Technologies. P. 108—115.

8. **Cheng Y., Park J., Sandhu R.** A user-to-user relationship-based access control model for online social networks // Proceedings of the 26th IFIP Annual WG 11.3. Conference on Data and Application Security and Privacy (DBSec'12) 2012. P. 8—24.

9. **Cheng Y., Park J., Sandhu R.** Relationship-Based Access Control for Online Social Networks: Beyond User-to-User Relationships // PASSAT 2012. IEEE, 2012. P. 646—655.

10. **Cheng Y.** Access control for online social networks using relationship type patterns. Dissertation, Doctor of Philosophy in Computer Science. The University of Texas in San Antonio, TX, USA, 2014. URL: http://www.profsandhu.com/dissert/Dissertation_Cheng.pdf

11. **Cheng Y., Park J., Sandhu R.** Attribute-aware Relationship-based Access Control for Online Social Networks // Data and Applications Security and Privacy XXVIII. Lecture Notes in Computer Science. 2014. Vol. 8566. P. 292—306.

V. A. Vasenin, Professor, Head of Department, e-mail: vasenin@msu.ru,
A. A. Itkes, Researcher, e-mail: itkes@imec.msu.ru,
K. A. Shapchenko, Senior Researcher, e-mail: shapchenko@iisi.msu.ru, Institute of Mechanics, Lomonosov Moscow State University

On the Application of Social Networking Access Control Models to One Class of Multi-User Content Management Systems

Many recent multi-user content management systems share similarities with social networking services because of the number of users, user and group dynamics, user involvement in generating and modifying the content. A notable challenge in such systems is often to specify access control rules for different resources in an efficient manner. Traditional access control models, although providing the necessary functions, tend to lead to excessive amount of administrative actions in certain cases related to rearrangement of the user structure including adding users and groups, reassigning users to different groups, and performing other maintenance tasks. Thus, introduction of a more declarative approach to the specification of access control rules may be a viable solution.

Given the noted similarities several state-of-the-art formal access control models for social networks are analyzed for their potential applications in the content management systems under study. A system for managing research information and science metrics is selected as an example of target content management system, namely the system "Nauka—MGU" ("ISTINA") [1]. This target system is used in Lomonosov Moscow State University for user-assisted collection and analysis of data about publications, research projects, and teaching activities. This system actively uses a set of access rules to prevent accidental mistakes during editing of the entered information and also to restrict access to certain analysis data based on attributes of the user and the resource being accessed.

Three families of formal access control models for social networks are analyzed, including models by P. Fong et al. [2—4], by B. Carminati et al. [5—7], and models UURAC, URRAC and UURAC$_A$ by Y. Cheng, J. Park, and R. Sandhu [8—11]. Two features of access control rules specification which are particularly important for the target system but not present in the analyzed models at the same time were noted: use of the full social graph with user-to-resource and resource-to-resource relations in addition to the more common user-to-user relations; and the use of attributes for entities and relations as the means of introducing parameters in access rules.

In the conclusion of the paper a refined requirements list is provided for the access control model for the target system, including requirements for specifying access control rules using the full social graph of the system, using attributes of objects and relations, accounting for transitive relations, and performing restricted discretionary user-level configuration of the relevant access rules.

Keywords: *access control, access rule, access control model, content management system, social network, research information, science metrics*

References

1. **Sadovnichii V. A., Afonin S. A., Bakhtin A. V., Bukhonov V. Yu., Vasenin V. A., Gankin G. M., Gasparyants A. E., Golomazov D. D., Itkes A. A., Kozitsyn A. S., Tumaikin I. N., Shapchenko K. A**. Intellektual'naya sistema tematicheskogo issledovaniya nauchno-tekhnicheskoi informatsii ("ISTINA") (Intellectual system for topical analysis of scientific and technical information "ISTINA"). Moscow: Izd-vo MGU, 2014. 262 p. (in Russian).

2. **Fong P., Anwar M., Zhao Z.** A privacy preservation model for Facebook-style social network systems. *ESORICS 2009: Proceedings of 14th European Symphosium on Research in Computer Security*. Saint-Malo, France, 21—23 September, 2009. Springer, 2009, pp. 303—320.

3. **Fong P.** Relationship-Based Access Control: protection model and policy language. *Proceedings of the first ACM conference on Data and application security and privacy*, CODASPY'11, New York, USA, 2011. ACM, 2011, pp. 191—202.

4. **Fong P., Siahaan I.** Relationship-Based Access Control Policies and Their Policy Languages. *Proceedings of the 16th ACM Symposium on Access Control Models And Technologies*, SACMAT'11, New York, USA, 2011. ACM, 2011, pp. 51—60.

5. **Carminati B., Ferrari E., Perego A.** Enforcing Access Control in Web-Based Social Networks. *ACM Trans. Information System Security*, 2009, vol. 13, no. 1, pp. 1—38.

6. **Carminati B., Ferrari E., Heatherly R., Kantarcioglu M., Thuraisingham B.** A semantic web based framework for social network access control. *Proceedings of the 14th ACM symphosium on Access control models and technologies*, SACMAT'09, New York, USA, 2009. ACM, 2009, pp. 177—186.

7. **Carminati B., Ferrari E., Heatherly R., Kantarcioglu M., Thuraisingham B.** Semantic web-based social network access control. *Computers and Security,* 2011, vol. 30, no. 2—3: Special Issue on Access Control Methods and Technologies, pp. 108—115.

8. **Cheng Y., Park J., Sandhu R.** A user-to-user relationship-based access control model for online social networks. *Proceedings of the 26th IFIP Annual WG 11.3 Conference on Data and Application Security and Privacy* (DBSec'12), 2012, pp. 8—24.

9. **Cheng Y., Park J., Sandhu R.** Relationship-Based Access Control for Online Social Networks: Beyond User-to-User Relationships. *PASSAT 2012. IEEE*, 2012, pp. 646—655.

10. **Cheng Y.** Access control for online social networks using relationship type patterns. Dissertation, Doctor of Philosophy in Computer Science. The University of Texas in San Antonio, TX, USA, 2014, available at: http://www.profsandhu.com/dissert/Dissertation_Cheng.pdf

11. **Cheng Y., Park J., Sandhu R.** Attribute-aware Relationship-based Access Control for Online Social Networks. *Data and Applications Security and Privacy XXVIII. Lecture Notes in Computer Science*, 2014, vol. 8566, pp. 292—306.

ИНФОРМАЦИЯ

Институт прикладной математики им. М. В. Келдыша РАН

проводит в г. Новороссийск, пос. Абрау-Дюрсо
с 21 по 26 сентября 2015 г.

XVII Всероссийскую научную конференцию

«Научный сервис в сети Интернет»

Конференция посвящена основным направлениям и тенденциям использования интернет-технологий в современных научных исследованиях. Основная цель данной конференции — предоставить возможность для обсуждения, апробации и обмена мнениями о наиболее значимых результатах, полученных ведущими российскими учеными за последнее время в данной области деятельности.

Сайт конференции: http://agora.guru.ru/abrau2015

УДК 004.[021+023]

Н. Н. Ефанов[1], мл. программист, e-mail: nefanov@parallels.com,
А. Л. Мелехова[1,2], вед. программист, преподаватель, e-mail: annam@parallels.com,
А. О. Бондарь[1], мл. программист, e-mail: abondar@parallels.com,
[1]ООО "Параллелз", Москва,
[2]Московский физико-технический институт (МФТИ)

Алгоритмы динамического управления энергопотреблением в облачной системе

Вопросы сокращения энергозатрат вычислительных систем в настоящее время очень актуальны. В связи с активной миграцией ресурсоемких информационно-вычислительных приложений в облачные системы к их энергоэффективности стали предъявляться повышенные требования. Цель данной работы — описать во всей полноте стоящие на этом направлении задачи и представить в обобщенном виде наиболее интересные и распространенные их решения. Рассмотрены различные алгоритмы управления энергопитанием облачных систем и представлены результаты их сравнительного анализа.

Ключевые слова: *энергопотребление, виртуализация, облачные вычисления, динамическое управление энергопотреблением*

1. Постановка задачи

В последние годы прослеживается возрастающий интерес исследователей к вопросам, связанным с энергопотреблением вычислительных систем [1—10]. Расходы на электроэнергию, необходимые для работы центров для хранения и обработки данных (датацентров), неуклонно растут. В США, например, их значение достигает 3 % суммарных затрат на электроэнергию по всем отраслям [11]. Это обстоятельство делает разработки в области энергоэффективности в высокой степени востребованными.

Динамическим управлением энергопотреблением (*Dynamic power management — DPM*) называют процесс реализации набора политик (наперед принятых правил и действий), имеющих своей целью снижение энергетических расходов при эксплуатации вычислительной системы и реализованных по средствам алгоритмов перераспределения ресурсов и переключения режимов работы системы[1]. В случае облачной архитектуры *DPM* реализуется как на системном, так и на локальном уровнях [1]. Локальные политики концентрируются на изменении режима работы отдельно взятой машины на базе некоторой предполагаемой модели ее поведения [3, 12]. Как правило, такие политики направлены на борьбу с динамическими потерями энергии, которые происходят при изменении состояния устройств. Обычно они реализуются путем переключения триггеров, составленных на КМОП-транзисторах, и используют в качестве входных данных частоту работы устройства, напряжения и т. д., не учитывая стационарные Джоулевы потери. В свою очередь политики системного уровня занимаются перераспределением всех доступных вычислительных ресурсов аппаратного комплекса между виртуальными машинами (ВМ), работающими в рамках этой инфраструктуры. Повышение энергоэффективности достигается уплотнением виртуальных машин на физических машинах (ФМ), при котором разгруженные ФМ выключаются или переходят в состояния низкого энергопотребления. В облачных системах локальная составляющая *DPM* является низкоуровневой подзадачей системной *DPM*.

При разработке политик и алгоритмов *DPM* исследователи встречают ряд ограничений, значительно усложняющих задачу нахождения эффективных решений для управления облачной системой. К таким ограничениям относят:

- трудности учета и корректной оценки затрат на живую миграцию (*live migration*) ВМ с одной ФМ на другую [13, 14];
- особенности конфигураций конкретных облачных инфраструктур (например, некоторые ВМ жестко связаны с определенной ФМ и не могут участвовать в процессе миграции [1]);
- существенную вариативность нагрузки, а именно: различное программное обеспечение, исполняемое на ВМ, выдает нагрузки неоднородного характера, что усложняет моделирование системы [4—6, 14];

[1] Далее в статье авторы во многом говорят о задаче *DPM*, которую решают. В противовес самому *DPM*, который есть реализованный процесс управления на базе решенной задачи.

• потенциальные конфликты между алгоритмами *DPM* и алгоритмами балансировки нагрузки, которые направлены на поддержание высокой производительности системы (низкого времени отклика, качества обслуживания и т. д.);

• высокую сложность общего решения *DPM* как оптимизационной задачи, которая без конкретизации некоторых свойств системы сводится к необходимости решения комбинаторных *NP*-трудных задач большой размерности [1, 2].

Таким образом, поиск эффективного решения *DPM* актуален, несмотря на весь объем проведенных исследований [1, 2].

В данной работе представлен анализ ряда существующих подходов к решению задачи *DPM* с учетом всех перечисленных ограничений. Ввиду сложности их описания имеет смысл разбить *DPM* на следующие подзадачи:

♦ выбор момента сбора статистики и анализа данных;

♦ оценка загруженности ВМ и ФМ на некоторый период времени;

♦ миграция ВМ;

♦ балансировка нагрузки, поиск лучшего расположения, исходя из критерия снижения энергопотребления;

♦ контроль целевого диапазона;

♦ локальный *DPM* на конкретной ФМ как на конечном элементе системы.

Решения перечисленных подзадач интегрируются в общее решение *DPM облачной системы*. Выбор момента сбора статистики в абсолютном большинстве систем происходит эмпирически — один раз в заранее установленный интервал времени. Сначала собирают данные. На их основании происходит оценка загруженности ВМ и ФМ. В случае неоптимальной конфигурации с учетом мощностных и временных затрат на миграцию ВМ может быть принято решение о миграции. В этот же момент может быть скорректирован целевой диапазон. Далее происходит собственно процесс миграции. На завершающей стадии применяют локальные политики *DPM*.

В рамках настоящей статьи детально рассмотрена каждая из описанных подзадач. При этом авторы ограничились лишь кратким изложением вопросов миграции и балансировки нагрузки.

2. Терминология

Введем понятия, которыми будем оперировать в данной статье. Рассматриваемая система состоит из набора физических машин: $ФМ_j$, $j \in \{1, ..., N\}$; виртуальных машин $ВМ_i$, $i \in \{1, ..., M\}$, которые исполняются на этих физических машинах. В рамках решения задачи *DPM* выбирается некоторый промежуток времени T, на котором проводится анализ поведения системы. Зададим разбиение на n интервалов, определив границы разбиений: t_k, $k = 0...n$. В рамках каждого интервала разбиения система считается квазистационарной (характерные величины не меняются). Таким образом, для вводимых величин:

$$\forall k \in [1...n],\ \ t \subset [t_{k-1}, t_k],\ \ f(t) = f(t_{k-1}).$$

Назовем виртуальную машину $ВМ_i$ расположенной на физической машине $ФМ_j$ в момент времени t_k, $k = 1...n$, если в течение $t \in [t_{k-1}, t_k]$ $ВМ_i$ исполняется на $ФМ_j$ или рекомендована к миграции на нее. Нагрузка $V_i(t)$ — величина, характеризующая выполняемые в некоторый момент времени задачи на виртуальной машине $ВМ_i$. Суммарную нагрузку будем описывать опуская нижний индекс.

Выбор упомянутого выше промежутка времени T может сильно повлиять на качество результатов оптимизирующих действий. По этой причине для определения целесообразности выбора T для конкретной конфигурации системы вводится величина $K(t)$, характеризующая нагрузку в системе:

$$K(t) = \max\left(K(t,i)\right),$$

$$K(t,i) = \frac{\left(V_i(t) - V_i(t_0)\right)}{V_i(t_0)},\ i = 1...M.$$

Критерий целесообразности выбора T по $K(t)$ существенно зависит от выбранного подхода к решению задачи *DPM* и потому не имеет общего описания.

В существующих вычислительных системах крайне остро стоят вопросы измерения мощности энергетических затрат, связанных с функционированием системы. В некоторых случаях достаточно легко измерить мощность этих затрат, а в других случаях прямое измерение фактически невозможно. Поэтому в рамках задачи *DPM* разделяют два типа мощностей энергетических затрат — реальные (измеряются в джоулях) и абстрактные (некая весовая величина). Часто между двумя этими величинами существует функциональная зависимость [4].

Реальную мощность энергетических затрат физической машины обозначим как $ФМ_j p(t)$. Она является суммой мощности динамических затрат $v_j(t)$ и статических затрат $F_j(t)$. Такое разделение предполагает, что статическое энергопотребление возникает на включенной $ФМ_j$ вне зависимости от нагрузки на машине, в то время как динамическое является следствием исполнения ВМ. Абстрактную мощность энергетических затрат, вызванных исполнением $ВМ_i$ на этой физической машине $ФМ_j$, обозначим $P_{j,i}(t)$. На рис. 1 изображено поведение системы в течение двух последовательных характерных отрезков времени T_1 и T_2 с точки зрения системного *DPM*.

Введем понятие расположения l, описывающего как ВМ распределены по ФМ (рис. 2). Удобно описывать l с помощью бинарной матрицы $[M \times N]$, (i, j)-й элемент которой указывает на принадлежность $ВМ_i$ к $ФМ_j$; bl — лучшее расположение, sl — стартовое расположение.

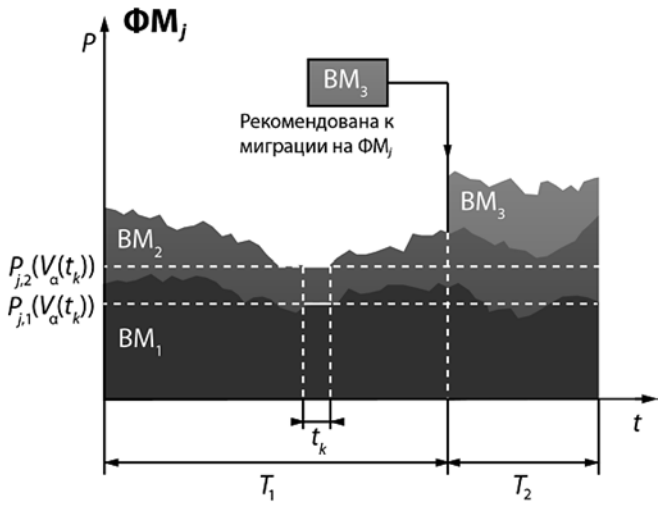

Рис. 1. Динамика нагрузки на ФМ, исполняющей несколько ВМ

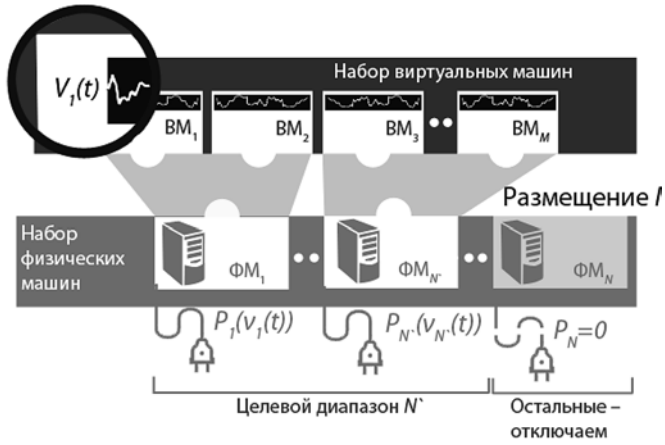

Рис. 2. Размещение ВМ на ФМ и целевой диапазон

Часто возникает ситуация, что в силу специфики[2] облачной системы BM_i может исполняться только на $ФМ_j$. В этом случае BM_i называют привязанной к $ФМ_j$ и фиксируют ее расположение в рамках задачи оптимизации.

