ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Программная инженерия

ПрИн

8
2015

Учредитель: Издательство "НОВЫЕ ТЕХНОЛОГИИ"

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

№ 8
August
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 004.453

Е. А. Киселёв, канд. техн. наук, науч. сотр., e-mail: kiselev@jscc.ru,
Межведомственный суперкомпьютерный центр Российской академии наук,
В. В. Корнеев, д-р техн. наук, проф., зам. директора по научной работе,
e-mail: korv@rdi-kvant.ru, ФГУП "НИИ "Квант", г. Москва

Оптимизация отображения программ на вычислительные ресурсы

Предложены и проанализированы способы уменьшения времени выполнения параллельных программ путем оптимизации их размещения на ресурсах суперкомпьютера. Приведены результаты оптимизации запуска теста HPCG на ресурсах суперкомпьютера "МВС-10р" с помощью разработанных программных средств. Дана временная оценка работы алгоритма отображения PCSA.

Ключевые слова: *суперкомпьютер, тест HPCG, отображение программ, информационный граф, латентность коммуникационной среды, анализ структуры коммуникационной среды, анализ структуры параллельной программы, метод моделирования отжига, MPI-программы*

Введение

Программные тесты, которые применяют для оценки производительности вычислительных систем (ВС), как правило, моделируют типовой набор решаемых на этой системе прикладных задач и соответствующих им программ. По этой причине приемы повышения производительности исполнения тестов могут быть использованы для программ этого набора. Для объективной оценки производительности вычислительных систем тест *High Performance LINPAC* (HPL) [1], моделирующий решение системы линейных алгебраических уравнений (СЛАУ) вида $\mathbf{A}x = \mathbf{b}$ методом LU-разложения с выбором ведущего столбца, где $\mathbf{A}$ — плотно заполненная квадратная матрица размера N, а $\mathbf{b}$ — плотно заполненный вектор-столбец, дополняется новым тестом *High Performance Conjugate Gradient* (HPCG) [2]. В программной реализации теста HPCG применяется метод сопряженных градиентов с предобусловливанием [3] для решения СЛАУ вида $\mathbf{A}x = \mathbf{b}$, где $\mathbf{A}$ — симметричная положительно определенная разреженная квадратная матрица размера N и вектор-столбец $\mathbf{b}$ сгенерированы с использованием нескольких шаблонов разреженности. При оценке на тесте HPL, если имеется достаточный объем памяти узла, удается найти размер N, при котором длительность обмена данными между вычислительными узлами ВС существенно меньше, чем время вычислений в узлах. Это делает показатель производительности на тесте HPL слабо зависящим от параметров коммуникационной среды. В отличие от HPL, тест HPCG не позволяет варьировать объем вычислений узла на единицу передаваемых данных. По этой причине этот тест учитывает не только возможности вычислений с плавающей точкой, но и коммуникативные механизмы, поддерживающие эти вычисления [4, 5].

Для получения более высоких оценок производительности на тесте HPCG для конкретной ВС можно воспользоваться изложенной в работе [6] методикой оптимизации размещения параллельных программ на вычислительных узлах многопроцессорной вычислительной системы (МВС).

Оптимизация размещения программы теста HPCG состоит в выполнении следующих этапов:

- построение информационного графа программы теста HPCG;
- построение графа подсистемы МВС, на которой планируется осуществить запуск теста;
- вычисление оптимального отображения информационного графа программы HPCG на структуру графа подсистемы МВС;
- запуск теста HPCG в соответствии с построенной алгоритмом отображения схемой размещения ветвей программы на ядрах подсистемы МВС.

В настоящей работе демонстрируется применение процедуры оптимизации размещения теста HPCG на процессорах "МВС-10р" Межведомственного суперкомпьютерного центра (МСЦ РАН). Тест HPCG рассматривается как "черный ящик", что иллюстрирует возможность оптимизации выполнения параллельных программ, отчужденных от разработчиков.

Информационный граф теста HPCG

В информационном графе параллельной программы, использующей MPI (*Message Passing Interface*), вершины моделируют MPI-процессы, а ребра — информационные обмены между ними. Каждому ребру информационного графа назначается вес, равный усредненному объему данных, переданных между MPI-процессами за время выполнения программы.

В структуре информационного графа выделяют коммуникационные шаблоны — типовые схемы взаимодействия между MPI-процессами. Как правило, параллельная программа использует фиксированный набор коммуникационных шаблонов, которые повторяются в процессе ее выполнения. Каждый коммуникационный шаблон описывается отдельным информационным графом. Учет коммуникационных шаблонов при распределении MPI-процессов параллельной программы по процессорам МВС позволяет выявлять конкуренцию за коммуникационные ресурсы на различных этапах выполнения программы.

Для автоматизации процесса построения информационного графа и коммуникационных шаблонов параллельной программы разработано программное средство IGTrace [7]. Это программное средство позволяет получить информацию о числе операций приема/передачи данных, объеме переданных данных между ветвями программы и о числе вычислительных операций, выполненных каждой ветвью. Построение информационного графа и выделение коммуникационных шаблонов выполняется исключительно на основе данных трассировки MPI-приложения, которое рассматривается как "черный ящик".

Процесс построения информационного графа теста HPCG с помощью IGTrace осуществляется в два этапа (рис. 1). На первом этапе готовится файл трассы приложения. Для этого используется программное средство трассировки VampirTrace [8]. На этапе сборки параллельной программы компиляторы VampirTrace (vtcc, vtf90) автоматически встраивают в исходный текст программы функции, которые необходимы для формирования файлов трассы. После запуска приложения на ресурсах МВС в рабочем каталоге программы формируется набор файлов трассы в формате otf (Open Trace Format) [8]. На втором этапе выполняется запуск программы IGTrace с указанием: имени сформированного otf-файла; формата .graphml или .dot файла для хранения информационного графа; типа (коммуникационные шаблоны или агрегированный граф) информационного графа.

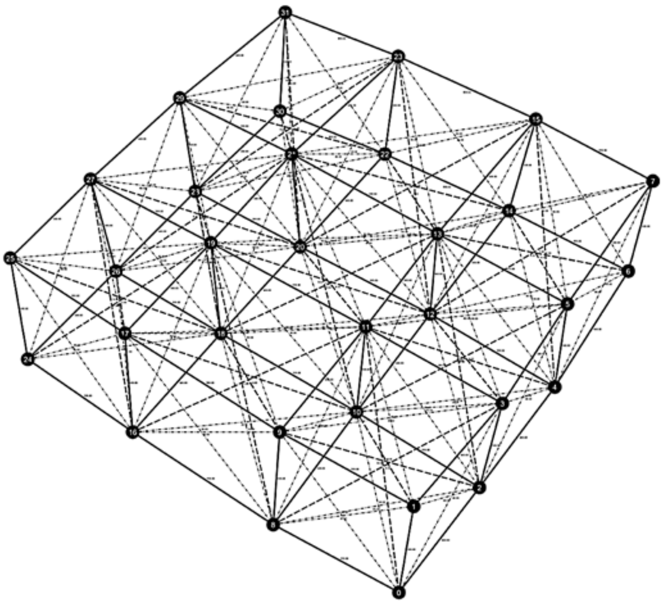


Рис. 2. Информационный граф запущенного на 32 процессорах теста HPCG

Пример информационного графа теста HPCG, запущенного на 32 процессорах, приведен на рис. 2.

Анализ информационного графа теста HPCG, полученного при различном числе вычислительных ядер (16, 32, 64, 128, 256, 512, 768, 1024) программным средством IGTrace, показал, что в тесте используются коммуникационные шаблоны трех типов (рис. 3, см. вторую сторону обложки). В коммуникационном шаблоне типа 1 обмены между MPI-процессами выполняются по схеме "3D-решетка". Этот шаблон используется для передачи 90,1 % данных. В коммуникационном шаблоне типа 2 выполняются диагональные пересылки между MPI-процессами, образующими грани "3D-ячейки". Шаблон типа 2 используется для передачи 9 % данных. В коммуникационном шаблоне типа 3 выполняются диагональные пересылки между ветвями, образующими одну "3D-ячейку". Коммуникационный шаблон типа 3 используется для передачи 0,9 % данных.