В каждый момент времени часть ФМ может быть выключена и на них не могут исполняться ВМ. Часть машин будем считать постоянно включенными, их набор будем называть опорным множеством *n*. Включенные в текущий момент времени ФМ назовем целевым диапазоном *N'*(рис. 2).

Применительно к ФМ будем использовать терминологию из области локального *DPM*. Обозначим U_{tj} долю загрузки процессора $ФМ_j$ (100 %-ная загрузка процессора = 1). При этом $F_j(t)$, т. е. статическая нагрузка, не зависит от U_{tj}. Процессор может находиться на разных уровнях энергопотребления. В общем случае у процессора присутствует как минимум одно активное состояние (*Pn-state*, где $n = 0,1...$) для исполнения инструкций и обработки запросов, а также некоторое число состояний пониженного энергопотребления (*Cn-states*, где $n = 0,1...$). Переход из активного состояния в сторону пониженного происходит аппаратными (по тайм-ауту) или программными (по инструкции) средствами. Возврат в активное состояние связан с обработкой внешнего запроса. Переходы $ФМ_j$ между состояниями характеризуются временами перехода $t_{down,\,s}/t_{up,\,s}$ в состояния с меньшим/большим энергопотреблением и соответствующими мощностями потерь $p_{down,\,s}$ и $p_{up,\,s}$. Здесь *s* "пробегает" по набору состояний, работа с которыми описана подробнее в разд. 5 и 6, а также в подразд. 3.3 и 3.5. Также вводится параметр $t_{penalty}$ ($t_{penalty} = t_{up,\,s}$, если процессор находится в состоянии *s*), который указывает, какое количество времени будет затрачено процессором на пробуждение из спящего режима.

Для решения задачи локального *DPM* в первую очередь рассматривают значение $ФМ_j p(t)$. Нижний индекс указывает на различные значения мощности, например, применительно к отдельной ФМ:

- $ФМ_j p_{cluster}(t)$ — потребляемая устройством мощность, функция от нагрузки;
- $ФМ_j p_{peak}(t)$ — величина, средняя по пиковым мощностям;
- $ФМ_j p_{idle}(t)$ — величина, средняя по мощностям, потребляемым в режиме простоя.

В ряде случаев стоимость единицы электроэнергии может существенно различаться. Используем для ее обозначения коэффициент c_p.

Источник полезной нагрузки облачной системы — запросы, сгенерированные клиентами-пользователями. Этот процесс характеризуется двумя величинами: $\alpha_{zj}(t)$ и $\phi_{zj}(t)$, где $\phi_{zj}(t)$ — число ресурсов ВМ, размещенной на $ФМ_j$; $\alpha_{zj}(t)$ — число запросов *z*-го клиента. Просуммировав $\alpha_{zj}(t)$ по *j*, получим

Рис. 3. Обработка запросов (t_{in} и t_{out} — моменты поступления и конца обработки запроса соответственно)

[2] Специфика состоит в конфигурации ВМ и в возможности реализовать данную конфигурацию на ФМ. Так, если в параметрах ВМ обозначен процессор поколения Intel Nehalem, а в парке ФМ, управляемых облачной системой, присутствует лишь одна машина данного поколения, то вероятна привязка.

число запросов на ФМ в интервал времени t, а просуммировав по z и посчитав среднее арифметическое по j, получим величину α, используемую при определении целевого множества из N' ФМ, которые в данный момент работают [7, 8]. Контроль целевого диапазона и его вариаций будет рассмотрен в разд. 5. Для контроля нагрузки в этой задаче удобно пользоваться величинами α и α_d — частотой приходов задач и их средней частотой завершения соответственно, а также β — средним временем выполнения задач (рис. 3).

3. Оценка загруженности систем

Рассмотрим различные методы оценки загруженности системы. Эта оценка является одной из ключевых подзадач *DPM*, так как на ее основании будет происходить поиск лучшего расположения и осуществляться миграция.

3.1. Среднее по кванту времени

Алгоритм "*среднего по кванту времени*" основан на усреднении нагрузки по интервалу времени T, переменному в ходе работы [9]. В начальный этап облачная система находится в состоянии с некоторым стартовым расположением виртуальных машин *sl*. В соответствующем расположении можно оценить нагрузку ВМ_i на отрезке времени $[t_{k-1}, t_k]$.

Средняя нагрузка за время T вычисляется по формуле

$$\bar{V}_i(j,T) = \frac{1}{T}\sum_{k=1}^{n} V(j,k)(t_k - t_{k-1}).$$

Тогда средняя суммарная нагрузка на отрезке времени $[t_{k-1}, t_k]$ на ФМ_j, равна

$$p_j(i,T) = \sum_{i=1}^{m_j} \bar{V}_i(j,T),$$

где суммирование ведется по ВМ, расположенным на ФМ_j.

Средняя суммарная нагрузка ВМ за время T вычисляется суммированием по средним для $\text{ВМ}_i(T)$.

К сильным сторонам алгоритма следует отнести стабильность при $V_i(k) = \text{const}$ на каждом из квантов времени $[t_{k-1}, t_k]$. В этом случае по дисперсии легко оценить возможную нагрузку при различных распределениях величины дополнительного спроса по машинам. К недостаткам алгоритма стоит отнести неоптимальные результаты при быстро изменяющихся нагрузках *BM*, где $V_i(k) \neq \text{const}$ на $[t_{k-1}, t_k]$. В этом случае для улучшения результатов необходимо "измельчение" временной сетки или усложнение ее структуры. Изначально такой подход был применен в рамках реализации генетического алгоритма балансировки нагрузки компьютерной системы [9], но он может быть использован достаточно широко, в том числе в *DPM*.

3.2. Оценка нагрузки физической машины как конечного узла системы

Оценивая независимо нагрузку на каждой из ФМ и зная расположение и загруженность ВМ, мы получаем данные, необходимые для осуществления *DPM*. После предварительной калибровки коэффициентов модели, связывающей нагрузку в системе со значениями энергопотребления на конкретной ФМ, алгоритм позволяет получать прогноз мощности реальных энергетических затрат в произвольный момент времени [4].

Для U_t — загрузки процессора ФМ:

$$F_j(t) = (\text{ФМ}_j p_{idle}(t) + (PUE - 1)\text{ФМ}_j p_{peak}(t)),$$

где PUE — коэффициент полезного действия (КПД) ФМ,

$$v(U_t) = \left(\text{ФМ}_j p_{peak} - \text{ФМ}_j p_{idle}\right)\left(2U_t - U_t^r\right),$$

где r — эмпирическая константа. В ряде источников указано [4], что $r \approx 1$ приемлемо.

Тогда

$$\text{ФМ}_j p_{cluster}(U_t) = F_j(t) + v(U_t) + e,$$

где e — аппаратная константа (эмпирическая). В идеале параметры следует выбрать так, чтобы $\text{ФМ}_j p_{cluster}(0) = 0$.

К преимуществам метода можно отнести его простоту и возможность широкого применения в структурах с однородными ФМ. В комбинации с методами системного уровня рассмотренный алгоритм позволяет решить проблему оценки нагрузки в произвольный момент времени. Недостатком метода является необходимость калибровать модель для каждой конкретной ФМ: ($p_{cluster}$, p_{peak}, p_{idle}, e, r) [5, 6]. Результаты применения описанного метода на тестовом стенде приведены в работе [4], теоретические обоснования даны в работе [10].

3.3. Оценка загруженности на основе теории возобновления

В данном подходе изменение состояния ВМ представляют как случайную величину, имеющую стационарное распределение [3]. Под состоянием ВМ подразумевается прежде всего уровень энергопотребления, но в некоторых случаях могут использоваться и другие характеристики (например, число полученных сетевых пакетов). Число пришедших на исполнение запросов в некоторый момент времени также рассматривается как случайная величина. Время пребывания в том или ином состоянии и частота переходов между ними определяют потери энергии с точностью до калибровочной эмпирической константы. В идеальном случае калибровочная константа должна быть равна нулю при сохранении состояния и достигать максимума при переходах. Время для каждого состояния и перехода рассчитывается исходя из следующих особенностей.

Время простоя (*idle state*) является средним значением между временем от окончания обработки предыдущего запроса до прихода нового запроса и временем дополнительного простоя при переходе в режим пониженного энергопотребления (*low-power state*).

Время перехода в режим пониженного энергопотребления и *время перехода в активное состояние* оценивают из математического ожидания случайной величины — времен переходов, известных и полученных на основании предварительного анализа. Способы расчета времени, проведенного в каждом из состояний, и частоты переходов приведены в работе [3].

Алгоритм дает возможность получить точные оценки и исследовать эволюцию системы, оперируя малым количеством показаний в текущий момент. Однако при этом в системе должно быть выполнено стационарное распределение случайной величины, характеризующей смену состояний, что является ограничением. Ряд нестационарных подходов представлен в описании алгоритмов локальной политики, основанных на этом методе (разд. 6). Данный подход в применении к задаче управления жесткими дисками дал хорошие результаты [3]. Теоретическое обоснование приведено в математическом труде Doob [15].

3.4. Оценка по средствам полумарковской модели

В рамках метода *TISMDP* (*Time-Indexed Semi-Markov Dynamic Policy*) на основе полумарковской модели используется оценка, приведенная в работе [3]. Эта оценка обобщает Марковские процессы принятия решений, позволяющие выбирать действия при изменении состояний системы. Использование этого подхода к локальной оптимизации дано в подразд. 6.3. Преимуществом метода является его применимость к непрерывному моделированию эволюции системы с учетом обобщения полумарковских подходов на произвольное распределение вероятностей. Симуляция [3] демонстрирует высокие показатели энергоэффективности в случае экспоненциального распределения случайной величины, характеризующей смену состояний.

3.5. Оценка из метода "Адаптивное дерево обучения для решения локального DPM"

В данном методе проводится оценка затрат электроэнергии на переход на заданный уровень энергопотребления и потери при дополнительном простое [12].

Рассмотрим промежуток t_{stat} пребывания в одном из состояний ФМ с n режимами пониженного энергопотребления и одним режимом полной нагрузки ($n + 1$) с соответствующими мощностями $p_0, ..., p_n$ и известными параметрами переходов $(t, p)_{s,j}$. Тогда при известном t_{stat} для s-го состояния затраты энергопотребления C можно оценить следующим образом:

$$C_s = t_{down,s}p_{down,s} + t_{up,s}p_{up,s} + (t_{stat} - t_{down,s} - t_{up,s})p_s.$$

При этом $C_{i+1} \leqslant C_i$. Согласно алгоритму вычисления проводятся последовательно для каждого состояния, начиная с 0-го ($s = 0, ..., n$).

В рамках метода дана рекомендация объединения промежутков t_{stat} в целях уменьшения паразитных потерь при частом переключении состояний.

Сложность данного алгоритма составляет $O(n)$ в случае заранее рассчитанных затрат на переход. Недостатком метода являются необходимость калибровки по большому числу параметров и сложность этой калибровки [16].

4. Поиск лучшего расположения

На основании оценки загруженности системы, полученной с помощью алгоритмов, представленных в разд. 3, и оценки затрат на живую миграцию необходимо построить оптимальное расположение ВМ на ФМ — *bl*. Следует отметить, что многие разработки игнорируют накладные расходы на живую миграцию, что существенно снижает возможности их применения в реальных системах.

4.1. Поиск лучшего расположения методом динамического программирования

Некоторые подходы к *DPM* как к оптимизационной задаче позволяют снижать ее вычислительную сложность. Например, можно применить метод динамического программирования (ДП) при фиксировании требований (входящих запросов) α_{ij} к ВМ на каждом шаге. При этом алгоритмическая сложность уменьшается ($NP \Rightarrow P$), а задача сводится к выпуклой задаче линейного программирования [17, 18], в результате решения которой будет получена оптимальная конфигурация системы с учетом *DPM* и определены затраты C, как функция от мощностей, издержек миграции и перераспределения ресурсов, а также инертности отклика системы [19]. Далее опишем общую схему работы алгоритма.

Инициализация. Метод опирается на предположение, что сервисное время по сети распределено экспоненциально.

ВМ делятся на 4 группы:

1) уходящие с обработки в текущий промежуток времени;

2) обрабатываемые, с убывающей частотой запросов;

3) обрабатываемые, с возрастающей частотой запросов;

4) не обрабатываемые в предыдущий промежуток времени.

Виртуальные машины, обрабатываемые в предыдущем промежутке времени, включены в три первые группы. Далее проводится сортировка по группам по возрастанию требований ВМ к системе (мощнейшая — первая и т. д.). Начальная конфигурация известна. Группы рассматриваются в порядке возрастания номера.

Ввиду привязки ВМ к клиентам можно рассуждать в терминах запросов к ФМ, а именно — сколько ресурсов следует выделить клиентской задаче, исполняемой на данной ФМ (в контексте соответствующей ВМ).

На начальном этапе возможно произвольное расположение ВМ.

Исполнение. В данном подходе выполняется поиск оптимальной конфигурации ВМ клиента на серверах с помощью метода динамического программирования. На каждой итерации внутреннего цикла α_{zj} фиксирован и на каждом шаге цикла изменяется на *ga* — мелкость разбиения. При этом оптимальные ресурсы ϕ_{zj} определяются из условий Куна—Каруша—Таккера [17] для данной выпуклой оптимизационной задачи с учетом энергозатрат, затрат на миграцию и затрат на перераспределение ресурсов (математическое обоснование представлено в работе [18]).

На рис. 4 представлен псевдокод (применительно к ресурсам процессоров).

С учетом выделения ресурсов, миграций и отклика системы потери вычисляются по следующей формуле:

$$C\left(j,\alpha_{zj}\right) = c_p T\left(p_j^{static} + p_j^{dynamic}\right)\phi_{zj} + d_{zj}C(M) + \\ + T\delta\beta_i\alpha_{zj}\lambda_z \exp\left\{-\left(C_j^{p(\max)}\phi_{zj}\beta_{zj} - \alpha_{zj}\lambda_z^{apriori}\right)R_z^{CPU}\right\},$$

где R_z^{CPU} — время отклика процессора на z-й задаче; $\delta\beta_z$ — временные потери при запросе; $C(M)$ — затраты на миграцию; d_{zj} — бинарная матрица необходимости рассмотрения миграции задачи с j-й ФМ (задается как параметр модели); λ_z — частоты приходов новых задач от клиента, а $\lambda_z^{apriori}$ — априорная оценка этой величины; β_{zj} — среднее время обслуживания задачи на ФМ$_j$; $p_j^{dynamic}$, p_j^{static} — динамическая и статическая составляющие мощности на ФМ$_j$; $C_j^{dynamic}$ — затраты энергопотребления на ФМ$_j$, а $C_j^{p(\max)}$ — значение этого параметра при максимальной мощности. Теперь нижний индекс z означает принадлежность задачи к клиенту.

Улучшенный метод позволяет целиком решить задачу *DPM* на системном уровне за (в худшем случае) полиномиальное время.

В случае запросов от нескольких клиентов может потребоваться дополнительная корректировка на сервере. Решим в условиях предыдущего алгоритма выпуклую задачу:

$$\min\left(\begin{matrix} c_p p_j^{dynamic}\sum\limits_{i\in I_j}\phi_{ij} + \sum\limits_{i\in I_j}\delta\beta_i\alpha_{ij}\lambda_i^{apriori}\times \\ \times\exp\left(-\left(C_j^{dynamic}\right)\phi_{ij}\beta_{ij} - \lambda_i^{apriori}\right)R_i^{CPU}\end{matrix}\right)$$

```
1 ga = мелкость разбиения
2 for (j = 1..N) # Проход по всем ФМ
3     for (α_zj = (1/ ga)…1) ) # Постепенная, «парциальная» нагрузка ФМ
4      φ_ij = оптимальные ресурсы из условий Куна—Каруша—Таккера
5           calc(C(j, α_zj)) # Расчёт затрат для выбранной ФМ
6     end
7 end
8 Y = N
9 for (j=1…N)  # Проход по всем ФМ
10    for(x=1...ga) # Постепенная, «парциальная» нагрузка ФМ
11          D[x,y] = «infinity» # Матрица ДП (динамического программирования)
12          for (z = 1 ...x)  # «Парциальный» расчёт минимальных значений
13                D[x,y] = min(D[x,y],D[x-1,y-z]+calc(C(j, z/ga))) # Оптимальнее
                                  #оставить текущее значение или перебросить
                                  # часть задач  на ФМ(y-z) на j
14          D[x,y]=min(D[x,y],D[x-1,y])     # Оставить предыдущее значение или
                                            # вновь рассчитанное оказалось оптимальнее?
15    end
16 end
17 Backtracking # Откат как элементарный шаг ДП
18 out: α_zj^best, φ_zj^best
19 submodule calc (C( j, α_zj))# Подмодуль вычисления C
```

Рис.4. Псевдокод (применительно к ресурсам процессора)

относительно

$$\phi_{zj} \geqslant l_{zj}\left(\left(\alpha_{zj}\lambda_z^{apriori} - \ln\left(h_z^{CPU}/R_z^{CPU}\right)/\beta_{zj}c_j\right)\right)\forall z \in$$
$$\in I_j\Sigma_{z\in I_j}\phi_{zj} \leqslant 1,$$

где для компактности введены следующие параметры: l_{zj} — расположение задачи на ФМ$_j$; I_j — множество ВМ на j-й ФМ; h_z^{CPU} h_z — возможная временная задержка. Символ принадлежности z данному множеству означает исполнение данной задачи на одной из ВМ в этом множестве.

Подметод можно реализовать на уровне конкретного сервера, без усложнения составляющей верхнего уровня. Применение данных алгоритмов к моделированию на основе данных Amazon EC2 [20] дает хорошие результаты. Алгоритмы в несколько ином виде, при котором вместо единого множества ВМ рассматриваются группы ВМ, принадлежащих каждому клиенту, впервые представлены на симпозиуме IEEE/ACM в 2012 г. [19].

4.2. Эвристический алгоритм поиска лучшего расположения

В архитектуре GreenCloud [21] используется жадный алгоритм поиска лучшего расположения *bl*, рассматривающий единичные миграции из текущего состояния *l* с определением затрат на живую миграцию и оценку загруженности на каждом шаге. У графа поиска вершины — расположения, ребра — единичные миграции. Сильной стороной алгоритма является гибкость за счет использования эвристик. Слабой стороной является неоднозначность выбора базовой ФМ, вследствие чего вероятен выбор лишь локального оптимума (что вообще характерно для решения с использованием жадных алгоритмов). Метод разработан в рамках GreenCloud-архитектуры [21] группами исследователей из IBM China Research Laboratory, Университета Мак Гилла и Университета Нью-Мекскико в 2009 г.

4.3. Поиск лучшего расположения с помощью связующего дерева

Интересные результаты получены путем построения связующего дерева и применения к нему генетического алгоритма. В работе [9] метод применялся в приоритете к балансировке нагрузки. Реализуя его в *DPM* следует изменить критерий отбора генов. Нагрузки на ФМ можно оценивать различными методами, а далее, определяя дисперсию, получать вероятность отбора для "особи" — расположения ВМ$_i$ на ФМ$_j$ — l_{ij}. Преимуществами метода являются его гибкость и "обучаемость", работа без большого объема начальной статистики.

К недостаткам метода можно отнести сильные вариации критериев отбора, плохую сходимость в случае неправильной процедуры мутации и неоднозначность условия остановки.

5. Контроль целевого диапазона

В этом разделе описан метод, позволяющий оценить нужное количество включенных (активных) ФМ в зависимости от нагрузки. Важными параметрами являются верхняя граница H, при которой все N машин включаются в работу, и нижняя граница D, при которой все $(N - N')$ машин простаивают. В простейшем случае одна задача соответствует одной ВМ, иначе необходима более тщательная калибровка системы. Осуществляется мониторинг целевого множества.

Приведем оценки для системы на основе равноправных ФМ. Затраты энергии

$$C = C_1\alpha + C_2\bar{N},$$

где $\bar{N}$ — среднее число включенных машин; C_1, C_2 — затраты на удержание одного процесса (клиента, ВМ исполнителя задачи) и одного сервера в системе в секунду.

Определение C_1 и C_2 в рамки алгоритма не входит; оно идет экспериментально из калибровки системы [6].

Параметры (D, H, N') вычисляются из соотношений, полученных эвристически с учетом Марковости смены состояний и минимизации:

$$N' = \alpha_d\beta + 0,5\left(1 + \sqrt{1 + f\left(\frac{c_1}{c_2}\right)}\right);\ H = N;\ D = N' - 1.$$

Авторы этого метода, используя результаты исследования поведения ряда тестовых систем и некоторые эвристические соображения, предлагают использовать в качестве f функцию вида $f(x) = 4\alpha_d\beta x$.

Таким образом, фиксируются рамки целевого множества N', чем достигается баланс между производительностью и затратами электроэнергии.

Недостатками подхода являются необходимость калибровки системы для определения параметров модели C_1, C_2 и допущение того, что процессы смены состояний, прихода и обслуживания запросов являются Марковскими. Это накладывает ограничения как на применимость оценки, так и на границы H, D.

6. Оптимизация управления питанием на уровне ФМ (локальный *DPM*)

Существенное число алгоритмов рассматривают локальный *DPM* через призму набора состояний устройств, соответствующих стандарту ACPI [12]. В случае процессора реализуются два типа состояний в рамках стандарта DVFM [1]:

1) p_0, ..., p_n — состояния с адаптивным переключением частоты и напряжения;

2) набор C-состояний с жестко фиксированной частотой, в одном из которых реализована совокупность P-состояний (рис. 5).

Динамические потери энергии в большинстве случаев можно считать пропорциональными частоте

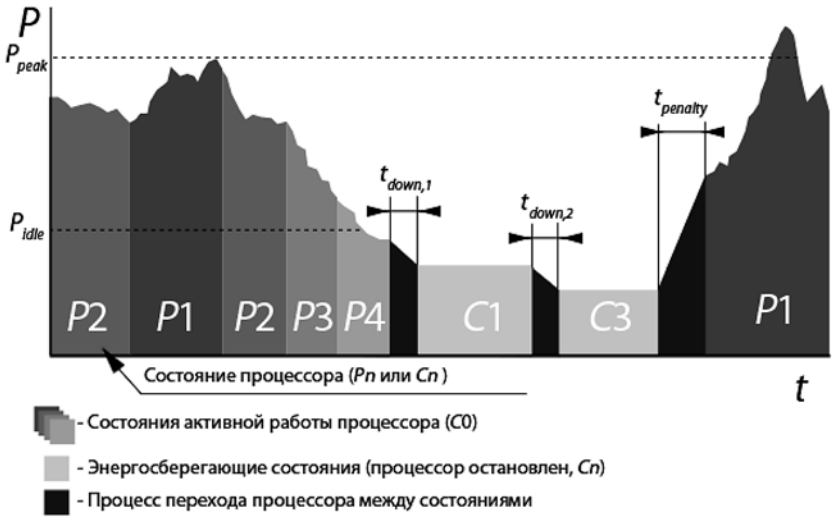

Рис. 5. *DPM* на уровне ФМ

и напряжению. Коэффициент пропорциональности является функцией свойств аппаратуры. Важно отметить, что частое включение/выключение ФМ влечет за собой паразитные затраты на миграцию и на изменение состояний.

6.1. Предсказание активности по показателям от предыдущего интервала

В качестве показателей выступают времена пребывания устройства в активных и энергосберегающих режимах и история переключений между ними. Результатом предсказания является рекомендация по выбору режима энергопотребления процессора [3]. Алгоритм прост в реализации, однако невосприимчив к пиковым нагрузкам.

6.2. Взвешенное предсказание активности по показателям от предыдущего интервала

Модификация предыдущего метода, в которой оценка проводится на основании взвешенных величин активностей в предыдущие интервалы и получении средних оценок [3]. Метод дает фильтрацию быстрых пиков нагрузки и редукцию количества различных предположений, но увеличивает объем обрабатываемых данных.

6.3. Марковские и полумарковские системы

В этом подразделе предложены политики управления питанием, основанные на предположении, что процессы переходов между состояниями системы — Марковские или полумарковские [3]. Простейшие подходы к решению подразумевают стационарность совокупности смены состояний системы (однородная цепь Маркова) либо периодический пересчет матрицы переходных вероятностей (построение неоднородной цепи в реальном времени). Для упрощения решения можно использовать адаптивные марковские алгоритмы, которые оперируют *lookup*-таблицей всевозможных стационарных стратегий и на каждом шаге работы выбирают наиболее приемлемую стратегию или же строят аппроксимации разложением по ним. Выделим два класса моделей, а именно — *DTMDP* и *CTMDP*.

Модель *DTMDP* — дискретная по времени модель, принимающая решения о смене состояний в соответствии с геометрическим распределением вероятности. Адаптивная модель *DTMDP* — способ применить данную модель к нестационарным политикам. Для работы в соответствии с этой моделью необходимо сначала собрать различные варианты эволюции системы, а затем применить адаптивный Марковский алгоритм совместно с коррекцией по внешним данным.

Модели *CTMDP* соответствует непрерывный Марковский процесс. По сравнению с предыдущим подходом использование этой модели позволяет сэкономить электроэнергию, не поднимая ФМ постоянно из состояния пониженного энергопотребления. Установлено, что метод на основе *CTMDP*-модели дает лучшие результаты при экспоненциальном распределении [22].

Преимущество подходов со стационарным распределением случайной величины по сравнению с динамической политикой заключается в отсутствии обработки по "внешнему" событию и сбора реальных данных в процессе работы. Кроме того, теоретически [3] методы можно реализовать без использования дополнительной памяти. На практике такой подход плохо применим: система способна эволюционировать и изменять поведение, а для коррекции *DPM* необходимы реальные параметры. Адаптивное расширение в общем случае дает лучшие показатели, но увеличивает число обращений и усложняет систему дополнительным анализом данных.

Другой фундаментальной проблемой является то, что в общем случае процесс может не быть Марковским. Данные алгоритмы применяют для *DPM* над устройствами, для которых процесс обработки запросов можно моделировать Марковской цепью с небольшим набором состояний (обычно достаточно двух состояний для входящих запросов: "активное" и "энергосберегающее"). Но алгоритмы плохо применимы для управления устройствами с большой дискретизацией нагрузок, в частности, процессором.

Использование *Event-Driven-политики* позволяет похожим образом моделировать поведение системы при изменении закона распределения. Ниже описаны две модели такого класса [8].

Semi-Markov Dynamic Policy. Обобщение модели *CTMDP*, в которой время разбивается на интервалы. Принятие решения о переходе индуцируется внеш-

ним событием. При этом интервалы между двумя переходами могут быть распределены произвольно, что и является преимуществом данной модели.

Time-Indexed SMDP: отличается от предыдущего подхода расширенным множеством состояний. Период времени T разбивается на равные интервалы. Состояние простоя и одно из состояний пониженного энергопотребления заменяются на наборы *time-indexed*-состояний, переходы в которые определяются условиями поведения системы в предыдущие кванты времени (конечный автомат из работы [3]). К этим условиям относятся, например, решение о смене состояния в результате простоя системы в предыдущий интервал, приходы запросов от пользователей ВМ или сервисные команды ФМ. Построенная политика должна учитывать достаточное количество подобных состояний, обобщая различные процессы. Желательно покрытие как можно большего числа различных нагрузок и последовательности их изменения. Выгодным отличием от предыдущего метода является унификация политики энергопотребления в одном конечном автомате [3].

6.4. Алгоритмы динамического обучения

Существует ряд алгоритмов, позволяющих динамически "обучать" систему верной стратегии *DPM*. В частности, к их числу относятся генетические алгоритмы [9], преимущества и недостатки которых описаны выше, а также метод на основе дерева принятия решений (*Learning Tree*) [16].

* * *

В работе в наиболее полном виде сформулирована задача *DPM* и даны качественные оценки различных решений подзадач *DPM*. По результатам их анализа авторы предлагают следующие рекомендации.

Подход к оценке нагрузки на виртуальные и физические машины должен выбираться в зависимости от конкретной конфигурации вычислительной системы. В частности, должно быть учтено характерное время системы T. Суммарную нагрузку от ВМ рекомендуется определять, где это возможно, в приближении $V_i(t) = \text{const}$, $t \in [t_{k-1}, t_k]$, простейшим методом (см. подразд. 3.1), варьируя T в соответствии с k, оценивая нагрузки на каждой ФМ (методом из подразд. 3.2), а затраты на переходы между состояниями ФМ — методом из подразд. 3.5, проведя необходимые калибровки.

Схема определения целевого диапазона ФМ в предположении, что получение и обработка запросов — Марковские процессы, представлена в методе, описанном в разд. 5. Эта модель также требует калибровки.

Выбор лучшего расположения наиболее оптимально осуществлять с помощью генетического алгоритма (см. подразд. 4.2) [9], однако на практике могут возникнуть сложности с поиском критерия отбора. В общем случае, если получить критерий не удается, следует использовать жадный алгоритм, описанный в подразд. 4.1, подобный тому, который применяется в GreenCloud [21].

На уровне ФМ рекомендуется использовать адаптивный алгоритм на основе полумарковского процесса. В случае таких устройств, как жесткий диск, сетевая карта, WLAN-карта, адаптивная полумарковская цепь применительно к нестационарным нагрузкам позволяет получить устойчивые результаты снижения энергопотребления и фильтрацию пиков. В случае устройств с более сложным набором состояний (например, процессоры), рекомендуется использовать обучающие алгоритмы.

Важно также находить баланс между эффективностью *DPM* и общей производительностью системы.

В ряде случаев имеет смысл решать задачу в рамках общей задачи оптимизации облака совместно с задачами балансировки нагрузки и планирования ресурсов. Причина в том, что их решения по отдельности могут конфликтовать или не достигать той степени эффективности, которая потенциально возможна при кооперации.

Список литературы

1. **Smeds J.** Evaluating power management capabilities of low-power cloud platforms. Master's thesis, Abo Akademi University. Finland, 2010.
2. **Veni T., Mary Saira Bhanu S.** Dynamic energy management in cloud data centers: a survey // International Journal on Cloud Computing: Services and Architecture (IJCCSA). 2013. Vol. 3, N. 4. P. 13—26.
3. **Šimunić T., Benini L., Glynn P., De Micheli G.** Event-Driven Power Management // IEEE transactions on computer aided design of integrated circuits and systems. 2001. N. 7 (20). P. 840—857.
4. **Qureshi A., Weber R., Balakrishnan H., Guttag J., Maggs B.** Cutting the electric bill for internet-scale systems // In Proceedings of SIGCOMM, 2009. NY, USA. 2009. P. 123—134.
5. **Economou D., Rivoire S., Kozyrakis C.** Full-system power analysis and modeling for server environments // Workshop on Modeling Benchmarking and Simulation (MOBS), 2006. Berkley, USA, 2006. P. 3—3.
6. **Kansal A., Zhao F., Liu J., Kothari N., Bhattacharya A. A.** Virtual machine power metering and provisioning // Proceedings of the 1st ACM symposium on Cloud computing. SoCC'10. 2010. P. 39—50.
7. **VMware** Distributed Power Management Concepts and Use, 2010. URL: http://www.vmware.com/resources/techresources/1080. (Дата обращения 30.09.2014).
8. **Mazzucco M., Mitrani I**. Empirical evaluation of power saving policies for data centers // SIGMETRICS Performance Evaluation Review. 2012. Vol. 40, N. 3. P. 18—23.
9. **Gu J., Hu J., Zhao T., Sun G.** A New Resource Scheduling Strategy Based on Genetic Algorithm in Cloud Computing Environment // Journal of Computers. 2012. Vol. 1, N. 7. P. 42—52.
10. **US Environmental** Protection Agency**.** Report to Congress on Server and Data Center Energy Efficiency: Public Law. P. 109—431. URL: http://hightech.lbl.gov/documents/data_centers/epa-datacenters.pdf (Дата обращения 30.09.2014).
11. **Koomey J.** Worldwide electricity used in data centers// Environmental Research Letters. 2008. vol. 3, no. 034008. September 23. URL: http://stacks.iop.org/1748-9326/3/034008. (Дата обращения 30.09.2014).
12. **Advanced** Configuration and Power Interface Specification. Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd, Toshiba Corporation Re-

vision. 2010. 4.0a. URL: http://www.acpi.info/DOWNLOADS/ACPIspec40a.pdf (Дата обращения 30.09.2014).

13. **Waldspurger C. A.** Memory resource management in VMware ESX Server // ACM SIGOPS Operating Systems Review — OSDI'02: Proceedings of the 5th symposium on Operating systems design and implementation. New York: ACM, 2002. Vol. 36, Is. SI. P. 181—194.

14. **Gulati A., Holler A., Ji. M., Shanmuganathan G., Waldspurger C., Zhu X**. VMware Distributed Resource Management: Design, Implementation and Lessons Learned // In VMware Technical Journal. March 2012. URL: https://labs.vmware.com/vmtj/vmware-distributed-resource-management-design-implementation-and-lessons-learned (Дата обращения 30.09.2014).

15. **Doob J. L.** Renewal Theory From the Point of View of the Theory of Probability // Transactions of the American Mathematical Society.1948, Vol. 63, N. 3. P. 423—438.

16. **Chung E., Benini L., De Micheli G.** Dynamic Power Management Using Adaptive Learning Tree // In Proc. IEEE/ACM International Conference on Computer Aided Design. 07—11 November, 1999. San Jose, CA, USA. NJ: IEEE Press Piscataway, 1999. P. 274—279.

17. **Wilf H. S.** Algorithms and Complexity. Philadelphia: University of Pennsylvania. 1994. PA 19104-6395. 135 p.

18. **Martello S., Toth P.** Knapsack problems: algorithms and computer implementations. New York: John Wiley & Sons, Inc., 1990, 308 p.

19. **Goudarzi H. Ghasemazar M., Pedram M.** SLA-based Optimization of Power and Migration Cost in Cloud Computing // 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. 13—16 May 2012. Ottawa. IEEE, 2012. P. 172—179. URL: http://sportlab.usc.edu/~massoud/Papers/SLA-cloudcomp-ccgrid12.pdf

20. **Amazon** Cloud EC2. URL: http://aws.amazon.com/ec2/#pricing (Дата обращения 30.09.2014).