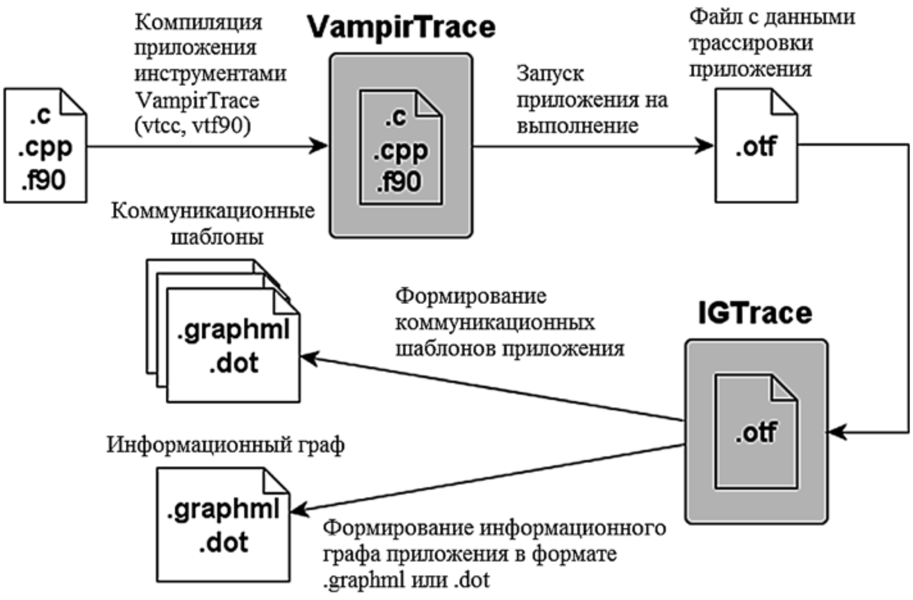


Рис. 1. Этапы построения информационного графа MPI-приложения программным средством IGTrace

Граф многопроцессорной вычислительной системы

Структура МВС представляется в виде дерева, содержащего M уровней. Каждый уровень образуется отдельным видом вычислительных компонентов МВС (стойки, вычислительные узлы, процессоры), которые объединены линиями связи своего уровня. Каждый уровень дерева описывается отдельным графом МВС, вершины которого моделируют вычислители (вычислительные узлы, процессоры, ядра), а ребра — каналы связи между ними.

Для автоматизации процесса построения графа МВС разработаны программные средства измерения и визуализации латентности коммуникационной среды. Латентность коммуникационной среды определяется с точностью до микросекунды для каждой пары вычислителей (ядра, процессоры или вычислительные узлы). Результаты измерений представляют в виде матрицы **M** размера n, где n — число вычислителей.

Для визуализации результатов тестирования и выделения уровней иерархии коммуникационной среды разработано кроссплатформенное программное средство GLVisual [9]. Программное средство кластеризует вычислители методом k-средних на основании значений латентности матрицы **M** и отображает их в виде трехмерной диаграммы. Для каждого кластера приводится список входящих в его состав вычислителей и среднее значение латентности передачи данных между его вычислителями и вычислителями других кластеров.

На рис. 4 представлена диаграмма латентности коммуникационной среды "МВС-10р". На диаграмме отображены три уровня иерархии коммуникационной среды. Первый уровень иерархии коммуникационной среды — уровень взаимодействия вычислительных ядер в рамках одного микропроцессора. Второй уровень иерархии — уровень взаимодействия процессоров одного вычислительного узла (ВУ). Третий уровень иерархии — уровень взаимодействия ВУ МВС.

Поскольку ВУ МВС объединены по топологии *"Non blocking fat tree"*, структура коммуникационной среды на третьем уровне взаимодействия может быть представлена полным графом. В каждом ВУ используется два 8-ядерных микропроцессора Intel Xeon, поэтому на втором уровне иерархии каждый ВУ описывается информационным графом из двух смежных вершин, моделирующих два микропроцессора. На первом уровне иерархии каждый микропроцессор представлен полным графом с восемью вершинами. Пример информационного графа МВС из 16 ВУ представлен на рис. 5.

Отображение информационного графа HPCG на структуру графа МВС

Отображение информационного графа параллельной программы $I = (V, E)$, где V — множество вершин, моделирующих MPI-процессы, а E — множество ребер, моделирующие информационные обмены, на граф МВС $S = (C, L)$, где C — множество вершин, моделирующих вычислительные устройства (вычислительные узлы, процессоры, ядра), а L — множество ребер, моделирующих коммуникационные каналы между вычислительными устройствами, обозначается как ϕ: $V \to C$ и представляется матрицей $\mathbf{X}_\phi$:

$$\mathbf{X}_\phi = (x_{ij}),$$

где $x_{ij} = \begin{cases} 1, \phi(v_i) = c_j \\ 0, \phi(v_i) \neq c_j \end{cases} \quad v_i \in V, c_j \in C, |V| = |C|.$

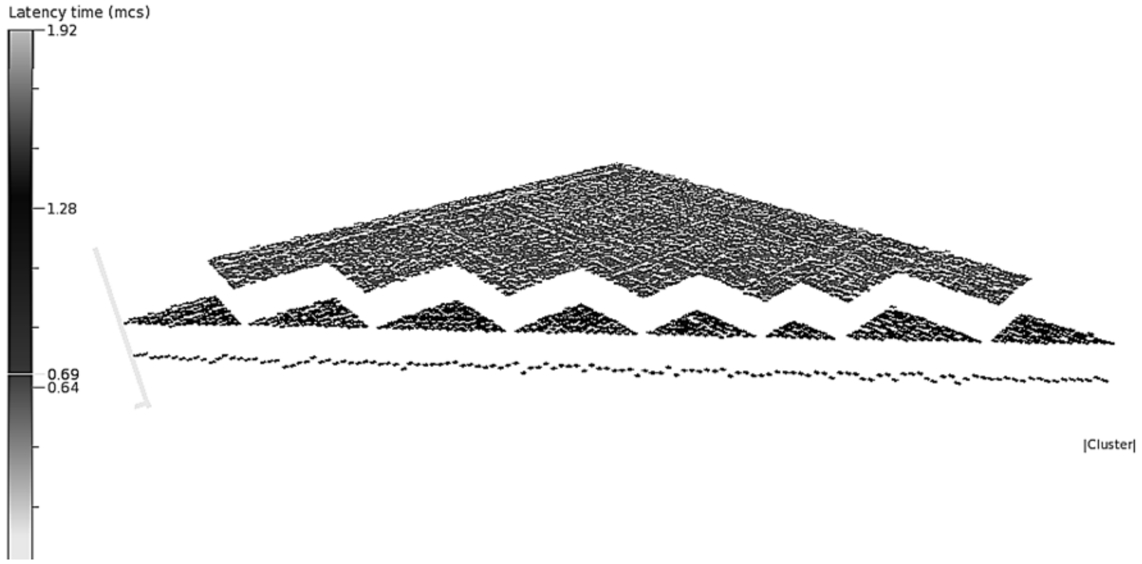
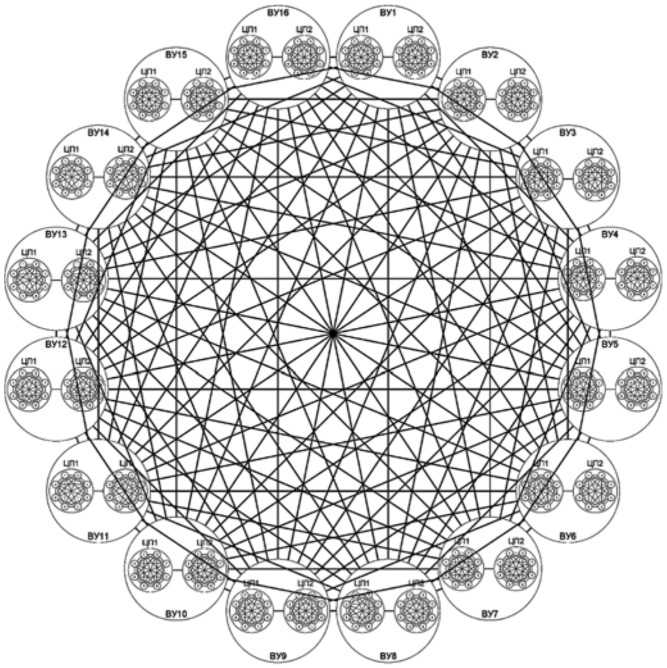