21. **Liu L., Wang H., Liu X., Jin X., He W., Wang Q., Chen Y.** GreenCloud: A New Architecture for Green Data Center // ICAC-INDST '09 Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session. 2009. P. 29—38. URL: http://cse.unl.edu/~ylu/csce896/papers/p29-liu.pdf

22. **Qiu Q. and Pedram M.** Dynamic power management based on continuous — time Markov decision processes // Proceedings of the 36th annual ACM/IEEE Design Automation Conference. 1999. New York: ACM, 1999, P. 555—561. URL: http://sportlab.usc.edu/~massoud/Papers/dpm-contMarkov.pdf

N. N. Efanov[1], Junior Software Developer, e-mail: nefanov@parallels.com,
A. L. Melekhova[1, 2], Lead Software Developer, Lecturer, e-mail: annam@parallels.com,
A. O. Bondar[1], Junior Software Developer, e-mail: abondar@parallels.com
[1]Parallels, Moscow,
[2] Moscow Institute of Physics and Technology

Algorithms of Distributed Power Management for Cloud Computing

Nowadays, when the electricity price is that high, the demand on methods to reduce the power consumption for computing systems is high. While many computational tasks have migrated into the cloud, cloud power consumption becomes a case of study for many researchers. There are lots of investigations in this area, but most of them do not take into account a complete vision of the system. For example, many academic researches ignore the migration price, while most practical solutions are based on empirical results.

In this paper we give the aggregated picture of cloud system from the power management point of view. We cover both global and local power management and make a connection between them. We formulate power management subtasks that should be solved in the cloud system. We discuss the most promising results and do their critical evaluation. We cover dynamic programming solutions, learning tree algorithms, semi-markov representations and many other methods. The key focus is in evaluation of power consumption and power optimization on the local node. In the end we provide our recommendation on distributed power management politics implementation. This work is must-read for all who intend to build their own cloud or develop own cloud solution.

Keyword: *power consumption, virtualization, cloud computing, dynamic power management, energy proportional, energy efficiency*

References

1. **Smeds J.** Evaluating power management capabilities of low-power cloud platforms. Master's thesis, Abo Akademi University. Finland, 2010.

2. **Veni T., Mary Saira Bhanu S.** Dynamic energy management in cloud data centers: a survey. *International Journal on Cloud Computing: Services and Architecture* (IJCCSA), 2013, vol. 3, no. 4, pp. 13—26.

3. **Simunic T., Benini L., Glynn P., De Micheli G.** Event-Driven Power Management. *IEEE transactions on computer aided design of integrated circuits and systems*, 2001, no. 7 (20), pp. 840—857.

4. **Qureshi A., Weber R., Balakrishnan H., Guttag J., Maggs B.** Cutting the electric bill for internet-scale systems. *In Proceedings of SIGCOMM*, 2009, NY, USA, 2009, pp. 123—134.

5. **Economou D., Rivoire S., Kozyrakis C.** Full-system power analysis and modeling for server environments. *Workshop on Modeling Benchmarking and Simulation (MOBS)*, 2006, Berkley, USA, pp. 3—3.

6. **Kansal A., Zhao F., Liu J., Kothari N., Bhattacharya A. A.** Virtual machine power metering and provisioning. *Proceedings of the 1st ACM symposium on Cloud computing. SoCC'10.* 2010, pp. 39—50.

7. **VMware** Distributed Power Management Concepts and Use, 2010, available at: http://www.vmware.com/resources/techresources/1080

8. **Mazzucco M., Mitrani I**. Empirical evaluation of power saving policies for data centers. *SIGMETRICS Performance Evaluation Review*, 2012, vol. 40, no 3, pp. 18—23.

9. **Gu J., Hu J., Zhao T., Sun G.** A New Resource Scheduling Strategy Based on Genetic Algorithm in Cloud Computing Environment. *Journal of Computers,* 2012, vol. 7, no. 1, pp. 42—52.

10. **US Environmental** Protection Agency**.** Report to Congress on Server and Data Center Energy Efficiency: Public Law, pp. 109—431, available at: http://hightech.lbl.gov/documents/data_centers/epa-datacenters.pdf

11. **Koomey J.** Worldwide electricity used in data centers. *Environmental Research Letters*, 2008, vol. 3, no. 034008, September 23, available at: http://stacks.iop.org/1748-9326/3/034008.

12. **Advanced** Configuration and Power Interface Specification. Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd, Toshiba Corporation Revision. 2010. 4.0a, available at: http://www.acpi.info/DOWNLOADS/ACPIspec40a.pdf

13. **Waldspurger C. A.** Memory resource management in VMware ESX Server. *ACM SIGOPS Operating Systems Review — OSDI'02: Proceedings of the 5th symposium on Operating systems design and implementation,* 2002. New York: ACM, 2002, vol. 36, is. SI, pp. 181—194.

14. **Gulati A., Holler A., Ji M., Shanmuganathan G., Waldspurger C., Zhu X**. VMware Distributed Resource Management: Design, Implementation and Lessons Learned. *VMware Technical Journal*, March 2012, available at: https://labs.vmware.com/vmtj/vmware-distributed-resource-management-design-implementation-and-lessons-learned

15. **Doob J. L.** Renewal Theory From the Point of View of the Theory of Probability. *Transactions of the American Mathematical Society*, 1948, vol. 63, no. 3, pp. 423—438.

16. **Chung E., Benini L., De Micheli G.** Dynamic Power Management Using Adaptive Learning Tree. *In Proc. IEEE/ACM International Conference on Computer Aided Design*. San Jose, CA, USA. 07—11 November, 1999. NJ: IEEE Press Piscataway, 1999, pp. 274—279.

17. **Wilf H. S.** Algorithms and Complexity. Philadelphia: University of Pennsylvania. 1994, PA 19104-6395, 135 p.

18. **Martello S., Toth P.** Knapsack problems: algorithms and computer implementations. New York: John Wiley & Sons, 1990, 308 p.

19. **Goudarzi H., Ghasemazar M., Pedram M.** SLA-based Optimization of Power and Migration Cost in Cloud Computing. *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.* 13—16 May 2012. Ottawa. IEEE, 2012, pp. 172—179, available at: http://sportlab.usc.edu/~massoud/Papers/SLA-cloud-comp-ccgrid12.pdf

20. **Amazon** Cloud EC2, available at: http://aws.amazon.com/ec2/#pricing

21. **Liu L., Wang H., Liu X., Jin X., He W., Wang Q., Chen Y.** GreenCloud: A New Architecture for Green Data Center. *ICAC-INDST '09 Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session*, 2009, pp. 29—38, available at: http://cse.unl.edu/~ylu/csce896/papers/p29-liu.pdf

22. **Qiu Q., Pedram M.** Dynamic power management based on continuous-time Markov decision processes. *Proceedings of the 36th annual ACM/IEEE Design Automation Conference*. 1999. New York: ACM, 1999, pp. 555—561, available at: http://sportlab.usc.edu/~massoud/Papers/dpm-contMarkov.pdf

УДК 004.9

М. Р. Когаловский, канд. техн. наук, доц., руководитель центра интернет-ресурсов, e-mail: kogalovsky@ipr-ras.ru, Институт проблем рынка РАН, Москва,
С. И. Паринов, д-р техн. наук, зам. директора, e-mail: sparinov@gmail.com, Центральный экономико-математический институт РАН, Москва

Научные коммуникации в среде семантически обогащаемых электронных библиотек[1]

В последние годы ведутся разработки научных электронных библиотек, позволяющих пользователям децентрализовано декларировать в онлайновом режиме семантические связи между их информационными объектами. Благодаря этому семантически обогащается контент системы и в ее среде могут эффективно осуществляться научные коммуникации. В статье предложена технология для этих целей, реализованная в системе Соционет.

Ключевые слова: *электронная библиотека, семантическая связь, таксономия связей, система Соционет, научная коммуникация, сервис оповещения*

Введение

Коммуникации между учеными играют важную роль в деятельности глобального научного сообщества. Главная цель научных коммуникаций заключается в обеспечении функционирования глобальной системы передачи знаний и сложившегося разделения труда ученых. За многовековую историю науки сформировались как прямые, так и опосредованные формы научных коммуникаций. Прямые коммуникации представляют собой различные виды личных контактов ученых — выступления с докладами и обмен мнениями на научных мероприятиях, в процессе защиты диссертаций и т. д. Традиционными формами опосредованных научных коммуникаций являются: публикация результатов исследований в научных изданиях; восприятие идей в процессе чтения опубликованной работы и ее цитирование; рецензирование опубликованных или готовящихся к публикации работ и др.

Создание сети Интернет и доступных в глобальной среде информационных сервисов привело к существенной модернизации инфраструктуры и технологии научных коммуникаций. Появились возможности для более оперативного осуществления традиционных видов опосредованных научных коммуникаций с вовлечением значительно более широкого круга заинтересованных ученых. Одно из направлений развития форм научных коммуникаций основано на технологии семантического структурирования контента научных электронных библиотек [1—5]. Главными особенностями подхода, предлагаемого в указанных работах, являются: возможность декларации семантических связей между информационными объектами библиотеки децентрализовано *в онлайновом режиме* на основе поддерживаемой таксономии связей и тем самым *семантического обогащения* ее контента; представление семантических связей как полноценных информационных объектов (*first-class object*) специального типа; наличие в системе специального сервиса уведомления авторов информационных объектов — участников связей о событиях в системе, касающихся рассматриваемых связей — их создания, изменения их свойств, их удаления.

В обладающей такими возможностями научной электронной библиотеке можно отслеживать факты использования представленных в ней научных результатов для получения новых результатов и организовывать прямые коммуникации по некоторому набору сценариев между авторами и пользователями представляемых в системе научных результатов. Благодаря этому научные коммуникации, начинающиеся как опосредованные (ученый читает статью другого ученого), с помощью системы могут превратиться в прямые, независимо от того, знакомы ли лично эти ученые.

Набор возможных сценариев коммуникаций рассчитан на то, что в ответ на полученные уведомления авторы использованных работ могут реагировать различными способами, в зависимости от того, каким образом результаты их труда используют другие ученые. Возможны, в частности, следующие сценарии:

- автор использованного материала помогает ученому-потребителю его научных результатов повысить качество работы последнего, например, консультируя его, как полнее и правильнее использовать соответствующий результат;

[1] Работа поддержана РГНФ, проект 14-07-12010-в.

- автор использованного материала дорабатывает его и предоставляет потребителям более качественный научный продукт;
- автор использованного материала выражает несогласие (протест) с неправильным толкованием или применением его результата, обращая внимание на качество материала, в котором он используется, и т. д.

Важно отметить, что перечисленные сценарии прямых научных коммуникаций в значительной мере решают проблему контроля качества научных результатов, которая традиционно считается самым серьезным препятствием в развитии средств самоархивирования ("самиздата") результатов исследований. По мнению авторов, развитие и распространение на практике адресных научных коммуникаций, подобных описанным выше, чрезвычайно важны для научного сообщества и могут радикально изменить в лучшую сторону степень кооперации и уровень координации в научном сообществе, а также и эффективность научной системы в целом.

Авторам не известны другие проекты, в которых бы обеспечивалась поддержка коммуникаций пользователей на основе онлайнового семантического структурирования контента научной электронной библиотеки.

Данная статья посвящена описанию и анализу предлагаемого авторами подхода к развитию новых форм научных коммуникаций в онлайновой среде электронной библиотеки, а также представляет результаты его реализации.

Статья организована следующим образом. В разд. 1 дана общая характеристика предлагаемого подхода к созданию среды поддержки новых форм научных коммуникаций. В качестве базовой платформы для его реализации используется система Соционет [6]. В разд. 2 изложены принципы организации контента этой системы, представление и средства описания семантики связей между информационными объектами. В разд. 3 рассмотрены наиболее распространенные формы научных коммуникаций в среде электронной библиотеки, осуществление которых возможно с помощью технологии, реализующей предлагаемый подход. Разд. 4 посвящен описанию механизмов системы Соционет, которые используются для обеспечения новых форм виртуальных научных коммуникаций. В заключение отмечаются достоинства новых форм научных коммуникаций.

1. Общая характеристика подхода и среда его реализации

В ряде зарубежных работ (например, в работах [7—9]) предложены технологии, которые позволяют устанавливать связи между информационными объектами библиотеки с явным образом декларированной семантикой. Авторы настоящей работы называют такие связи *семантическими.* Аналогичная технология разработана авторами данной статьи [1—3, 5, 10—12]. Однако в отличие от других реализованных проектов в этой области, известных авторам, эта технология обеспечивает возможности научных коммуникаций в сообществе пользователей электронной библиотеки. Это достигается прежде всего за счет возможности декларации семантических связей авторизованными пользователями децентрализовано *в онлайновом режиме,* использования *механизма оповещения* авторов тех информационных объектов, которые стали участниками декларированных связей, а также представления семантических связей в системе *как обычных информационных объектов.* Тем самым они могут сами стать участниками других семантических связей.

Благодаря декларации семантических связей могут устанавливаться научные отношения различного рода между представленными в электронной библиотеке информационными объектами, например, между персонами и публикациями, между публикациями, между организациями и персонами и др. Семантика устанавливаемых отношений определяется на основе таксономии связей, поддерживаемой в электронной библиотеке.

Создание в онлайновом режиме семантической связи некоторого класса представляет вместе с тем *акт коммуникации* автора создаваемой связи как с сообществом пользователей библиотеки, так и персональной коммуникации с авторами информационных объектов — участников связей. Содержание передаваемого при этом сообщения их адресатам представляется классом/подклассом таксономии, а также, возможно, комментарием в описании связи. Коммуникация с автором целевого объекта связи имеет место, когда создается семантическая связь между профилем автора этой связи и некоторой публикацией. Например, это может быть связь какого-либо оценочного класса, выражающая одобрение целевой публикации связи, негативное к ней отношение и др. Другой случай — коммуникация между автором создаваемой связи и авторами публикаций-участников этой связи — возникает, например, при создании связи между двумя публикациями. Важно при этом, что авторы публикаций-участников создаваемой связи в обоих случаях уведомляются о создании связи с их публикациями. Они могут не только ознакомиться с параметрами связи, но и отреагировать на нее. Конечно, персональная коммуникация в виде уведомления авторов публикаций-участников созданной связи возможна лишь в том случае, если при регистрации в системе они указали свой адрес электронной почты. Акт коммуникации с авторами публикаций-участников связи порождается также и в случае, когда эта связь уже существует и изменяются значения некоторых ее атрибутов, прежде всего атрибута, указывающего класс связи.

Реализация рассматриваемого в данной статье подхода осуществлена в среде крупной научной информационной веб-системы Соционет [6], которая эксплуатируется в академическом научном сообществе около полутора десятилетий. Соционет поддерживает для авторизованных пользователей возмож-

ности своего рода социальной сети [4], позволяющей в онлайновом режиме децентрализовано декларировать семантические связи между представленными в системе информационными объектами (публикациями различных видов, наборами данных, персонами в различных ролях, организациями и др.). Сформирована таксономия семантических связей, определяющая семантику научных отношений между информационными объектами системы. На основе динамически развиваемой семантической структуры контента для системы Соционет созданы средства генерации наукометрических данных, более информативных по сравнению с традиционно используемыми на практике. Для пользователей обеспечиваются также контекстная визуализация установленных связей и визуальная навигация по семантической структуре контента. Для поддержки научных коммуникаций в среде системы предусматривается мониторинг изменений структуры связей — создание новых связей, удаление или изменение свойств существующих связей. Специальный сервис Соционет уведомляет авторов информационных объектов, если такие объекты становятся участниками новой связи или связи с изменившимися свойствами. Таким образом стимулируется реакция оповещаемых авторов на эти события — порождение новой коммуникации, в результате которой в системе декларируется новая связь. Ее участником может, в частности, являться связь — стимулятор данного действия. Могут также измениться свойства этой связи.

Конкретные формы возможных коммуникаций и средства их реализации в среде системы Соционет рассмотрены далее.

2. Контент системы и семантические связи информационных объектов

Система Соционет построена с использованием технологии открытых архивов (*Open Archives Initiative*, OAI) [13, 14]. Она обладает необходимыми механизмами интероперабельности для импортирования в свою среду других открытых архивов (включая созданных ее средствами), а также может предоставлять накопленные в ней метаданные для харвестинга (сбора метаданных) внешними системами, поддерживающими такие популярные протоколы, как OAI-PMH, REST API и FTP.

Соционет обеспечивает доступ к информационным объектам различных типов — статьям, монографиям, научным отчетам, справочникам, классификаторам и др. В соответствии с технологией OAI эти объекты могут храниться на различных узлах Веба, поддерживаться и использоваться независимо от системы Соционет, имеют собственных владельцев и уникально идентифицируются их URL. В Соционет указанные информационные объекты представлены их описателями — *метаобъектами*, состоящими из наборов метаданных.

Соционет имеет дело и с информационными объектами другого рода — с организациями и персонами (пользователями системы, в частности, авторами информационных объектов ранее указанных типов). Эти объекты также представляются в Соционет их метаобъектами, называемыми их *профилями*. Однако в отличие, например, от публикаций и монографий, сами они не имеют представления в Вебе. К числу таких виртуальных информационных объектов Соционет относятся также автономные семантические связи (см. далее), новости, научные артефакты и др. Каждый метаобъект имеет в системе *уникальный идентификатор*. Состав метаданных в метаобъекте зависит от типа описываемого им информационного объекта.

Однотипные информационные объекты Соционет группируются в *коллекции* по критерию, которым руководствуется администратор открытого архива. Открытый архив может включать произвольное число коллекций.

Основным информационным ресурсом открытого архива является его *репозиторий метаданных*. Он включает некоторые общие сведения об архиве в целом, а также коллекции метаобъектов, соответствующих информационных объектов системы с каталогами, которые воспринимаются пользователями как коллекции и каталоги этих информационных объектов. Доступ пользователей к информационным объектам осуществляется посредством их метаобъектов, содержащих ссылки на их представления в Вебе.