Рис. 4. Визуализация результатов тестирования и выделение уровней иерархии коммуникационной среды



ВУ - вычислительный узел (1) - вычислительное ядро
ЦП - центральный процессор

Рис. 5. Граф МВС из 16 вычислительных узлов (256 вычислительных ядер), объединенных сетью с топологией *"Non blocking fat tree"*

Согласно приведенному выражению элемент x_{ij} матрицы $\mathbf{X}_ф$ равен 1, если вершина с номером i информационного графа программы отображена на вершину с номером j графа МВС, и 0 в противном случае. Отображение информационного графа программы на граф МВС выполняется при условии, что число вершин в информационном графе программы и графе МВС совпадает.

Таким образом, задачу размещения параллельной программы (далее используется сокращение "задача размещения") на ресурсах МВС можно свести к задаче нахождения матрицы $\mathbf{X}_ф$, для которой время выполнения параллельной программы на заданной вычислительной системе наименьшее.

Задача размещения программ является NP-полной. Для МВС с большим числом ВУ (от 30 и более) применяют эвристические алгоритмы, позволяющие за приемлемое для практических случаев время определять рациональное размещение параллельной программы на МВС. На практике применяют более простые алгоритмы размещения, не учитывающие информационный граф программы и граф МВС. К их числу относятся: *Processor Allocation Random* (PAR) — распределение ветвей программы по ВУ МВС случайным образом; *Processor Allocation Linear* (PAL) — выбор первых M свободных ВУ; *Processor Allocation Round Robin* (PARR) — распределение ветвей программы по ВУ системы из N вычислительных узлов в порядке, согласно которому первая ветвь программы размещается на первом ВУ первого по порядку вычислительного узла, вторая — на первом ВУ второго вычислительного узла, ветвь N — на первом ВУ вычислительного узла N, ветвь $N + 1$ — на втором ВУ первого вычислительного узла и т. д.

В работе [6] представлен параллельный алгоритм *Parallel Clustering Simulated Annealing* (PCSA) отображения информационного графа программы на граф МВС. Идея алгоритма PCSA состоит в локализации коммуникационного трафика интенсивно обменивающихся ветвей программы по "близко расположенным" процессорам в рамках заданного сегмента сети МВС и сокращения его влияния на обмены других ветвей программы. Интенсивность взаимодействия ветвей определяется числом операций приема—передачи данных и длительностью их выполнения. Алгоритм PCSA реализуется в два этапа: этап декомпозиции информационного графа на подграфы максимального веса и этап отображения сформированных подграфов на структуру межъядерных связей МВС.

Процесс декомпозиции графа программы состоит из двух этапов — сжатия и разбиения графа. На этапе сжатия исходный граф G_0 преобразуется в последовательность графов G_1, G_2, ..., G_m меньших размерностей, таких, что $|V_0| > |V_1| > |V_m|$. Графы меньших размерностей формируются объединением смежных вершин, инцидентных ребру максимального веса. Сжатие графа $G_i = (V_i, E_i)$ осуществляется стохастическим алгоритмом HEM *(Heavy Edge Matching)* [11], вычислительная сложность которого составляет $O(|E_i|)$. Сжатие осуществляется до тех пор, пока число вершин, сжатых в одну, не превышает число ядер в одном ВУ МВС. Разбиение графа $G_m = (V_m, E_m)$ на k подмножеств осуществляется алгоритмом рекурсивной бисекции GGP *(Graph Growing Partitioning)* [12]. Вычислительная сложность алгоритма составляет $O(|E|\log_2 k)$. Число подмножеств соответствует необходимому для выполнения программы числу ВУ МВС.

После завершения этапа декомпозиции информационного графа выполняется этап отображения информационного графа на граф МВС. Каждому ВУ ставится в соответствие подграф из выделенных на этапе сжатия вершин информационного графа. Отображение подграфа программы на граф, описывающий структуру межъядерных связей ВУ, осуществляется алгоритмом SA *(Simulated Annealing)* [10], основанном на методе моделирования отжига. Вычислительная сложность данного этапа алгоритма составляет $O(c^2)$, где c — максимальное число ядер в процессоре МВС.

Для построения схемы размещения параллельного приложения HPCG на ресурсах МВС необходимо воспользоваться разработанным программным средством GraphMap, реализующим алгоритм PCSA. В качестве входных параметров необходимо указать имена файлов с информационным графом программы и графом МВС в формате .graphml, сформированных программными средствами IGTrace и GLVisual. В результате выполнения приложения GraphMap будет сформирован файл схемы разме-

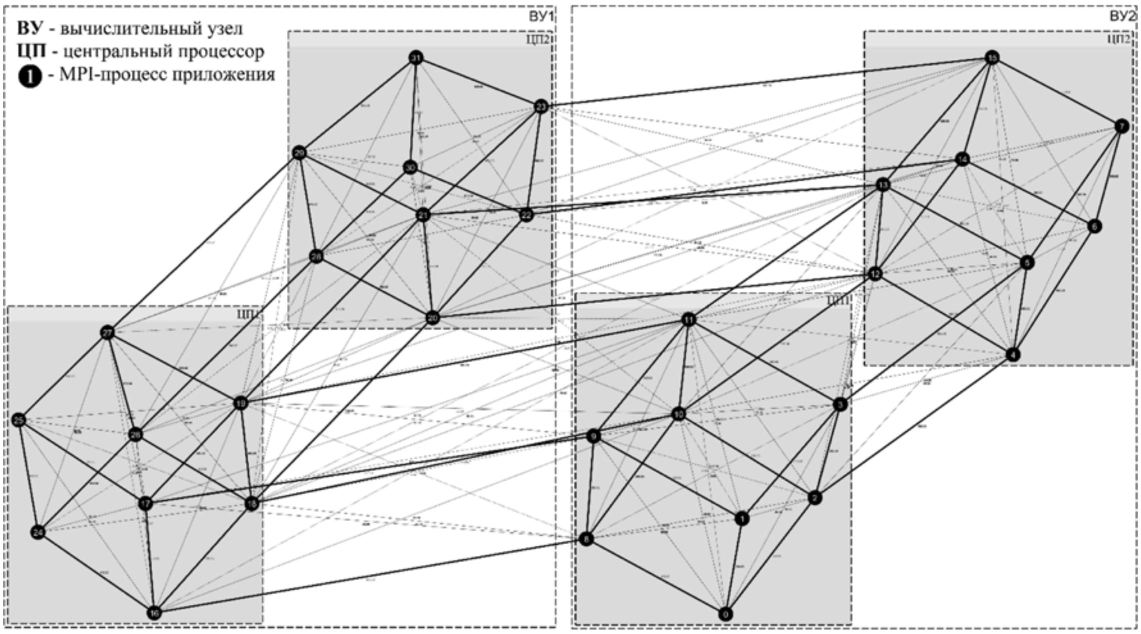


Рис. 6. Схема размещения MPI-процессов теста HPCG по 32 ядрам ВУ "МВС-10р"

щения MPI-процессов программы на ядрах МВС. Схема размещения представляет собой набор строк вида "номер_MPI-процесса > Имя_узла:номер_ядра".

Пример схемы размещения MPI-процессов HPCG по 32 ядрам двух ВУ "МВС-10р", построенной алгоритмом PCSA, представлен на рис. 6.

Апробация алгоритма PCSA и разработанных авторами программных средств проводилась на вычислительных ресурсах МСЦ РАН. Запуск теста HPCG проводился на 256 процессорах "МВС-10р". Точность получаемых результатов обеспечивалась многократным запуском теста HPCG на одних и тех же процессорах МВС.

Ниже приведены результаты запуска теста со стандартной схемой размещения и со схемой размещения, построенной алгоритмом PCSA:

HPSG, Гфлоп/с:

без PCSA 99,24
с PCSA. 109,16

Таким образом, запуск теста HPCG с использованием алгоритма PCSA позволяет увеличить результат измерения производительности более чем на 9 % по сравнению со стандартной схемой размещения. Такой результат достигается благодаря снижению латентности и увеличению пропускной способности при передаче данных между интенсивно взаимодействующими MPI-процессами теста. Применение сформированной алгоритмом PCSA схемы размещения позволило сократить на 7,5 % время выполнения коллективных операций и на 4,8 % время обмена служебной информацией между MPI-ветвями.

На рис. 7 приведены результаты оценки времени (T_{PCSA}) построения схемы размещения программы

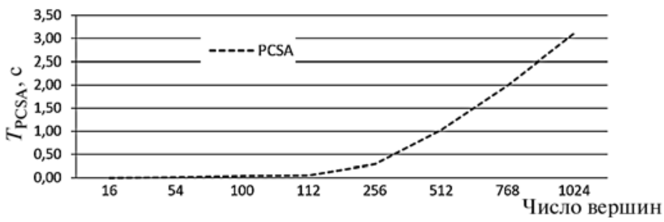


Рис. 7. Оценка времени построения схемы размещения теста HPCG на ресурсах суперкомпьютера "МВС-10р"

на ресурсы МВС с использованием алгоритма PCSA в зависимости от числа вершин в информационном графе теста HPCG.

Заключение

На примере теста HPCG представлены методика и результаты ее применения для оптимизации размещения ветвей параллельной программы на ресурсах суперкомпьютера, показывающие необходимость учета информационных обменов между ветвями и структуры коммуникационной среды суперкомпьютера. Методика позволяет "на лету" оптимизировать размещение параллельных программ с числом ветвей, не превышающем 1024, на выделенных для выполнения ресурсах суперкомпьютера. Для программ с большей степенью параллелизма построение схемы размещения должно выполняться заблаговременно. Предметом дальнейшего исследования является определение возможности корректировки схемы размещения ветвей параллельной программы при условии выхода из строя или замены части запланированных для выполнения вычислительных узлов.

Список литературы

1. **Dongarra J., Luszczek P., Petitet A.** The LINPACK Benchmark: Past, Present, and Future. URL:/ http://web.eecs.utk.edu/~luszczek/pubs/hplpaper.pdf.
2. **Dongarra J., Heroux M.** Toward a New Metric for Ranking High Performance Computing Systems. Sandia Report. SAND2013-4744. Printed June 2013. URL: http://www.sandia.gov/~maherou/docs/HPCG-Benchmark.pdf.
3. **Heroux M., Dongarra J., Luszczek P.** HPCG Technical Specification. Sandia Report. SAND2013-8752. October, 2013. URL: https://software.sandia.gov/hpcg/doc/HPCG-Specification.pdf.
4. **Barrett R., Berry M., Chan T. F.** et al. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. URL: http://www.netlib.org/templates/templates.pdf.
5. **Болдырев Ю. Я.** Зато ближе к реальным приложениям. Интервью с Джеком Донгаррой по поводу новых тестов HPCG (Джек Донгарра, Майкл Эру) // Суперкомпьютеры. 2013. № 16. С. 8—12. URL: http://www.supercomputers.ru/images/stories/Supercomputers_16-2013.pdf.
6. **Киселёв Е. А., Аладышев О. С.** Алгоритм эффективного размещения программ на ресурсах многопроцессорных вычислительных систем // Программные продукты и системы. 2012. № 4. С. 18—25.
7. **Киселёв Е. А.** Построение информационного графа параллельной программы на основе данных профилирования и трассировки // Научный сервис в сети Интернет: экзафлопсное будущее: Труды Международной суперкомпьютерной конференции. 19—24 сентября 2011 г., г. Новороссийск. М.: Изд-во МГУ, 2011. С. 541—546.
8. **Henschel R.** Introducing OTF / Vampir / VampirTrac. Technical Report. Center for Information Services and High Performance Computing (ZIH), 2007. URL: https://wiki.eclipse.org/images/9/9e/Ptp_workshop_20070523_vampir.pdf.
9. **Киселёв Е. А.** Программные средства построения коммуникационной схемы многопроцессорной вычислительной системы // Высокопроизводительные вычислительные системы. Материалы 7 Международной научной молодежной школы: сб. науч. тр. Таганрог: Изд-во ТТИ ЮФУ, 2010. С. 76—79.
10. **Lee C., Bic L.** On the mapping problem using simulated annealing // Computers and Communications, Conference Proceedings., Eighth Annual International Phoenix Conference on. IEEE, 1989. P. 40—44.
11. **Karypis G., Kumar V.** Multilevel k-way partitioning scheme for irregular graphs // Journal of Parallel and Distributed computing. 1998. Vol. 48, N 1. P. 96—129.
12. **Karypis G., Kumar V.** A fast and high quality multilevel scheme for partitioning irregular graphs // Journal on Scientific computing. 1999. Vol. 20, N 1. P. 359—392.

E. A. Kiselev, Researcher, e-mail: kiselev@jscc.ru,
Joint Supercomputer Center of the Russian Academy of Sciences, Moscow,
V. V. Korneev, Professor, Director for Research, e-mail: korv@rdi-kvant.ru,
State Enterprise "Research and Development Institute "Kvant", Moscow

Optimization of Mapping Programs to Computing Resources

Possibility to decline execution time of parallel programs with optimization of mapping programs to supercomputer resources is considered. The optimization process performs mapping application graph to parallel computer system graph. An application graph represents aggregate information on communication operations between parallel application's processes. It contains aggregate information about number of operations transmit/receive and the amount of data transferred for each processes pair. A system graph discloses a parallel computer system's communication structure. For each communication channel it comprises a latency values and bandwidth. Parallel implementation of the simulation annealing algorithm is proposed. The basic idea of the algorithm is the localization of communication exchanges between the intensity of the interaction program branches and reduce competition for communication resources. Building application graph and a parallel system graph are performed automatically using the developed software. Results of optimization test High Performance Conjugate Gradient execution on "MBC-10p" supercomputer resources using created software tools are presented. Time evaluation of mapping algorithm is done.

Keywords: *supercomputer, test HPCG, mapping of program, information graph, latency of communication fabric, analysis of communication fabric structure, analysis of parallel program structure, simulated annealing tool, MPI-program*