Между информационными объектами открытого архива могут устанавливаться *бинарные ориентированные связи*. Связываемые объекты могут принадлежать разным коллекциям из одного или разных архивов, поддерживаемых Соционет. Информационный объект, из которого исходит связь, далее называется *исходным объектом связи*, а объект, на который связь направлена, — ее *целевым объектом*. Связи между информационными объектами представляются в системе как связи между соответствующими им метаобъектами.

Различают связи с *предопределенной* и с *явно описанной* семантикой (*семантические связи*). Семантика связей первого вида может в дальнейшем при необходимости уточняться путем дополнения явного описания. Таким образом, они в результате становятся семантическими. Примерами связей с предопределенной семантикой являются следующие.

- Связь между организацией и персоной — ее сотрудником, обозначающая, что данная организация является местом работы сотрудника. При уточнении семантики этой связи может быть указана, например, должность персоны в организации.
- Связь между организацией и коллекцией информационных объектов, обозначающая, что организация является владельцем ресурсов этой коллекции.
- Связь между персоной и публикацией или другим информационным объектом, указывающая авторство данной персоны по отношению к этому информационному объекту. Семантика этой связи может быть уточнена, например, для коллективной

публикации, указанием вклада персоны в ее подготовку.

Такие связи в репозитории метаданных представляются с помощью метаданных — значений атрибутов соответствующих метаобъектов. Так, для задания связи между персоной и организацией в профиле персоны указывается *идентификатор профиля организации* (уникальный идентификатор, присваиваемый этому метаобъекту организации при его порождении в системе) в качестве значения атрибута *Организация*. Для задания обратной связи, имеющей в таком случае вид *одна-ко-многим*, нужно в профиле организации задать список *уникальных идентификаторов профилей персон*, являющихся ее сотрудниками. Связь авторства между публикацией и персоной в репозитории метаданных представляется указанием *идентификатора профиля персоны* в качестве значения атрибута *Автор* в метаобъекте данной публикации. Связь между персоной-автором и ее публикациями представляется списком *идентификаторов метаобъектов этих публикаций* в профиле персоны-автора.

Учитывая способ представления связей с предопределенной семантикой в репозиториях метаданных Соционет, назовем такие связи *встроенными*. Помимо указанных выше классов встроенных связей, в репозиториях метаданных поддерживаются также некоторые системные связи: между открытым архивом и его коллекциями, между коллекцией и составляющими ее метаобъектами. Нужно отметить, что встроенные связи могут создаваться только создателями метаобъектов исходных объектов связей, поскольку кроме них никто не имеет доступа к этим метаобъектам в целях их изменения. Заметим, что встроенные связи могут быть с предопределенной семантикой или семантическими.

Наряду со встроенными связями в Соционет поддерживаются *автономные* (внешние по отношению к связываемым метаобъектам) связи, которые могут быть только семантическими. Важная особенность автономных связей состоит в том, что они рассматриваются как обычные *самостоятельные информационные объекты* специального типа. Как и для объектов других типов, из автономных связей могут строиться коллекции информационных объектов-связей. Такие связи в отличие от встроенных сами могут быть участниками других связей.

Автономные связи, аналогично информационным объектам-организациям и персонам, представляемым только их профилями в репозитории метаданных, также представляются в виртуальной среде только их метаобъектами в системе. Метаобъект для объекта-связи включает следующие метаданные: уникальный идентификатор этого метаобъекта, уникальные идентификаторы исходного и целевого объектов связи, класс связи, уникальный идентификатор профиля ее автора, отметка времени момента ее создания и комментарий. Такие связи могут создавать в системе авторизованные пользователи в онлайновом режиме, порождая при этом их метаобъекты в репозитории метаданных.

В системе Соционет семантика создаваемых пользователями связей определяется при их создании на основе двухуровневой *таксономии связей,* которая реализована в виде набора *контролируемых словарей* имен классов (подклассов) связей. Каждому классу верхнего уровня таксономии соответствует отдельный контролируемый словарь, а его подклассам — элементы этого словаря. При разработке таксономии использованы созданные в последние годы модульные комплексы онтологий SPAR (*The Semantic Publishing and Referencing Ontologies*) [15, 16] и SWAN (*Semantic Web Applications in Neuromedicine*) [17], рекомендация SKOS (*Simple Knowledge Organization System*) [18], спецификации проекта CERIF для полной модели данных (*Full Data Model*) [19] и онтологии [20], определяющей систему терминов для обозначения сущностей этой модели и отношения между ними, а также классификатор *вкладов* авторов в подготовку коллективной публикации [21].

При создании связи учитываются типы связываемых объектов (персона, организация, монография, статья, связь и т. п.). Возможность создания связи определяется *матрицей допустимости,* в которой для каждой пары типов указаны допустимые классы связей.

Более подробные сведения о контенте Соционет, его семантическом структурировании и о таксономии связей системы можно найти в работах [1, 2, 5, 12, 22].

3. Онлайновые научные коммуникации в системе Соционет

Как уже отмечалось, авторизованные пользователи в системе Соционет могут в онлайновом режиме создавать семантические связи между информационными объектами. Создание конкретных связей фактически реализует акты коммуникации между авторами создаваемых связей и авторами связываемых информационных объектов. Содержание передаваемого при этом сообщения определяется классом/подклассом и другими атрибутами установленной связи, в частности, комментарием в ее описании, если он был указан. Рассмотрим наиболее часто встречающиеся разновидности научных коммуникаций и способы их реализации в системе Соционет.

Публикация результатов исследований и предоставление их научному сообществу без посредничества издателей. Развитие веб-технологий и технологий издательского дела открыло новые возможности для публикации результатов выполненных исследований в научных электронных библиотеках с помощью специалистов или самостоятельно *(самоархивирование*) без традиционного посредничества издателей. Такая форма коммуникации автора публикации с научным сообществом радикальным образом сокращает время поступления сообщения ее автора адресатам в виде его публикации и многократно расширяет круг потенциальных адресатов этого сообщения — специалистов, знакомящихся

с этой публикацией. Стимулирующее влияние на развитие такой формы коммуникации оказывают активно поддерживаемые в мировом научном сообществе инициативы открытого доступа к результатам научных исследований. В системе Соционет имеется механизм — *персональный робот*, который отслеживает появление в ее контенте публикаций по тематике, интересующей данного пользователя. В случае их появления пользователь получает уведомление от системы.

Сообщение пользователя системы автору некоторой публикации оценочного мнения о ней. Если пользователь системы создает связь одного из оценочных классов, которые предусмотрены в таксономии связей, между своим профилем и некоторой публикацией (ее метаобъектом), то он тем самым осуществляет коммуникацию с ее автором. Посредством этой коммуникации ее инициатор — создатель связи сообщает автору публикации свое мнение о его работе. Такая коммуникация аналогична по смыслу традиционному рецензированию. Содержание такой рецензии может выражаться не только в структурированном виде — именем класса связи (позитивная оценка, негативная оценка и т. п.), но и неструктурированными данными — факультативным текстовым комментарием в метаобъекте — описателе связи. При создании такой связи в системе активируется сервис уведомления, который направляет автору публикации сообщение по электронной почте о появлении новой связи, участником которой является его публикация, с указанием ссылки на метаобъект этой связи. Конечно, работа подобного механизма коммуникаций возможна только если их участники имеют в Соционет персональные профили с указанными в них адресами их электронной почты. В метаобъекте связи автор публикации может узнать о классе и авторе связи, времени ее создания, ознакомиться с комментарием.

Сообщение автору публикации о возможном изменении оценочного мнения некоторого пользователя о его работе. Авторизованный пользователь системы обладает полномочиями на изменение описания созданной им семантической связи. Изменение, в частности, может быть вызвано изменением мнения автора связи о ее целевой публикации. Для выполнения коммуникации в метаобъекте данной связи проводится замена ранее указанного класса связи или изменение атрибута-комментария. После сохранения модифицированного метаобъекта сервис уведомления, как и в предыдущем случае, отправляет автору ее целевой публикации сообщение по электронной почте.

Сообщение автору мнения пользователя системы о научном отношении между его публикацией и некоторой другой. Авторизованный пользователь может создать связь между двумя публикациями из контента системы, например, если по его мнению в исходной публикации высказано некоторое оценочное мнение о целевой публикации. Между такими публикациями может быть создана связь независимо от того, имеется ли в ее исходной публикации явное цитирование целевой публикации. В таком случае сообщение с предлагаемой оценкой направляется авторам обеих публикаций с указанием, как и ранее, уникального идентификатора метаобъекта созданной связи. В частном случае коммуникаций рассматриваемого вида создателем семантической связи между публикациями является автор исходной публикации и существует встроенная связь цитирования с предопределенной семантикой, констатирующей только факт цитирования, но не его мотивы. В этом случае автор исходной публикации может доопределить семантику этой связи с помощью таксономии связей и тем самым превратить существовавшую связь в семантическую. В таких случаях автор исходной публикации может создать семантическую связь не только оценочного, но и других классов таксономии. Например, он может указать, что в целевой публикации используется метод или данные, содержащиеся в его работе, либо что в целевой работе обнаружен плагиат из нее.

Сообщение автору публикации о возможном изменении мнения пользователя системы о научном соотношении этой и некоторой другой публикации. Такое сообщение порождается в случае, когда пользователь системы, например, осознал ошибочность своего мнения и вносит какие-либо изменения в метаобъект созданной им связи. После запоминания модифицированного метаобъекта, как и в предыдущем случае, авторам обеих связанных публикаций-участниц обновленной связи направляется уведомление по электронной почте.

Проведение научных дискуссий. Благодаря представлению автономных семантических связей в системе Соционет как обычных информационных объектов, они могут сами становиться участниками других связей. Используя эту возможность, можно поддерживать в системе дискуссионные форумы.

Пусть, например, пользователь системы выразил мнение о некоторой представленной в ней публикации, создав для этого связь какого-либо оценочного класса между своим профилем и метаобъектом указанной публикации. Уведомленный системным сервисом автор этой публикации или какой-либо иной пользователь системы может в свою очередь выразить мнение как о рассматриваемой публикации, так и о мнении, высказанном первым пользователем. В последнем случае он создает связь между своим профилем и метаобъектом связи, созданной первым пользователем, и т. д. Во всех случаях создания новых связей сервис уведомления оповещает авторов затрагиваемых публикаций и связей. Таким образом, в дискуссионный процесс могут вовлекаться и авторы оцениваемых публикаций, и создатели связей, разумеется, не обязательно оценочных классов.

Ограничимся обсуждением перечисленных разновидностей новых форм виртуальных научных коммуникаций. Возможны различные сценарии их использования.

4. Реализация новой технологии научных коммуникаций

К числу функциональных компонентов Соционет, используемых для реализации описанных форм коммуникаций, помимо средств самоархивирования результатов научных исследований относятся модуль создания персональных профилей пользователей системы, модуль создания контролируемых словарей таксономии семантических связей, модуль создания и обновления связей, сервис уведомления.

Модуль создания персональных профилей пользователей системы позволяет создавать профили пользователей в репозитории метаданных открытого архива, управляемого системой, в том числе и пользователей — авторов информационных объектов системы. Для поддержки научных коммуникаций необходимыми атрибутами профиля пользователя являются уникальный идентификатор профиля, присваиваемый ему системой, и адрес электронной почты пользователя. Этот адрес необходим для направления пользователю системных сообщений-уведомлений.

Модуль создания контролируемых словарей позволяет в интерактивном режиме создавать контролируемые словари, представляющие поддерживаемую в системе таксономию семантических связей. Возможно расширение таксономии путем дополнения подклассов в существующие контролируемые словари классов связей и/или создания новых контролируемых словарей. Состав словарей и их содержание могут безболезненно пополняться, если при этом не затрагивается состояние уже созданных коллекций связей. Включение в таксономию связей новых контролируемых словарей требует внесения дополнений в матрицу допустимости классов связей (см. разд. 2). Теоретически, в системе может одновременно поддерживаться несколько конкурирующих таксономий связей, каждая из которых представлена своим набором контролируемых словарей классов связей.

Модуль создания семантических связей позволяет создавать, пополнять и изменять коллекции автономных семантических связей, а также удалять отдельные связи из коллекций или полные коллекции связей. Все эти операции осуществляются пользователями в онлайновом режиме и модерируются системным администратором. К коллекциям автономных связей применимы все имеющиеся в системе функциональные возможности управления коллекциями любого типа данных. Заметим, что информационные объекты-участники связей могут быть *внутренними* для системы (содержащимися в ее контенте) или *внешними.* Внешние объекты не представлены в системе, для них нет соответствующих им метаобъектов. Они должны быть доступны в Вебе по их унифицированному идентификатору ресурса (URI). Допустимость внешних информационных объектов, а также публикаций, представленных в системе только их библиографическими описаниями, позволяет охватить в системе семантическими связями более широкое цифровое научное информационное пространство.

Для заданной пары информационных объектов может быть создано несколько связей. Один и тот же автор связей не может создать несколько связей одного класса с противоречивой семантикой для заданной пары объектов, но имеет возможность создать несколько связей разных классов. Для одной и той же пары объектов разными авторами связей может быть создано несколько связей одного класса, в том числе и с противоречивой семантикой.

Сервис уведомления авторов публикаций и связей создан в системе специально для поддержки коммуникаций между ними, как описано выше. Этот сервис активизируется в случае осуществления в системе какого-либо описанного акта коммуникации. В уведомлении, направляемом системой, указываются: уникальный идентификатор метаобъекта публикации или связи данного автора, ставшей участницей новой связи, или связи с измененными характеристиками; уникальный идентификатор метаобъекта этой связи, с помощью которого автор может получить к нему доступ и тем самым получить всю информацию о связи.

Система может уведомлять пользователей и в ряде иных случаев. Например, при появлении созданных разными авторами семантически противоречивых связей между некоторой парой информационных объектов, их авторы могут оповещаться об этой ситуации.

Сервис уведомления стимулирует ответную реакцию авторов публикаций-участников связей, а также пользователей системы — создателей связей, ставших участниками других связей. Тем самым этот системный механизм является *движителем коммуникационного процесса.*

Всякое обновление контента системы, а также структуры и свойств семантических связей, в Соционет модерируется системным администратором в соответствии с установленным регламентом. Автор публикации или связи может блокировать системные уведомления в случае, когда они становятся участницами новой семантической связи, либо изменились свойства уже существующей связи, в которой они участвуют. Для этого он должен отключить сервис уведомления в настройках своего персонального профиля. При необходимости этот режим может быть снова включен.

Заключение

Возможность свободного децентрализованного создания в системе связей информационных объектов, обладающих различной семантикой и отображающих разнообразные научные отношения между объектами научной сферы деятельности, позволяет использовать электронную библиотеку, которая предоставляет пользователям такие возможности как полигон для новых форм научных коммуникаций. Рассылка уведомлений авторам информационных объектов и создателям связей стимулирует новые коммуникации указанных персон, способствующие повышению качества научных результатов участников

коммуникаций, а также существенно ускоряет процессы создания и тестирования нового научного знания. Новые формы научных коммуникаций в среде электронных библиотек с рассмотренными функциональными возможностями характеризуются открытостью сообщений, передаваемых при их осуществлении. Это обеспечивает более высокий уровень ответственности участников коммуникаций перед научным сообществом. Для усиления "общественного контроля" за такого рода активностью результаты всех действий пользователя по созданию семантических связей, а также реакции на них фиксируются в его статистическом портрете [23]. Эта статистика является публично доступной и в определенной степени формирует научную репутацию соответствующего пользователя.