References

1. **Dongarra J., Luszczek P., Petitet A.** The LINPACK Benchmark: Past, Present, and Future. available at: http://web.eecs.utk.edu/~luszczek/pubs/hplpaper.pdf
2. **Dongarra J., Heroux M.** Toward a New Metric for Ranking High Performance Computing Systems. Sandia Report. SAND2013-4744. Printed June 2013, available at: http://www.sandia.gov/~maherou/docs/HPCG-Benchmark.pdf.
3. **Heroux M., Dongarra J., Luszczek P.** HPCG Technical Specification. Sandia Report. SAND2013-8752. October, 2013, available at: https://software.sandia.gov/hpcg/doc/HPCG-Specification.pdf.
4. **Barrett R., Berry M., Chan T. F., Demmel J., Donato J. M., Dongarra J., Eijkhout V., Pozo R., Romine C., Van der Vorst H.** Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, available at: http://www.netlib.org/templates/templates.pdf.
5. **Boldyrev Yu. Ya.** Zato blizhe k real'nym prilozheniyam. Interv'yu s Dzhekom Dongarroi po povodu novykh testov HPCG (Dzhek Dongarra, Maikl Eru). *Superkomp'ytery*, 2013, no 16, pp. 8—12, available at: http://www.supercomputers.ru/images/stories/Supercomputers_16-2013.pdf (in Russian).
6. **Kiselyov E. A., Aladyshev O. S.** Algoritm ehffektivnogo razmeshcheniya programm na resursah mnogoprocessornyh vychislitel'nyh sistem. *Programmnye produkty i sistemy*, 2012, no. 4, pp. 18—25 (in Russian).
7. **Kiselyov E. A.** Postroenie informacionnogo grafa parallel'noj programmy na osnove dannyh profilirovaniya i trassirovki. *Nauchnyj servis v seti Internet: ehkzaflopsnoe budushchee: Trudy Mezhdunarodnoj superkomp'yuternoj konferencii*, 19—24 September 2011, Novorossijsk. Moscow: Izd-vo MGU, 2011, pp. 541—546 (in Russian).
8. **Henschel R.** Introducing OTF / Vampir / VampirTrac. Technical Report. Center for Information Services and High Performance Computing (ZIH), 2007, available at: https://wiki.eclipse.org/images/9/9e/Ptp_workshop_20070523_vampir.pdf.
9. **Kiselyov E. A.** Programmnye sredstva postroeniya kommunikacionnoj skhemy mnogoprocessornoj vychislitel'noj sistemy. *Vysokoproizvoditel'nye vychislitel'nye sistemy. Materialy 7 Mezhdunarodnoj nauchnoj molodezhnoj shkoly:* sb. nauch. tr. Taganrog: Izd-vo TTI YUFU, 2010, pp. 76—79 (in Russian).
10. **Lee C., Bic L.** On the mapping problem using simulated annealing. *Computers and Communications, Conference Proceedings, Eighth Annual International Phoenix Conference on*. IEEE, 1989, pp. 40—44.
11. **Karypis G., Kumar V.** Multilevel *k*-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed computing,* 1998, vol. 48, no. 1, pp. 96—129.
12. **Karypis G., Kumar V.** A fast and high quality multilevel scheme for partitioning irregular graphs. *Journal on Scientific computing,* 1999, vol. 20, no. 1, pp. 359—392.

УДК 004.942, 004.42, 004.514

К. К. Абгарян, канд. физ.-мат. наук, доц., и. о. зав. отделом, e-mail: kristal83@mail.ru,
П. А. Сеченых, аспирант, мл. науч. сотр., e-mail: p-sechenyh@mail.ru,
Е. С. Гаврилов, ст. преподаватель, мл. науч. сотр., e-mail: eugavrilov@gmail.com,
Московский авиационный институт (Национальный исследовательский университет),
Вычислительный центр имени А. А. Дородницына Российской академии наук, г. Москва

Объектно-реляционный подход к разработке системы компьютерного моделирования многомасштабной схемы расчета многослойных полупроводниковых наноструктур

Проанализированы особенности разработанной авторами объектно-реляционной архитектуры системы информационной поддержки многомасштабной схемы расчета свойств многослойной полупроводниковой наноструктуры.

Приведено описание некоторых математических моделей, реализованных в вычислительных модулях многомасштабной схемы расчета. Представлено также сравнение баз данных и информационных систем, используемых при решении задач материаловедения. Представлена реляционная модель хранения данных, которая применяется для решения ресурсоемких и разномасштабных задач. Показана реализация пользовательского интерфейса, который позволяет осуществлять выборку данных согласно заданным критериям, а также предоставляет возможность формировать входные данные для расчетных модулей.

Ключевые слова: *база данных, объектно-реляционный подход, многослойная полупроводниковая наноструктура, многомасштабная схема расчета*

Введение

Одной из важнейших современных высоких технологий является получение многослойных полупроводниковых наноструктур (МПНС) с заданными параметрами и прогнозируемыми свойствами. Такая МПНС — неоднородная структура, выращенная на подложке, она состоит из различных полупроводниковых слоев, имеющих определенную толщину (измеряется в нанометрах), химический состав и кристаллическую структуру, которые отличаются, в общем случае, шириной запрещенной зоны [1]. На рис. 1 (см. вторую сторону обложки) представлен пример МПНС.

Представленная в работе информационная система компьютерного моделирования может быть использована для разработки новых технологий, направленных на получение МПНС с уникальными свойствами для высокочастотной полупроводниковой техники нового поколения (транзисторов, усилителей мощности, малогабаритных радаров и т. д.). Такие приборы должны быть высокомощными и при этом обладать минимальными размерами. Приборы и устройства на основе МПНС, включающих слои нитридов алюминия, галлия или индия, находят применения в информационно-телекоммуникационных технологиях авиационных, ракетных и космических систем. Оборудование авиационного и космического применения, малогабаритные радары могут быть усовершенствованы благодаря применению транзисторов на базе МПНС нового поколения.

Схема расчета

На рис. 2 представлена многомасштабная схема расчета многослойной полупроводниковой наноструктуры (МСРПН), разработанная ранее в ВЦ РАН им. А. А. Дородницына [1].

Схема расчета состоит из двух основных блоков и базы данных о структурах и свойствах наносистем. Каждый из блоков соответствует стационарной и динамической моделям. Блоки МСРПН содержат различные расчетные модули, предназначенные для решения определенных задач.

В работе [1] рассматривается многомасштабный подход к решению задачи предсказательного ком-

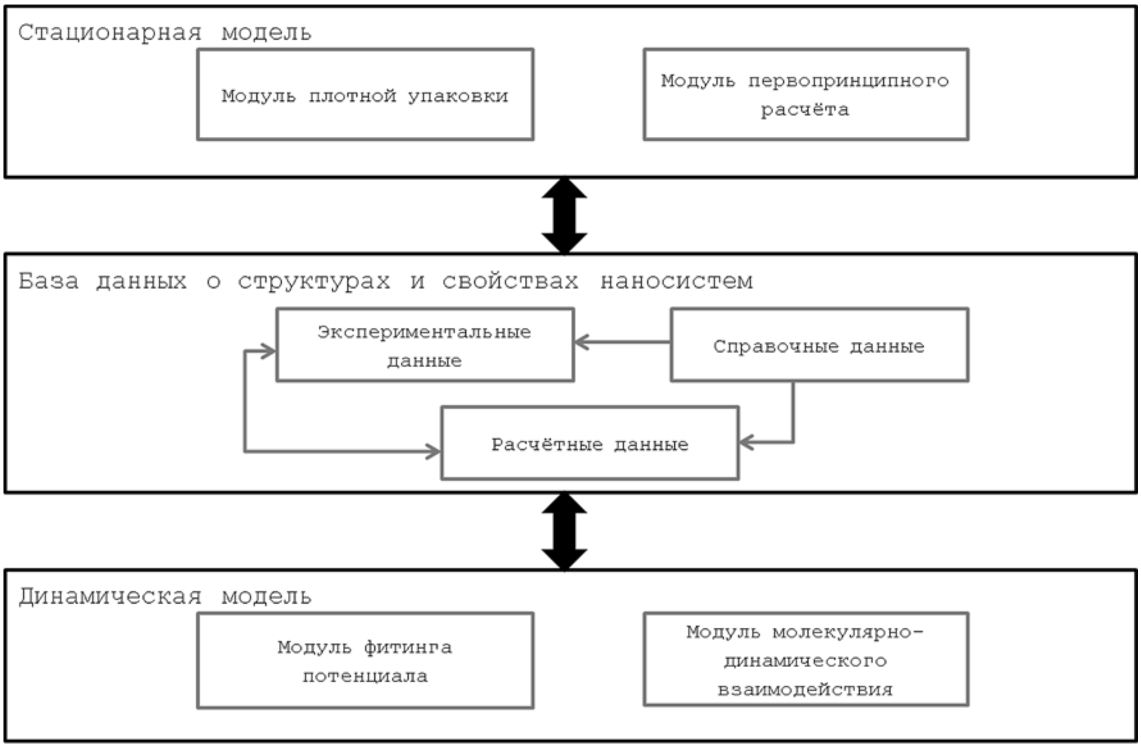


Рис. 2. Многомасштабная схема расчета МПНС

пьютерного моделирования МПНС с прогнозируемыми свойствами. Он основан на последовательном иерархическом рассмотрении и хранении данных во взаимосвязи "структура—свойство" на различных уровнях структурной иерархии. При моделировании всей МПНС необходимо воспроизвести каждый из составляющих ее основных слоев (монослой, приповерхностный и надповерхностный слои) и интерфейсные слои, находящихся на границе соседствующих основных слоев. В качестве примера приведено иерархическое представление для многослойной полупроводниковой наноструктуры GaN/AlGaN/AlN на подложке Al_2O_3 (рис. 3, см. вторую сторону обложки).

Данный подход является ресурсоемким в силу следующих причин. С одной стороны, необходимо по каждому слою МПНС хранить и извлекать информацию, полученную из различных вычислительных модулей многомасштабной схемы расчета; а также данные, полученные из экспериментов и других информационных источников. С другой стороны, для каждой МПНС необходимо хранить и извлекать информацию о ее макросвойствах (проводимости, теплопроводности и т. д.). Кроме того, вычислительные модули многомасштабной схемы расчета МПНС базируются на физических моделях, работающих на различных уровнях структурной иерархии. Так, информация, полученная на атомном уровне, используется для предсказания поведения системы в макроскопических масштабах [2]. По этой причине возникает необходимость обрабатывать (а именно — хранить, выбирать, соединять для расчета) большие объемы разнородных данных из множества источников.

Спецификой предметной области является также слабая априорная аналитическая предсказуемость моделируемого явления или процесса. Она связана с множественными вариационными расчетами, подбор оптимальных параметров в которых осуществляется на основе методов случайного поиска, а также с помощью многочисленных расчетов с фиксацией заданного набора переменных и вариацией одного или нескольких параметров. Таким образом, вычислительный эксперимент логически разбивается на последовательность шагов (с возможностью возврата) и движением по выбранному исследователем сценарию. При этом важным аспектом является существенная разнородность вычислительных модулей: большинство решаемых задач — это задачи оптимизации, в том числе задачи определения оптимальной кристаллической структуры для отдельных слоев МПНС, подбора параметров потенциалов межатомного взаимодействия при молекулярно-динамическом моделировании процессов роста МПНС и др. В связи с этим отличительной особенностью данного подхода к интеграции приложений можно считать острую необходимость композиции потоков управления, входных и выходных данных в единую объектную модель. Наконец, специфика разработки проекта предполагает независимые этапы разработки функциональных модулей от системных и информационных объектов, поэтому архитектура должна обеспечивать выполнение расчетных модулей в отдельных потоках или процессах, независимых от интеграционной оболочки.

Следует также отметить, что для всестороннего и полного исследования той или иной задачи появляется необходимость в большом количестве прецензионных и других расчетов. Это обстоятельство влечет за собой неизбежную "версионность" в вычислительном эксперименте, где основной акцент в сохранении расчетов различных версий ложится на разрабатываемую базу данных.

Постановка задачи для расчетных модулей

Модуль плотной упаковки. Данный расчетный модуль реализует модель плотной упаковки. Согласно этой модели атомы кристалла представляются твердыми шарами, взаимодействующими друг с другом, с ионными (или атомными) радиусами, зависящими от элемента, валентности и координационного числа. В основе модели лежит принцип максимального заполнения, согласно которому под действием сил притяжения атомы стремятся сблизиться друг с другом так, чтобы допустимое число кратчайших контактов было максимальным, что соответствует минимуму потенциальной энергии системы [3].

Объем одного шара с радиусом $r(A)$ элементарной ячейки

$$V = \frac{4}{3}\pi r^3(A).$$

Плотность упаковки

$$\rho = \frac{4}{3}\pi \frac{\Sigma_A \nu(A) r^3(A)}{V},$$

где V — объем элементарной ячейки; $\nu(A)$ — кратность соответствующей позиции Уайкова.

Для определения плотности упаковки необходимо определить минимальный объем V.

Кристаллические структуры, у которых плотность упаковки попадает в диапазон [0,5, 0,74], согласно принятой модели считаем устойчивыми. Если структура устойчива, запоминается соответствующая конфигурация базисных атомов элементарной ячейки. Далее определяются метрические параметры атомов элементарной ячейки для каждой устойчивой конфигурации.

Модуль первопринципного (*Ab-initio*) расчета. Расчеты свойств рассматриваемых структур проводятся в рамках положений теории функционала электронной плотности [4, 5], согласно которой:

- второй закон Ньютона $M\ddot{R} = \bar{F}(R)$, где M — масса иона, $\ddot{R}$ — ускорение, R — радиус-вектор;
- силы, действующие на ионы, $\bar{F} = -\frac{\partial E_{\text{полн}}}{\partial R}$, где $E_{\text{полн}}$ — полная энергия системы.

При расчете используется метод функционала плотности совместно с формализацией Кона-Шема [6]. В рамках этого метода задача об описании нескольких взаимодействующих электронов в статическом внешнем поле сводится к более простой задаче о независимых электронах, которые движутся в некотором эффективном потенциале, состоящем из статического потенциала атомных ядер, и учитывающем кулоновские эффекты, в частности, объемное взаимодействие и корреляцию.

Полная энергия системы вычисляется по формуле

$$E_{\text{полн}} = \sum_i \varepsilon_i - \frac{1}{2}\int \frac{n(R)n(R')}{|R-R'|} dR dR' + E_c[n] - \int \nu_c(n) n(R) dR,$$

где ε_i — энергия одной частицы; $E_c[n]$ — приближение локальной плотности; $n(R)$ и $n(R')$ — электронные плотности; R и R' — радиус-векторы; $\nu_c(n)$ — обменно-корреляционный потенциал.

Для первопринципного расчета используется программный комплекс Vienna Ab-initio Simulation Package (VASP) [4, 5].

Модуль фитинга потенциала межатомного взаимодействия (используется в молекулярно-динамических расчетах). Когезионная энергия системы атомов $E_{coh} = \frac{1}{2}\sum_{i \neq j} V_{ij}$, где $V_{ij} = f_c(r_{ij})[f_R(r_{ij}) + b_{ij} f_A(r_{ij})]$ — потенциальная энергия взаимодействия двух частиц i и j. В качестве примера рассмотрим потенциал Терсоффа, который используется при расчете свойств структур с ковалентной связью:

$$f_c(r_{ij}) = \begin{cases} 1, & r_{ij} < R_{ij} \\ \frac{1}{2}\left(1 - \sin\frac{\pi(r_{ij} - R_{ij})}{(S_{ij} - R_{ij})}\right), & R_{ij} < r_{ij} < S_{ij} \\ 0, & r_{ij} > S_{ij} \end{cases}$$ — *cutoff*-функция,

где $f_R(r_{ij})$, $f_A(r_{ij})$ — члены, отвечающие за отталкивание и притяжение; b_{ij} — многочастичный параметр взаимодействия; r_{ij} — расстояние между частицами; R_{ij} и S_{ij} — параметры, определяющиеся экспериментально из геометрических характеристик вещества.

Целевая функция записывается следующим образом:

$$F(\xi) = \omega_1\left(E_{coh}(\xi) - \hat{E}_{coh}\right)^2 + \omega_2\left(a(\xi) - \hat{a}\right)^2 + \omega_3\left(B(\xi) - \hat{B}\right)^2 + \omega_4\left(c'(\xi) - \hat{c}'\right)^2 + \omega_5\left(C_{44}(\xi) - \hat{C}_{44}\right)^2 + \omega_6\left(\zeta(\xi) - \hat{\zeta}\right)^2,$$

где a — параметр решетки; B — модуль объемной упругости; c' — модуль сдвига; C_{44} — постоянная эластичности; ζ — параметр Клейнмана; ξ — вектор идентифицируемых параметров потенциала Терсоффа; ω_j — весовые коэффициенты, j = 1, ..., 6. Значения $\hat{E}_{coh}$, $\hat{a}$, $\hat{B}$, $\hat{c}'$, $\hat{C}_{44}$, $\hat{\zeta}$ получены при помощи первопринципного расчета.