Список литературы

1. **Когаловский М. Р., Паринов С. И.** Семантическое структурирование контента научных электронных библиотек на основе онтологий // Современные технологии интеграции информационных ресурсов: сборник науч. трудов. СПб.: Президентская библиотека им. Б.Н. Ельцина, 2011. С. 26—45.
2. **Когаловский М. Р., Паринов С. И.** Классификация и использование семантических связей между информационными объектами в научных электронных библиотеках // Информатика и ее применения. 2012. Т. 6, вып. 3. С. 32—42.
3. **Когаловский М. Р., Паринов С. И.** Новый источник данных для наукометрических исследований // Труды XV Всероссийской науч. конф. “Электронные библиотеки: перспективные методы и технологии, электронные коллекции — RCDL—2013”. Ярославль, 14—17 октября 2013 г. Ярославль: Изд-во Ярослав. гос. ун-та. 2013. С. 107—117.
4. **Когаловский М. Р., Паринов С. И.** Технологии социальной сети для создания семантических связей информационных объектов в научной электронной библиотеке // Программирование. 2014. Т. 40, № 6. С. 22—33.
5. **Паринов С. И., Когаловский М. Р.** Технология семантического структурирования контента научных электронных библиотек // Труды XIII Всероссийской науч. конф. "Электронные библиотеки: перспективные методы и технологии, электронные коллекции — RCDL-2011". Воронеж, 19—22 октября 2011 г. Воронеж: Изд-во Ворон. гос. ун-та, 2011. С. 94—103.
6. **Паринов С. И., Ляпунов В. М., Пузырев Р. Л.** Система Соционет как платформа для разработки научных информационных ресурсов и онлайновых сервисов // Российский научный электронный журнал "Электронные библиотеки". 2003. Т. 6, вып. 1. URL: http://www.elbib.ru/index.phtml?page=elbib/rus/journal /2003/part1/PLP
7. **Dix A., Levialdi S., Malizia A.** Semantic halo for collaboration tagging systems. In the Social Navigation and Community-Based Adaptation Technologies Workshop-June 20th, 2006.
8. **Shotton D.** Open citations //Nature. 2013. Vol. 502. URL: http://www.nature.com/polopoly_fs/1.13937!/menu/main/topColumns/topLeftColum n/pdf/502295a.pdf
9. **Shotton D.** Use of CiTO in CiteULike. 2010. URL: http://opencitations.wordpress.com/2010/10/21/use-of-cito-in-citeulike/
10. **Parinov S.** Open Repository of Semantic Linkages // In: Proceedings of 11th International Conference on Current Research Information Systems e-Infrastructure for Research and Innovations (CRIS 2012), Prague, 2012. URL: http://socionet.ru/publication.xml?h=repec:rus:mqijxk:29
11. **Parinov S.** Towards a Semantic Segment of a Research e-Infrastructure: necessary information objects, tools and services Metadata and Semantics Research, Communications in Computer and Information Science / Eds. by J. M. Dodero, M. Palomo-Duarte, P. Karampiperis. Springer, 2012. Vol. 343, P. 133—145. URL: http://socionet.ru/pub.xml?h=RePEc:rus:mqijxk:30
12. **Parinov S., Kogalovsky M.** Semantic Linkages in Research Information Systems as a New Data Source for Scientometric Studies // Scientometrics. 2014. Vol. 98, Is. 2, P. 927—943.
13. **Open Archives** Initiative. URL: http://www.openarchives.org/
14. **The Open Archives** Initiative Protocol for Metadata Harvesting. Protocol Version 2.0 of 2002-06-14. Document Version 2008-12-07T20:42:00Z. URL: http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm
15. **Shotton D.** Introduction the Semantic Publishing and Referencing (SPAR) Ontologies. October 14, 2010. URL: http://opencitations.wordpress.com/2010/10/14/ introducing-the-semantic-publishing-and-referencing-spar-ontologies/
16. **Shotton D., Peroni S.** Semantic annotation of publication entities using the SPAR (Semantic Publishing and Referencing) Ontologies // Beyond the PDF Workshop, La Jolla, 19 January 2011. URL: http://imageweb.zoo.ox.ac.uk/pub/2010/Publications/Shotton&Peroni_semantic_annotation_of_publication entities.pdf
17. **Semantic** Web Applications in Neuromedicine (SWAN) Ontology. W3C Interest Group Note, 20 October 2009. URL: http://www.w3.org/TR/2009/NOTE-hcls-swan-20091020/
18. **SKOS** — Simple Knowledge Organization System Reference. W3C Recommendation, 18 August 2009. URL: http://www.w3.org/TR/skos-reference/
19. **CERIF** 1.3 Full Data Model (FDM): Introduction and Specification, euroCRIS, 2012. URL: http://www.eurocris.org/Uploads/Web %20pages/CERIF-1.3/Specifications/CERIF1.3_FDM.pdf
20. **CERIF** 1.3 Semantics: Research Vocabulary. CERIF Task Group, euroCRIS, 2012. URL: http://www.eurocris.org/Uploads/Web %20pages/CERIF-1.3/ Specifications/CERIF1.3_Semantics.pdf
21. **Allen L., Brand A., Scott J., Altman M. and Hlava M.** Credit where credit is due. URL: http://www.nature.com/polopoly_fs/1.15033!/menu/main/ topColumns/topLeftColumn/pdf/508312a.pdf
22. **Когаловский М. Р., Паринов С. И.** Метрики онлайновых информационных пространств // Экономика и математические методы. 2008. Т. 44, вып. 2. С. 108—120.
23. **Parinov S., Kogalovsky M., Lyapunov V.** A Challenge of Research Outputs in GL Circuit: From Open Access to Open Use // The Gray Journal. An International Journal on Grey Literature, 2014. Vol. 10, N. 2. P. 87—94.

M. R. Kogalovsky, Head of Internet Resources Center, e-mail: kogalovsky@ipr-ras.ru, Market Economy Institute of RAS, Moscow, **S. I. Parinov**, Deputy Director, e-mail: sparinov@gmail.com, Central Economics and Mathematics Institute of RAS, Moscow

Scientific Communications Based on Facilities of the Semantically Enrichable Digital Libraries

A new trend in developing of the scientific digital library concepts and technology emerged in the last years. Authorized users of a library get ability for declaration in a decentralized mode the scientific relationships between information objects from the library content by using the semantic linkage technique and taxonomy. We call the digital libraries owned by such facilities as the semantically enrichable digital libraries. In this paper we present our ap-

proach to execution of direct scientific communications between the authors of research outputs and the users of these outputs belonged to the library content. This functionality is based on the semantic linkage technique and on the e-mail notification mechanism. The author of a publication gets an immediate e-mail message when someone creates semantic linkage with the publication. This message gives to the author an information who, how and why used his/her research outputs, since the linkage semantic is defined by the scientific linkage taxonomy including different variety of usage (for example, adoption of idea, method or data), opinions, etc. This the first communication motivates further direct communication and cooperation since as a feedback to the user: the author can assist the user with better using of the author's research results; and/or the author can improve/ develop their research results in order to help the user with getting a better usage effect. Proposed approach has been implemented as a part of the Socionet facilities. This project is funded by Russian Foundation for Humanities, grant number 14-07-12010-v.

Keywords: *digital library, semantic linkage, linkage taxonomy, scientific communication, notification service, Socionet system*

References

1. **Kogalovsky M. R., Parinov S. I.** Semanticheskoe strukturirovanie kontenta nauchnyh elektronnyh bibliotek na osnove ontologiy. *Sovremennye tehnologii integracii informacionnyh resursov: sbornik nauchnyh trudov.* Sankt-Petersburg: Prezidentskaya biblioteka im. B. N. Elcina, 2011, pp. 26—45. (in Russian).

2. **Kogalovsky M. R., Parinov S. I.** Klassifikaciya i ispolzovanie semanticheskih svyazey mezhdu informacionnymi obektami v nauchnyh elektronnyh bibliotekah. *Informatika i ee primeneniya*, 2012, vol. 6, no. 3, pp. 32—42. (in Russian).

3. **Kogalovsky M. R., Parinov S. I.** Novyy istochnik dannyh dlya naukometricheskih issledovaniy. *Trudy XV Vserossiyskoy nauchnoy konferencii "Elektronnye biblioteki: perspektivnye metody i tehnologii, elektronnye kollekcii — RCDL-2013".* Yaroslavl, 14—17 October 2013. Yaroslavl: Izd-vo Yaroslav. gos. un-ta, 2013, pp. 107—117 (in Russian).

4. **Kogalovsky M. R., Parinov S. I.** Tehnologii socialnoy seti dlya sozdaniya semanticheskih svyazey informacionnyh obektov v nauchnoy elektronnoy biblioteke. *Programmirovanie*, 2014, vol. 40, no. 6, pp. 22—33 (in Russian).

5. **Parinov S. I., Kogalovsky M. R.** Tehnologiya semanticheskogo strukturirovaniya kontenta nauchnyh elektronnyh bibliotek. *Trudy XIII Vserossiyskoy nauchnoy konferencii "Elektronnye biblioteki: perspektivnye metody i tehnologii, elektronnye kollekcii — RCDL-2011*". Voronezh, 19—22 October 2011. Voronezh: Izd-vo Voronezh. gos. un-ta, 2011, pp. 94—103 (in Russian).

6. **Parinov S. I., Lyapunov V. M., Puzyrev R. L.** Sistema Socionet kak platforma dlya razrabotki nauchnyh informacionnyh resursov i onlaynovyh servisov. *Rossiyskiy nauchnyy elektronnyy zhurnal "Elektronnye biblioteki",* 2003, vol. 6, no. 1, available at: http://www.elbib.ru/index.phtml?page=elbib/rus/journal/2003/part1/PLP (in Russian).

7. **Dix A., Levialdi S., Malizia A.** Semantic halo for collaboration tagging systems. *In the Social Navigation and Community-Based Adaptation Technologies Workshop-June 20th*, 2006.

8. **Shotton D.** Open citations. *Nature,* 2013, vol. 502, available at: http://www.nature.com/polopoly_fs/1.13937!/menu/main/topColumns/topLeft Column/pdf/502295a.pdf

9. **Shotton D.** Use of CiTO in CiteULike. 2010, available at: http://opencitations.wordpress.com/2010/10/21/use-of-cito-in-citeulike/

10. **Parinov S.** Open Repository of Semantic Linkages. *Proceedings of 11th International Conference on Current Research Information Systems e-Infrastructure for Research and Innovations (CRIS 2012).* Prague, 2012, available at: http://socionet.ru/publication.xml?h=repec:rus:mqijxk:29

11. **Parinov S.** Towards a Semantic Segment of a Research e-Infrastructure: necessary information objects, tools and services. *Metadata and Semantics Research, Communications in Computer and Information Science* / Eds. by J. M. Dodero, M. Palomo-Duarte, P. Karampiperis, Springer, 2012, vol. 343, pp. 133—145, available at: http://socionet.ru/pub.xml?h=RePEc:rus:mqijxk:30

12. **Parinov S., Kogalovsky M.** Semantic Linkages in Research Information Systems as a New Data Source for Scientometric Studies. *Scientometrics*, 2014, vol. 98, is. 2, pp. 927—943.

13. **Open Archives** Initiative, available at: http://www.openarchives.org/

14. **The Open Archives** Initiative Protocol for Metadata Harvesting. Protocol Version 2.0 of 2002-06-14. Document Version 2008-12-07T20:42:00Z, available at: http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm

15. **Shotton D.** Introduction the Semantic Publishing and Referencing (SPAR) Ontologies. October 14, 2010, available at: http://opencitations.wordpress.com/2010/10/14/introducing-the-semantic-publishing-and-referencing-spar-ontologies/

16. **Shotton D., Peroni S.** Semantic annotation of publication entities using the SPAR (Semantic Publishing and Referencing) Ontologies. *Beyond the PDF Workshop*, La Jolla, 19 January 2011, available at: http://imageweb.zoo.ox.ac.uk/pub/2010/Publications/Shotton&Peroni_semantic_annotation_of_publication entities.pdf

17. **Semantic** Web Applications in Neuromedicine (SWAN) Ontology. W3C Interest Group Note, 20 October 2009, available at: http://www.w3.org/TR/2009/NOTE-hcls-swan-20091020/

18. **SKOS —** Simple Knowledge Organization System Reference. W3C Recommendation, 18 August 2009, available at: http://www.w3.org/TR/skos-reference/

19. **CERIF** 1.3 Full Data Model (FDM): Introduction and Specification. euroCRIS, 2012, available at: http://www.eurocris.org/Uploads/Web %20pages/CERIF-1.3/Specifications/CERIF1.3_FDM.pdf

20. **CERIF** 1.3 Semantics: Research Vocabulary. CERIF Task Group, euroCRIS, 2012, available at: http://www.eurocris.org/Uploads/Web %20pages/CERIF-1.3/Specifications/CERIF1.3_Semantics.pdf

21. **Allen L., Brand A., Scott J., Altman M., Hlava M.** Credit where credit is due, available at: http://www.nature.com/polopoly_fs/1.15033!/me nu/main/topColumns/topLeftColumn/pdf/508312a.pdf

22. **Kogalovsky M.R., Parinov S.I.** Metriki onlaynovyh informacionnyh prostranstv. *Ekonomika i matematicheskie metody*, 2008, vol. 44, no. 2, pp. 108—120. (in Russian).

23. **Parinov S., Kogalovsky M., Lyapunov V.** A Challenge of Research Outputs in GL Circuit: From Open Access to Open Use. *The Gray Journal. An International Journal on Grey Literature*, 2014, vol. 10, no 2, pp. 87—94.

УДК: 004.4'23

Д. И. Харитонов, канд. техн. наук, ст. науч. сотр., **Г. В. Тарасов,** науч. сотр.,
Р. В. Парахин, аспирант, e-mail: fadak@dvo.ru,
Институт автоматики и процессов управления ДВО РАН, Владивосток

Об одном подходе к использованию базы данных для хранения исходных текстов программ[1]

Представлен подход к хранению исходных кодов программ в реляционных базах данных. Описаны схема хранения программы в базе данных; операции импорта и экспорта исходного кода программы в/из реляционного представления в традиционное файловое представление; объектный интерфейс доступа к объектам базы данных из прикладных программ. В заключение представлен ряд направлений исследований, в которых хранение программ в базах данных могло бы дать ряд преимуществ по сравнению с традиционным файловым представлением.

Ключевые слова: *реляционные системы управления базами данных, трансляция программ, объектно-ориентированное программирование*

Введение

С момента появления первых больших компьютеров и первых программ для этих компьютеров заговорили о "кризисе программного обеспечения" [1]. По мнению авторов этого термина (Ф. Л. Бауэр, термин также использовал Э. В. Дейкстра) основная причина кризиса — резкий рост производительности вычислительных машин. Позднее в списке причин появилась технологическая сложность управления программным проектом. Она обусловлена сложностью исходной постановки задачи, сложностью создания и управления большим объемом программного кода, существенными затратами на отладку и тестирование выпускаемого программного продукта и рядом других факторов, препятствующих своевременному выходу качественного программного продукта. В отличие от успехов аппаратного обеспечения (с того времени производительность компьютеров увеличилась в миллионы раз), в программной инженерии не наблюдается существенных изменений в области создания верифицированного программного обеспечения. С каждым годом в новостных лентах и обзорах все больше появляется информация о различных обнаруженных в программных продуктах ошибках. Нередко оказывается, что эти ошибки присутствовали в коде программы годами.

Один из путей преодоления кризиса видится в создании "верифицирующего компилятора", который бы автоматически, наряду с трансляцией и оптимизацией, проверял бы корректность созданной программы. Впервые эта идея была предложена Р. В. Флойдом в 1967 г. В 2003 г. Ч. Э. Хоар отнес ее к разряду крупнейших нерешенных научно-технических задач современности (*Grand Challenge*) [2]. В 2008 г. группой авторов [3] был сформулирован ряд требований, которым должен удовлетворять язык программирования, программы на котором могут быть автоматически проверены верифицирующим компилятором. Одним из таких требований является необходимость разбиения кода на отдельные элементы различной степени детализации, верификация которых может быть проведена независимо от всех остальных элементов программы. В условиях изменения программы в ходе ее жизненного цикла первостепенной задачей является поддержание целостности взаимосвязей всех компонентов программы. Обеспечение целостности программного кода позволит повторно использовать результаты верификации неизменившихся компонентов программы и по частям проводить проверку новых или измененных компонентов программы.

Интересным направлением исследований в области повторного использования кода и обеспечения целостности программы является вовлечение в процесс управления программным кодом системы управления базами данных (СУБД), архитектура которой изначально предполагает наличие механизмов обеспечения целостности данных. Для решения этой задачи

[1] Работа выполнена при финансовой поддержке программы фундаментальных исследований Президиума РАН № 43 по стратегическим направлениям развития науки на 2014 г. "Фундаментальные проблемы математического моделирования".

необходимо разработать промежуточное представление программы. Такое представление должно быть:

- пригодно для хранения программы в базе данных;
- адаптировано для использования и редактирования существующими средствами разработки программ;
- эффективно в процессе анализа и верификации программы с помощью как существующих, так и новых инструментальных средств.

В данной статье предложено рассмотреть основы представления программы в реляционной базе данных в целях последующего расширения функций СУБД по управлению ее кодом.

Изложение материала в статье построено следующим образом. В разд. 1 дано краткое описание средств промежуточного представления программ, которые используются в различных инструментальных средствах трансформации программ и в оптимизирующих компиляторах. В разд. 2 описана информационная схема базы данных для хранения программных проектов (проектов управления кодом программы). В разд. 3 описан объектный интерфейс доступа к элементам базы данных. Разд. 4 и 5 посвящены процедурам импорта и экспорта кода программы из/в традиционное файловое представление. В заключении сделаны выводы о возможности применения предлагаемого подхода, а также определены направления его дальнейшего развития.

1. Существующие способы промежуточного представления программ

Промежуточное представление программ широко используется в различных областях компьютерных технологий [4, 5]. Трансляция, оптимизация, анализ, верификация и отладка программ — вот лишь немногие разделы компьютерных технологий, которые в своей основе опираются на некоторое внутреннее представление программы. Промежуточное представление помогает представить исходную программу таким образом, чтобы эффективно обеспечить решение конкретной задачи: провести трансляцию в исполняемый код, выполнить анализ производительности и т. д.

Существуют различные формы внутреннего представления программ. Наиболее широко используют описанные далее.

♦ Представление в виде ориентированных графов. Классическими примерами использования графов в промежуточных представлениях являются “дерево абстрактного синтаксиса”, “граф потока управления”, “граф информационных зависимостей” и др.

♦ Представление в виде троек или четверок операторов и операндов. Данное представление часто применяется на стадии генерации машинного кода в компиляторах.

♦ Различные формы линеаризованного представления графов — префиксная или постфиксная записи. Данные формы записи широко используют в различных стековых языках, они являются основой для организации порядка вычислений математических выражений.

Наиболее широкое применение внутреннее представление программы нашло в теории преобразования программ, которая охватывает все вопросы, связанные с их трансляцией и оптимизацией. Например, в хорошо известном свободном наборе компиляторов GCC используются перечисленные далее три вида внутренних представлений.

- GENERIC. Высокоуровневое представление исходного кода программы в виде дерева. Главная роль данного представления состоит в адекватном отображении конструкций и концепций исходного языка программирования в виде синтаксически связанных элементов дерева.
- GIMPLE. Трехадресное промежуточное представление исполняемых выражений из GENERIC-дерева. Данное представление адекватно отражает особенности исходного языка программирования и допускает эффективную оптимизацию и генерацию исполняемого кода.
- RTL. Заключительная форма представления исполняемого кода перед переводом в машинные инструкции. Каждое выражение в алгебраической форме описывает, какое действие выполняется согласно инструкции.