Требуется решить следующую задачу оптимизации: $F(\xi) \to \min$.

Модуль молекулярно-динамического взаимодействия. Местоположение и скорость всех дискретных частиц системы взаимодействующих объектов рассчитываются путем интегрирования системы обыкновенных дифференциальных уравнений [7]:

$$F_n = m_n \frac{dV_n}{dt}, \; V_n = \frac{dr_n}{dt},$$

где $r_n(t) = (x_n(t), y_n(t), z_n(t))$ — радиус-вектор n-й частицы системы, $V_n(t) = (u_n(t), v_n(t), w_n(t))$ — вектор скорости n-й частицы системы, $n \in [1, N]$; m_n — масса n-й частицы системы, $n \in [1, N]$; N — число частиц в системе; $F_n(t)$ — вектор силы, действующий на частицу с номером n, равный сумме сил, обусловленных взаимодействием с остальными $N - 1$ частицами. Интегрирование системы осуществляется методом скоростей Верле [8].

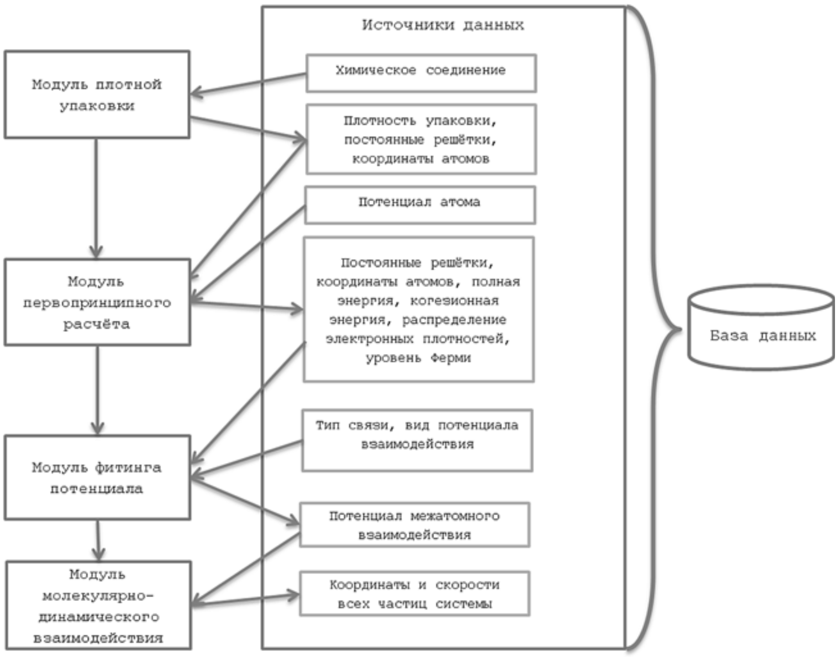


Рис. 4. Схема взаимодействия расчетных модулей

Взаимодействие расчетных модулей

В табл. 1 перечислены основные информационные потребности вычислительных модулей.

Функциональными компонентами предметной области являются рассмотренные выше вычислительные модули. К информационным компонентам относятся блоки входных и выходных данных, представленные в средней части схемы (рис. 4). Важным является то обстоятельство, что данные, полученные в результате работы вычислительного модуля, используются для последующего расчета. По этой причине возникает необходимость в хранении большого объема промежуточных результатов, которые при этом имеют разнородную структуру.

На основе изложенного выше можно сформулировать основные требования к информационной поддержке многомасштабной схемы расчета:

- для работы расчетных модулей следует хранить промежуточные результаты расчета, а также обеспечить передачу данных между модулями;
- задачи молекулярной динамики являются довольно ресурсоемкими, поэтому информационная

Таблица 1

Входные и выходные данные расчетных модулей

Модуль	Входные данные	Выходные данные	Число частиц
Плотной упаковки	Число различных атомов в химической формуле, группа симметрии (федоровская группа), позиции Уайкова, радиусы атомов	Плотность упаковки, координаты базисных атомов элементарной ячейки, соответствующие устойчивой структуре, постоянные решетки	60...100
Первопринципного расчета	Химическая формула, потенциал атомов, число атомов в элементарной ячейке, параметры решетки, уточненные координаты базисных атомов	Постоянные решетки, координаты атомов, полная энергия, когезионная энергия, распределение электронных плотностей, энергия Ферми	$10^2...10^3$
Фитинга потенциала межатомного взаимодействия	Тип связи, вид потенциала взаимодействия, полная энергия, когезионная энергия, координаты базисных атомов	Потенциал межатомного взаимодействия	10^6
Молекулярно-динамического взаимодействия	Потенциал межатомного взаимодействия	Координаты и скорости всех частиц системы	10^6

Таблица 2

Сравнительный анализ

Система или База данных	Свойства соединений		Область применения	Библиографические ссылки на источники информации	Доступ	Типы соединений
	физические	кристаллографические				
Chemical Abstracts Service (CAS)	Есть	Нет	Универсальная	Есть	Свободный	Все возможные (>74 000 000)
CSA METADEX	Есть	Нет	Производственная	Нет	Ограниченный	Только металлы и их сплавы
MIT	Ограничены	Нет	Производственная	Есть	Свободный	Только металлы, их сплавы и соли
Система, разработанная в ФТИ им. А. Ф. Иоффе РАН	Есть	Ограничены	Исследовательская	Нет	Свободный	Только полупроводники
Crystallography Open Database	Ограничены	Есть	Кристаллография	Нет	Свободный	Органические и неорганические
Inorganic Crystal Structure Data	Нет	Есть	Кристаллография	Нет (с 1913 г.)	Ограниченный	Неорганические (>117 000)
NIST Structural Database	Нет	Есть	Кристаллография	Нет	Ограниченный	Только металлы и сплавы
Ulrich's Periodical Directory	Нет	Нет	Исследовательская и критериально-поисковая	Есть	Свободный	Все возможные

поддержка должна обеспечивать хранение большого объема данных;

- необходим быстрый доступ к рассчитанным данным и наглядное представление результата расчета (визуализация).

Обзор существующих источников данных в материаловедении

Существует широкий спектр материаловедческих баз данных, однако все они ориентированы на решение тех или иных задач, обладающих определенной спецификой. В табл. 2 представлен сравнительный анализ существующих средств и источников данных предметной области.

Из краткого сравнительного анализа рассмотренных материаловедческих баз данных, обобщенного в табл. 2, можно сделать следующие выводы:

- достоинством вышеперечисленных информационных систем и баз данных является большой объем разнообразной информации о различных свойствах веществ;
- имеющиеся информационные ресурсы по своему наполнению достаточно четко поделены на системы, содержащие данные по физическим свойствам веществ, и базы данных кристаллографической направленности;
- немногие из перечисленных систем имеют библиографические ссылки на источники информации, что повышает вероятность хранения ошибочного или приблизительного, неверифицируемого контента;
- некоторые системы содержат поисковые машины по ограниченному набору критериев поиска, а также средства ограниченной визуализации, которых недостаточно для выполнения сложных и разнообразных пользовательских запросов.

По итогам краткого обзора существующих информационных систем можно сделать вывод о необходимости интеграции информационного контента кристаллических структур химических соединений и ряда физических свойств, характерных для полупроводниковых материалов. Такой подход предоставит, в свою очередь, связующие механизмы между расчетными модулями, реализующими физические модели различного масштаба и назначения.

Разработка системы информационной поддержки

На начальном этапе разработки был проведен анализ предметной области, классифицированы и структурированы соответствующие данные. В результате была спроектирована реляционная модель хранения данных [9], представленная на рис. 5 в нотации "сущность—связь" [10].

В представленной на рис. 5 структуре взаимосвязанных отношений можно выделить перечисленные далее четыре подсхемы.

• *Результаты расчетов*: каждому кристаллическому массиву, или монослою, ставятся в соответствие характеристики, полученные при помощи различных расчетных модулей, его химический состав и геометрия соединения.