Другой хорошо известной системой анализа, трансформации и оптимизации программ является система LLVM. В ее основе лежит промежуточное представление кода в виде набора инструкций, более высокоуровневых чем обычный ассемблер, в форме SSA (*Static Single Assignment*).

Следует также упомянуть отечественные разработки САПФОР [6] и OPC (Открытая распараллеливающая система) [7]. В основе системы САПФОР лежит база данных, которая обеспечивает обмен информацией между основными компонентами системы, включая анализатор, генератор, диалоговую оболочку. В OPC внутреннее представление программы хранится в древовидной структуре с различными типами вершин и уровнями представления конструкций программы. Неотъемлемой частью этого дерева является библиотека сервисных функций, отвечающая за его целостность и за преобразования, выполняемые над деревом.

Другие системы, например Polaris, Sage++, SUIF 2 Compiler System, Rose, Cetus, используют объектно-ориентированное внутреннее представление исследуемых программ, где каждый объект описывает некоторое понятие исходного языка программирования.

С помощью сопутствующих библиотек доступа к дереву объектов осуществляются импортирование программы во внутреннее представление, оптимизирующие преобразования и экспортирование результирующего представления.

Еще одним направлением использования внутреннего представления программы являются так называемые языки запроса программы (*Program Query Language*). Основная суть исследований на этом направлении состоит в разработке средств поддержки создания, изменения и анализа программного кода. Среди прочих пакетов можно выделить ASTLOG, JQuery, PQL, PTQL, Stratego и xtc. К этому же направлению следует отнести интегрированные среды разработки (например, Microsoft Visual Studio, Code::Blocks), которые в процессе работы над проектом создают базу данных исходного кода программы и обеспечивают процесс его редактирования.

2. Схема базы данных

Опишем схему предлагаемой авторами базы данных для хранения кода программы. При составлении схемы будем опираться на основные функциональные возможности интегрированных сред разработки (IDE). Во всех существующих IDE управление кодом программы осуществляется в терминах проекта. *Проект* — это структурная единица программного продукта, реализующая отдельную самостоятельную его часть. Программный продукт может состоять из одного и более различных проектов. Проекты обладают *конфигурациями*, которые описывают правила настройки кода программы. Чаще всего для описания правил используется язык препроцессора. Исходный код хранится в *текстовых файлах*, доступ к которым возможен в *многопользовательском режиме* (например, через системы управления версиями CVS, Subversion, git). *Промежуточное представление программы* генерируется на стадии трансляции и оптимизации и не хранится постоянно в течение всего времени работы над проектом. Таким образом, можно выделить пять основных разделов информационной схемы базы данных. Минимальный вариант схемы показан на рис. 1, см. третью сторону обложки. Коротко опишем основные сущности и связи.

Файлы (FileRepresentation). Хранение исходных текстов программы в базе данных в виде файлов организовано с использованием трех таблиц. Таблицы STORAGE и SCONTENT образуют иерархическую файловую систему. Таблица CONTENT_TYPE описывает ссылку на грамматику языка содержимого файла и определяет его тип. В настоящее время база данных ориентирована на работу с языками C/C++ и Fortran.

Проекты (Projects). Описание проектов хранится в трех таблицах: PROJECT_LIST; PROJECT_TYPE; PROJECT_DEPENDENCIES. Основная информация о проекте содержится в таблице PROJECT_LIST. Таблица PROJECT_DEPENDENCIES описывает зависимости между проектами. В таблице PROJECT_TYPE хранится информация о типах проектов с указанием по умолчанию типов файлов, принадлежащих данному проекту. Раздел "Проекты" связан с разделом "Файлы" двумя основными связями. Каждый проект должен начинаться с некоторой корневой директории, где хранятся файлы проекта. Каждый тип проекта определяется типом файла по умолчанию и, таким образом, связывает грамматику языка разбора содержимого файла с его проектом.

Промежуточное представление (SourceRepresentation). Зная список грамматик, которые принадлежат проекту программы, из исходных файлов можно получить промежуточное представление (ПП) программы. Для описания ПП разработаны четыре основные таблицы. Таблица CODE_UNITS описывает программный элемент, обладающий собственным уникальным именем. В терминах языка C/C++ примером именованного программного элемента являются классы, функции и методы классов, пользовательские типы данных, глобальные переменные и т. п. Каждый элемент программы имеет содержимое, которое описывается таблицей CODE_CONTENT. Будем различать активное содержимое (поле ACTIVE_CONTENT) и текущее содержимое (поле MODIFIED_CONTENT). Активное содержимое — это актуальная версия текста кода данного программного элемента. Дополнительные поля DTIME и VERSION таблицы CODE_CONTENT образуют способ хранения истории изменения содержимого программного элемента аналогично тому, как это делается в системах управления версиями. Текущее содержимое — это версия содержимого программного элемента, которая в данный момент времени редактируется пользователем. Процесс редактирования завершается переносом текущего содержимого в новое активное содержимое. В таблицу CODE_CONTENT добавлен флаг состояния state, обозначающий текущее состояние элемента. Таблицы CODE_DEP и DEP_TYPE формируют дерево информационных связей между программными элементами. С помощью этих структур можно будет хранить элементы информационного графа данных, графа вызовов функций и методов классов и другие типы семантических связей. Программные элементы могут иметь сложную внутреннюю структуру. Факт вложенности одного программного элемента в другой отражается в циклической связи по полю HOST таблицы CODE_UNITS.

Таким образом, в разработанной схеме ПП программы — это лес деревьев, каждое из которых описывает отдельный элемент программы. Все элементы программы сохраняются в базе данных на протяжении всего жизненного цикла программы. Элементы ПП программы формируются на стадии

разбора исходного кода программы во внутреннее представление по заданной грамматике. Алгоритм преобразования из файлового представления во внутреннее представление с использованием грамматик описан далее в разд. 4.

Конфигурация (SourceConfiguration). Исходный код практически каждой программы снабжен элементами препроцессора, который, как правило, используется для реализации механизмов переноса кода программы, внесения условных блоков и параметров при сборке исполняемого представления. В настоящее время конфигурирование кода ограничено двумя директивами препроцессора #include и #define, синтаксическое определение которых хранится в таблицах DEFINE и SINCLUDE. Все элементы конфигурации объединятся под одним именем, заданным в таблице CODE_CONFIG, которая в свою очередь связана с проектами. Таким образом, в базе данных любой проект может иметь произвольное число конфигураций, каждая из которых состоит из текстового описания подставляемых имен (для #define) и включаемых имен (для #include). Связь с конкретными элементами программы осуществляется по именам, заданным полями FQ_NAME и NAME.

Действия пользователя (Users). Все действия пользователя, связанные с редактированием элементов программы, фиксируются в отдельных таблицах схемы. Таблица USER_LIST хранит список пользователей, имеющих доступ к базе данных. Таблицы ACTIONS_LOG и code_actions сохраняют во времени типы действий, которые проводились с заданным элементом программы. На данный момент различают следующие типы действий: добавление; удаление; изменение содержимого; переименование; перемещение.

От описания способа хранения информации перейдем к описанию трех крупных операций, которые позволили бы работать с программой, размещенной в базе данных так, как если бы она размещалась в файлах.

3. Импорт программ в базу данных

Существующая практика программирования предусматривает написание программы в виде множества файлов с исходными текстами функций, процедур, объектов программы. К использованию файлов приспособлено большинство IDE, систем управления версиями, а также средств анализа программ. Очень сложно единовременно перейти от файловой формы организации исходных текстов программ к базам данных, не имея готовых аналогов существующим средствам поддержки разработки программ. При работе с этими базами данных предполагается, что наряду с хранимыми в базе данных процедурами и функциями также будут храниться и файлы с их исходными текстами, позволяющими использовать стандартные средства разработки.

```
class Students {
   public:
      void
calculate_average_ball()
      {
         float sum = 0;
         for (int i = 0; i < 5; ++i)
{
            sum += scores[i];
         }
         average = sum / 5.0;
      }
      char * name;
      int scores[5];
   private:
      float average;
};
```

Рис. 2. Пример класса на языке C++

В основе импорта программ в базу данных лежит синтаксический разбор текста. Чтобы сделать его универсальным, необходимо использовать формальное описание грамматики используемого языка программирования. В нашем случае применяется описание в форме Бэкуса—Наура (БНФ), где программа представляется как одно из допустимых высказываний. Высказывания при этом описываются правилами-продукциями из терминалов (терминальный символ) или нетерминалов (нетерминальный символ). Терминалы являются неделимой конструкцией. Нетерминалы также описываются продукциями из терминалов или нетерминалов. Рассмотрим небольшой класс на языке C++, отображенный на рис. 2. В этом классе есть один метод и три различных поля — члены класса. Метод и два члена класса имеют атрибут public, в то время как последний член класса — private.

Для синтаксического разбора примера класса воспользуемся грамматикой языка C++, описанной в черновой копии стандарта № 11 от 16.01.2012 г., находящейся в свободном доступе в сети Интернет[2]. В приложении А этого стандарта приведены правила грамматики языка C++, заданные в форме, близкой к БНФ. На рис. 3 изображено неполное дерево синтаксического разбора примера класса. Пропущен-

[2] http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2012/n3337.pdf

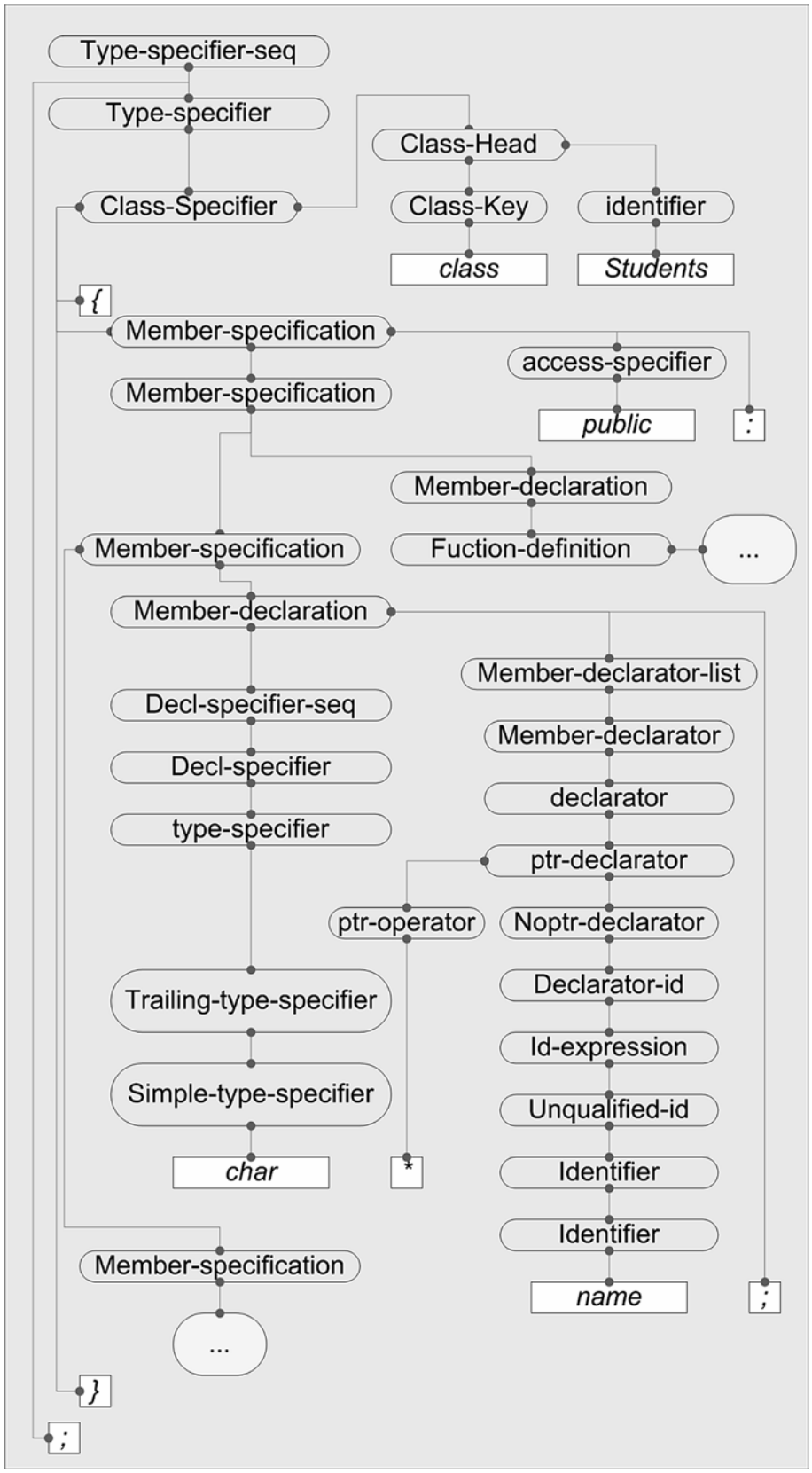

Рис. 3. Дерево синтаксического разбора примера класса

ные фрагменты выделены и подписаны троеточием. В остальном дереве терминалы записаны курсивом, а нетерминалы — прямым шрифтом. Дерево разбора изображено таким образом, что при прочтении терминалов сначала слева направо далее сверху вниз их порядок совпадает с исходным текстом программы. Как видно на рис. 3, на 11 терминалов приходится 30 нетерминалов. Причем 5 из 11 терминалов являются символами пунктуации. Кроме обилия дополнительных нетерминалов следует отметить, что дерево синтаксического разбора довольно чувствительно к изменению правил грамматики. Для упомянутого выше примера класса был построен вариант абстрактного семантического дерева, частично изображенного на рис. 4, см. третью сторону обложки.

В отличие от дерева синтаксического разбора на нем изображено только 8 символов, подобных нетерминалам БНФ, и 12 символов, подобных терминалам БНФ. Абстрактное семантическое дерево имеет меньше элементов, меньше связей, и следовательно, легче воспринимается и лучше подходит для анализа.

Для построения абстрактного семантического дерева в системе используется следующий подход. Параллельно грамматике языка с использованием БНФ записываются правила построения семантического дерева языка программирования. Каждый объект абстрактной семантики описывается деревом с четырьмя типами вершин: объект; элемент; альтернатива; список. Причем каждый узел имеет дополнительный атрибут — опциональный или обязательный. Между правилами абстрактной семантики и правилами БНФ устанавливается отображение один ко многим, точнее всем объектам и элементам абстрактной семантики назначается множество нетерминалов из БНФ. Далее после построения дерева синтаксического разбора выполняется следующий алгоритм преобразования, стартующий с корневой вершины. На шаге 1 ищется прообраз вершины в правилах абстрактной семантики. Если такой прообраз не находится, то алгоритм поочередно переходит к каждой связанной вершине и повторяется шаг 1. Если прообраз находится, то выполняется переход к шагу 2 — в результирующее абстрактное семантическое дерево добавляется вершина. Шаг 3 — для каждого элемента в правилах абстрактной семантики ищется образ, спуск при этом происходит по связанным узлам к листьям в дереве синтаксического разбора. Если для обязательного объекта или элемента образ не найден, то трансформация дерева заканчивается с ошибкой. Если образ найден, то осуществляется переход к шагу 2. Алгоритм работает до тех пор, пока остаются непросмотренные нетерминальные вершины.

Для помещения исходного кода в базу данных выполняется обход абстрактного семантического дерева разобранного файла. Для каждого объекта в дереве выполняются процедуры, записывающие в результирующий скрипт набор параметризованных команд. В качестве параметров команд могут использоваться значения элементов и объектов, связанных с выбранным объектом дерева. За счет выполнения двух процедур — на входе и на выходе, результирующий скрипт может иметь сложную структуру. Выполнение этого скрипта в базе данных приводит к наполнению исходного кода программы.

4. Экспорт программ из базы данных

Экспорт программы из базы данных необходим в первую очередь, когда требуется получить стандартное представление программы в виде файлов. В большинстве случаев в качестве элементов программы выбираются такие синтаксические конструкции языка программирования, которые могут быть вынесены в отдельный файл и скомпилированы.

Следует также отметить, что при разделении на файлы необходимо учитывать взаимосвязь программных элементов, которая проявляется в конструкциях подключения внешних модулей. В системе используется топологическая сортировка, в результате которой программные элементы сортируются таким образом, что первую группу элементов составляют те, которые не зависят от элементов внутри программы. Следующая группа элементов зависит только от элементов первой группы, третья группа — от элементов первой и второй групп и так до последней группы. Порядок следования элемента после топологической сортировки определяет порядок подключения внешних модулей. Список внешних модулей для каждого экспортируемого файла определяется объединением зависимостей программных элементов, включенных в этот файл.

Размещение каждого программного элемента в отдельный файл породило бы огромное число файлов, с которым было бы трудно работать. В базе данных для разделения программы на файлы предусмотрено поле SFILE в таблице CODE_CONTENT. При переходе к базе данных, как к основному месту хранения программы, отсутствие жесткой привязки к файлам добавляет “идеологическую” гибкость средствам редактирования программ. Поэтому необходима стратегия разделения программы на файлы.

Первый вариант стратегии разделения программы на файлы — использование исходных файлов, в которых размещались программные элементы. Это наиболее простая стратегия. Для ее реализации необходимо сохранять неизменным поле SFILE на протяжении жизненного цикла программы. Такая стратегия сохраняет привычное для пользователя деление программы на файлы и может рассматриваться как компромиссная при вынужденном использовании базы данных. Второй вариант стратегии — по-

строение файлов только для первого уровня иерархии программных элементов. В случае программ на языке C++ такой подход означает создание отдельного файла для каждого класса и пространства имен. Эта стратегия хороша тем, что число файлов и наименование каждого файла понятны пользователю. Поле SFILE заполняется непосредственно в процессе экспорта. Третий вариант стратегии разделения программы на файлы — создание минимального числа файлов. Этот вариант может рассматриваться для ускорения компиляции программы, например, при передаче программы в эксплуатацию. Для распределения программных элементов по файлам используется информация о программных настройках кода из таблицы CODE_CONFIG. Для наименования файлов при этом используется поле NAME из этой же таблицы. Все программные элементы с одинаковыми настройками записываются в один файл. Все три стратегии позволяют использовать посторонние средства управления версиями и анализа программ. Выбор стратегии разделения файлов определяется настройками проекта в таблице PROJECT_LIST — поле FILE_STRATEGY. Технически возможна также реализация других вариантов стратегии разделения на файлы.

Поскольку в системе ведется журнал изменений программных элементов, система может экспортировать проект на любой фиксированный момент времени. Для этого из таблицы CODE_CONTENT для каждого программного элемента выбирается содержимое с максимальной датой DTIME, которое меньше заданной. Если в поле STATE у этого содержимого установлен флаг deleted, то программный элемент считается удаленным с этого момента времени DTIME. Отсутствие записей означает, что программный элемент еще не был создан до выбранного момента времени. Кроме того, поле STATE содержит набор флагов {fixed, stable, edited, deleted, embedded, declaration}, описывающих состояние программного элемента. Такой набор соответствует состоянию программного элемента, полученного из исходного файла, работоспособного элемента, редактируемого пользователем элемента, удаленного элемента, агрегированного элемента, а также декларации элемента. Для уменьшения взаимного влияния пользователей, каждый из них может экспортировать программу с использованием своих редактируемых элементов либо экспортировать фиксированное состояние программы.

Экспортируемый текст программы помещается изначально во внутреннее хранилище — в таблицы STORAGE и SCONTENT. Далее программа-клиент извлекает из них файлы и записывает их на компьютер пользователя. При таком подходе возникает возможность генерировать текст только тех файлов, содержимое которых реально изменилось. Для различных программных элементов необходимо генерировать различные типы файлов. Благодаря флагу embedded поля STATE система отличает самостоятельные программные единицы от несамостоятельных, таких как inline — метод или класс, определенный внутри функции. Благодаря флагу declaration система отмечает описание деклараций, например, класса или функции, от реализации методов или переменных. Это позволяет системе более корректно выбирать место размещения программного элемента в экспортируемых файлах. Для формирования текста файлов необходимы заголовок, основная и заключительная части. На языке C++ заголовок содержит широкий набор директив #include и #define. Часть этих директив определяется из таблиц, контролируемых таблицей CODE_CONFIG. Другая часть определяется внутренними взаимосвязями программных элементов. Основная часть файла извлекается из поля CONTENT таблицы CODE_CONTENT экспортируемых программных элементов. Заключительная часть файла определяется языком и типом экспортируемых программных элементов. Так, в языке C++ include-файлы было принято защищать от рекурсивного подключения парными директивами #ifdef, #endif, где вторая директива располагается в самом конце файла. Процедура экспорта формирует результирующий текст за две стадии. На первой стадии генерируется скрипт экспорта. На второй стадии этот скрипт выполняется для получения конечного текста. Такой подход позволяет поэтапно контролировать весь процесс экспорта. Так, на первом этапе выполняется отладка процедуры генерации скрипта, который оперирует вызовами таких функций, как BeginCPPfile, BeginHfile, AddCppCodeUnit, EndCppfile, EndHfile. На втором этапе выполняется отладка самих функций. Двухстадийный подход позволит сократить затраты на разработку при добавлении в систему нового языка.

5. Модификация программ

Для редактирования программ применяется технология распределенных объектов, подобная той, что используется в технологиях OLE/ActiveX и CORBA. При этом принимается во внимание то обстоятельство, что база данных выступает в качестве одной из сторон взаимодействия. В нашем случае существуют две изолированные системы в виде программы-клиента, предоставляющей интерфейс редактирования программ, и базы данных, предназначенной для хранения и поддержания целостности представления программ. Программа-клиент реализована на языке C++. В качестве СУБД используется Oracle со встроенным расширением языка SQL — PL/SQL. В процессе программирования в СУБД использовались простые конструкции языка, что позволяет перенести код базы данных на PostgresSQL.

В разд. 2 была описана схема таблиц базы данных. Для доступа к данным большинства из этих таблиц реализованы отдельные пакеты процедур (packages), наименование которых отличается от наименования таблицы префиксом “P”. В рамках реализации технологии распределенных объектов каждый пакет имеет набор стандартных процедур и функций: AddNew; Get; Edit; Delete; Query, а также набор дополнительных процедур, определяемых спецификой таблицы. Большинство процедур в качестве одного из входных параметров принимают идентификатор записи — ключевое поле таблицы. Процедуры могут также принимать дополнительные параметры по значению (параметры IN) и по ссылке (параметры INOUT); возвращать значения параметрам, объявленным специальным образом (параметры OUT). В предложенной технологии распределенных объектов предусматривается, что некоторые функции пакетов могут возвращать списки объектов. В частности, стандартная функция Query при реализации без входных параметров должна возвращать содержимое всей таблицы. Вызов функции или процедуры может завершиться ошибкой. При этом никакие значения объектов не меняются, а на клиентскую сторону возвращаются код ошибки и ее краткое текстовое описание. Это описание может быть занесено в журнал транзакций на стороне клиента или представлено непосредственному пользователю. Синтаксис вызова функций и процедур PL/SQL подразумевает запись обращения в виде "PackageName.ProcedureName". Последнее обстоятельство указывает на аналогию с объектами язык C++, где каждая запись в таблице интерпретируется как отдельный объект, а идентификатор записи — как указатель или адрес объекта. Технически, начиная с восьмой версии, Oracle предоставляет возможность описания объектно-ориентированных типов. Однако, во-первых, это описание не согласуется с аналогичными возможностями СУБД PostgresSQL, а во-вторых, экспериментальным путем было выяснено, что производительность обращений к методам объектов существенно ниже производительности обращений к процедурам пакетов.

Семантика стандартной процедуры AddNew аналогична оператору new языка C++. По окончании выполнения эта процедура возвращает идентификатор вновь созданного объекта — записи в соответствующей таблице. Процедура может возвращать значения полей нового объекта. При наличии на стороне клиента идентификатора объекта, значение всех его полей может быть получено при помощи процедуры Get, принимающей в обязательном порядке единственный параметр — собственно сам идентификатор. Для изменения полей объекта вызывается процедура Edit, принимающая в качестве входных параметров идентификатор объекта и значения всех его полей. По окончании выполнения процедура Edit возвращает значения всех полей объекта, зафиксированных в базе данных. Процедура Delete удаляет объект — запись из базы данных. Это низкоуровневая процедура, которая может завершиться с ошибкой при наличии ссылок на объект со стороны других таблиц (объектов). Для корректного удаления объектов необходимо вызывать процедуры, обладающие спецификой, характерной для каждой конкретной таблицы.

Таблицы базы данных — это места длительного хранения объектов. С точки зрения программы-клиента, распределенные объекты существуют значительно дольше любой из сессий ее работы. По этой причине взаимодействие между программой-клиентом и SQL-сервером должно выполняться так, чтобы клиенту доставалась только необходимая информация.

Схема взаимосвязей элементов технологии распределенных объектов представлена на рис. 5. Серверная часть схемы рассмотрена далее. Для реализации технологии распределенных объектов на стороне клиента написана библиотека базовых классов, состоящая из базового удаленного объекта DoObject, класса значения ссылки на объект DoHolder, базового класса значения поля удаленного объекта DvValue, а также нескольких конечных классов значений полей удаленного объекта DvNumber, DvString, DvCLOB. Объекты каждого из упомянутых классов обладают указателем на объект доступа к базе данных, указателем на собственное значение и наименованием пакета, манипулирующего данными

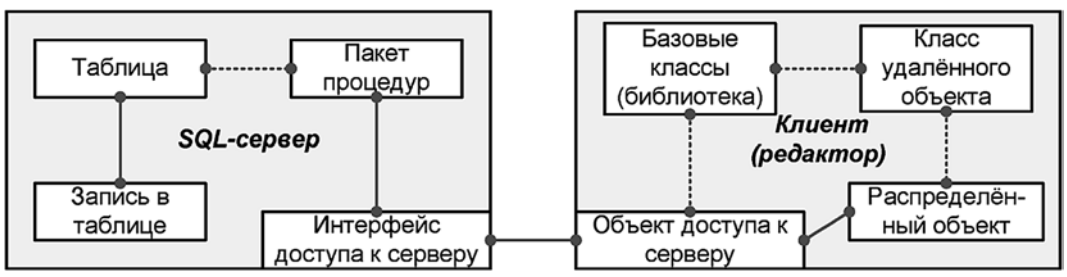

Рис. 5. Технология распределенных объектов для доступа к базе данных

на стороне сервера. Значение объекта класса DoObject является идентификатором объекта на стороне сервера. Значение объекта класса DoHolder является указателем на объект класса DoObject.

Схема базы данных, изображенная на рис. 1 (см. третью сторону обложки), реализована на сервере в виде набора таблиц и парных им пакетов процедур. В клиентской программе каждому пакету процедур ставится в соответствие класс удаленного объекта, формируемый при помощи макросов C++. Описание строится из множества вложенных друг в друга блоков. Главный блок начинается макросом BeginDistributedObject(PackageName), заканчивается макросом EndDistributedObject(PackageName) и предназначен для формирования объекта из класса DoObject. Внутри главного блока находится множество описаний полей и блоков описания методов распределенного объекта. Для описания полей используются макросы вида Member(DistributedType, Name), где DistributedType — это название класса распределенного объекта, связывание с которым произойдет через объект доступа к базе данных. Блоки методов описываются между парными макросами BeginProcedure(Name), EndProcedure(Name) или BeginFunction(Name), EndFunction(Name). Внутри блока методов описываются параметры при помощи макроса MethodParam(DistributedType, Name). Для функций также описывается возвращаемый курсор значений при помощи макроса BeginReturn(DistributedType), EndReturn(DistributedType), ReturnField(FieldName), где FieldName может ссылаться не только на поля собственно возвращаемого объекта, но и на поля его членов.

На стороне клиента главным элементом схемы является объект доступа к серверу. Этот объект поддерживает канал доступа к базе данных и хранит кэшированные списки объектов. Как следствие, при получении идентификатора объекта из базы данных на стороне клиента всегда будет существовать только одна версия распределенного объекта с таким идентификатором. Во время создания объекта доступа к базе данных в нем регистрируется ее схема. Таким образом, этот объект будет иметь полное представление о множестве таблиц, пакетов, процедур и функций для управления данными. После регистрации схемы происходит авторизация в базе данных и получаются необходимые первичные объекты для работы, такие как список PROJECT_TYPE и объект LAST_PROJECT. Далее все распределенные объекты на стороне клиента появляются в результате работы графического пользовательского интерфейса.

Заключение

В настоящей статье представлен подход к использованию базы данных в качестве средства хранения исходных текстов программ. Возможности баз данных существенно превышают сервис, предоставляемый файловыми системами, поэтому рано или поздно программирование обратится к открывающимся на рассматриваемом направлении перспективам. Авторы предполагают, что использование базы данных позволит сократить расстояние между разработкой, компиляцией и анализом программы. В статье рассмотрен один из вариантов того, как это может быть сделано за счет построения промежуточного представления, позволяющего одновременно с изменением программы проводить ее инкрементальный анализ. Подход предусматривает импорт исходных текстов программы из файлов; разбиение программы в процессе импорта на программные элементы; отделение программных элементов от окружения; раздельное редактирование и ведение журнала изменения программных элементов; построение дерева зависимостей программных элементов и экспорт программы в файлы с учетом этих зависимостей. Вследствие ограниченности формата статьи рассмотрены только основные аспекты, подчеркивающие разницу между представлением программы в виде файлов и предложенным представлением в базе данных. Авторы рассчитывают посвятить отдельные публикации более детальному рассмотрению вопросов формирования абстрактного семантического дерева и скриптов импорта и экспорта исходных файлов программы в базу данных и из базы данных. За пределами статьи остался также вопрос применения предложенного представления в базе данных. Необходимо отметить, что авторами ведутся работы над средствами анализа и распараллеливания программ. В частности, разрабатываются средства построения моделей программ в терминах расширений сетей Петри, а также средства трансформации программ. В настоящее время система находится еще в очень раннем — "сыром" состоянии и можно ожидать, что по мере ее "взросления" произойдут значительные изменения в схеме таблиц базы данных и в алгоритмах обработки исходных текстов. Однако основные идеи, представленные в настоящей статье, останутся неизменными.

Список литературы

1. **Naur P. and Randell B. (Eds.).** Software Engineering: Report of a conference sponsored by the NATO Science Committee. Garmisch, Germany, 7—11 Oct. 1968. Brussels, Scientific Affairs Division, NATO. 1969. 231 pp.

2. **Hoare C. A. R.** The verifying compiler: a grand challenge for computing research // Journal of the ACM. 2003. Vol. 50, N. 1. P. 63—69.

3. **Krone J., Ogden W., Sitaraman M., Weide B.** Refocusing the verifying compiler grand challenge. Clemson University, School

of Computing, 2008. URL: http://www.cs.clemson.edu/resolve/research/reports/RSRG-08-01.pdf

4. **Зубов М. В., Пустыгин А. Н., Старцев Е. В.** Применение универсальных промежуточных представлений для статического анализа исходного программного кода // Доклады Томского гос. ун-та систем управления и радиоэлектроники. 2013. № 1. С. 64—68.

5. **Аветисян А. И., Иванников В. П., Гайсарян С. С.** Анализ и трансформация программ // Всероссийский конкурсный отбор обзорно-аналитических статей по приоритетному направлению "Информационно-телекоммуникационные системы". 2008. 78 с.

6. **Катаев Н. А.** Особенности внутреннего представления системы САПФОР // Параллельные вычислительные технологии (ПаВТ'2013): труды междунар. науч. конф. Челябинск, 31 марта — 5 апреля 2013 г. Челябинск: Издательский центр ЮУрГУ, 2013. С. 380—386.

7. **Петренко В. В.** Внутреннее представление Reprise распараллеливающей системы // Труды Четвертой междунар. конф. "Параллельные вычисления и задачи управления" PACO'2008. Москва, 27—29 октября 2008 г. М.: Институт проблем управления им. В. А. Трапезникова РАН, 2008. С. 1241—1245.

D. I. Kharitonov, Senior Research Fellow, **G. V. Tarasov,** Research Fellow,
R. V. Parakhin, Postgraduate Student, e-mail: fadak@dvo.ru,
Institute of Automation and Control Processes FEB RAS, Vladivostok

On Database Usage for Storing Source Code of Programs

An approach to use databases for storing source code of programs is considered. Relational model for representing programs in databases is presented. Import and export operations for transferring program's source code from file representation to relational representation, and vice versa, are described. Object-oriented interface to access to databases objects from application programs is also reviewed. In conclusion a number of research areas are presented, in which the storing program in databases could provides some advantages over traditional file representation.

Keywords: *relational database management systems, program translation, object-oriented programming*

References

1. **Naur P., Randell B. (Eds.).** Software Engineering: Report of a conference sponsored by the NATO Science Committee. Garmisch, Germany, 7—11 Oct. 1968. Brussels, Scientific Affairs Division, NATO. 1969. 231 p.

2. **Hoare C. A. R.** The verifying compiler: a grand challenge for computing research. *Journal of the ACM*, 2003, vol. 50, no. 1, pp. 63—69.

3. **Krone J., Ogden W., Sitaraman M., Weide B.** *Refocusing the verifying compiler grand challenge.* Clemson University, School of Computing, 2008, available at: http://www.cs.clemson.edu/resolve/research/reports/RSRG-08-01.pdf

4. **Zubov M. V., Pustygin A. N., Starcev E. V.** Primenenie universal'nykh promezhutochnykh predstavleniy dlya staticheskogo analiza iskhodnogo programmnogo koda. *Doklady Tomskogo gos. un-ta sistem upravlenija i radiojelektroniki*. 2013, no. 1, pp. 64—68 (in Russian).

5. **Avetisjan A. I., Ivannikov V. P., Gajsarjan S. S.** Analiz i transformatsiya program. *Vserossijskij konkursnyj otbor obzorno-analiticheskih statej po prioritetnomu napravleniju "Informacionno-telekommunikacionnye sistemy"*. 2008, 78 p. (in Russian).

6. **Kataev N. A.** Osobennosti vnutrennego predstavleniya sistemy SAPFOR. *Parallel'nye vychislitel'nye tehnologii (PaVT'2013): trudy mezhdunar. nauch. konf.* Cheljabinsk, 2013, pp. 380—386 (in Russian).

7. **Petrenko V. V.** Vnutrennee predstavlenie Reprise rasparallelivayushchei sistemy. *Trudy Chetvertoj mezhdunar. konf. "Parallel'nye vychislenija i zadachi upravlenija". PACO'2008*. Moscow, Institute problem upravlenia im. V. A. Trapeznikova, 2008, pp. 1241—1245 (in Russian).

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

Сдано в набор 05.02.2015 г. Подписано в печать 18.03.2014 г. Формат 60×88 1/8. Заказ PI415
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**