• *Данные о химическом составе:* химическая формула соединения и элементы таблицы Менделеева связаны отношением "многие-ко-многим", для его реализации использована ассоциативная сущность. Химические элементы и типы радиусов также связаны соотношением "многие-ко-многим" (поскольку каждому химическому элементу может соответствовать набор радиусов для различных типов связи). Для реализации данного соотношения вводится ассоциативная сущность, которая включает в себя значения радиусов, а также ссылку на таблицу, содержащую дополнительную информацию — заряд и координационное число, соответствующие данному атому.

• *Данные о пространственной структуре соединения:* федоровская группа принадлежит к опреде-

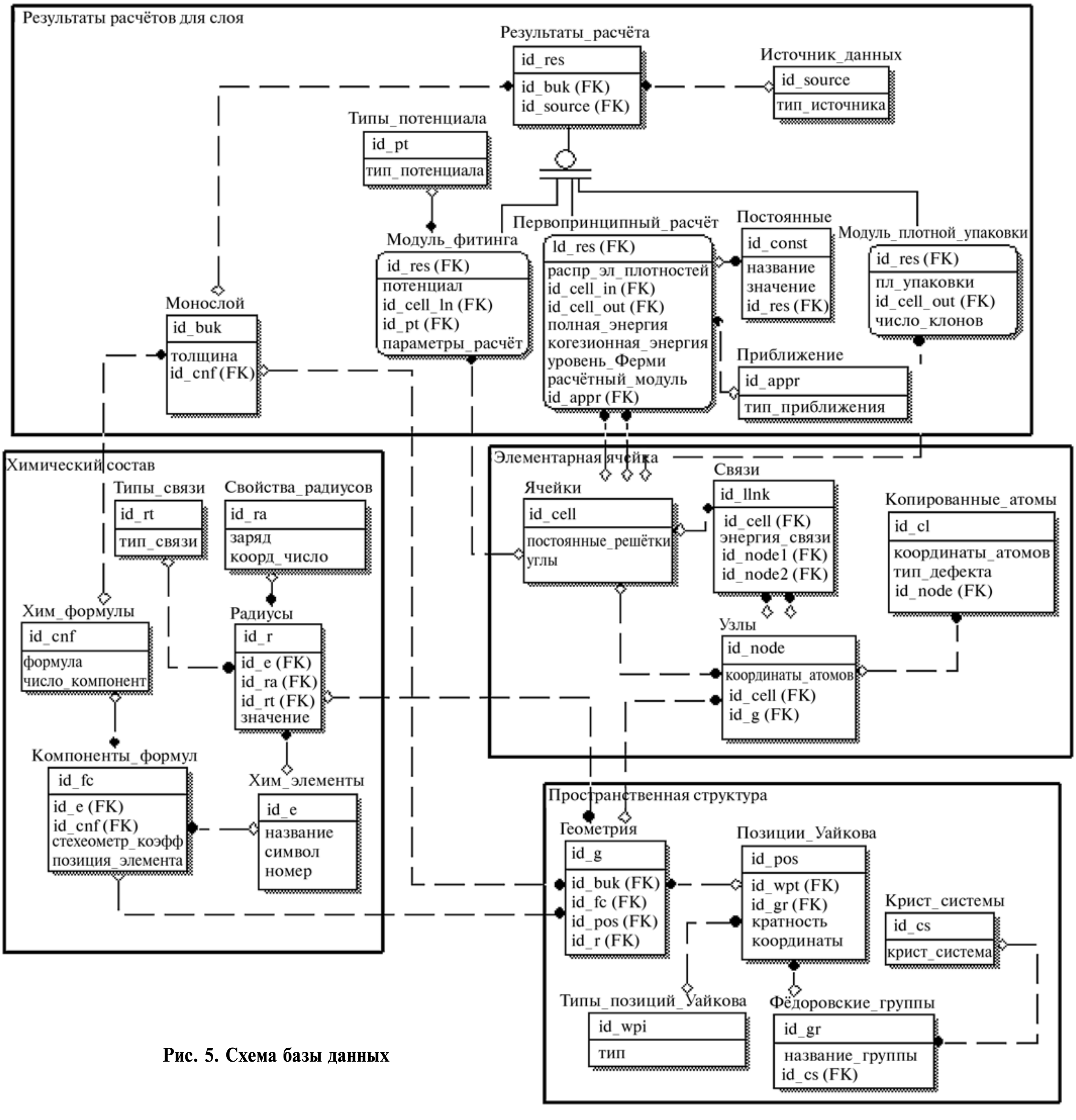


Рис. 5. Схема базы данных

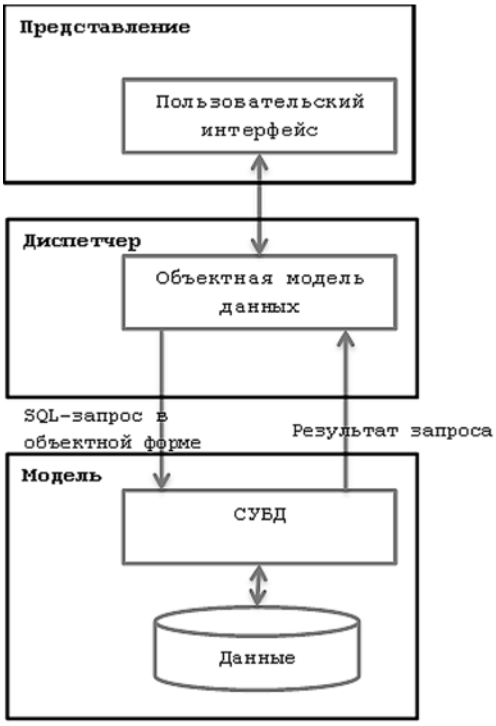


Рис. 6. Архитектура системы информационной поддержки

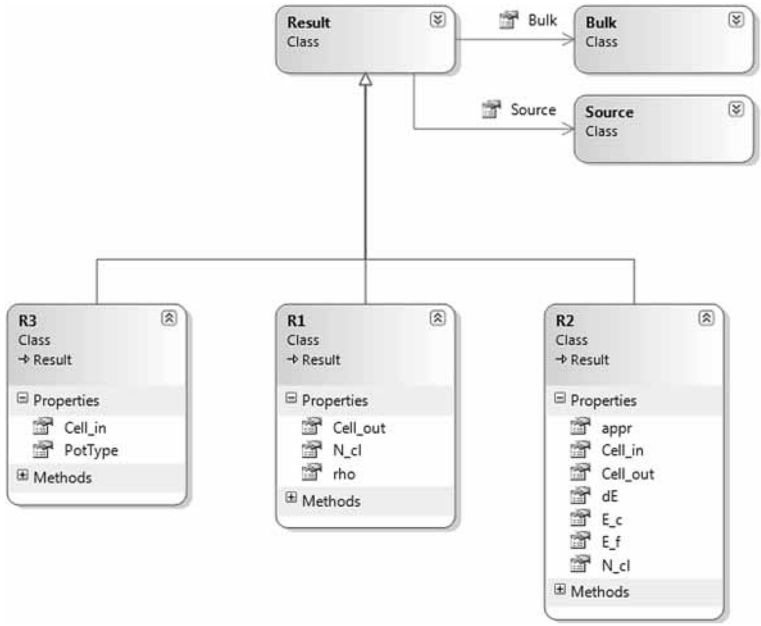


Рис. 7. Пример объектного отображения

ленной кристаллической системе. Для каждой федоровской группы определен набор позиций Уайкова, ссылающихся на нее при помощи внешнего ключа. Позиция Уайкова принадлежит к определенному типу, который, в свою очередь, определяется таблицей типов позиций Уайкова. Условие принадлежности атома какой-либо позиции Уайкова ограничивает область изменения координат атома.

- *Структурные характеристики элементарной ячейки*: элементарная ячейка представлена как собственными характеристиками, так и набором составляющих ее атомов, а также связей между ними. Атомы и связи ссылаются на содержащую их ячейку с помощью внешнего ключа. Кроме того, связь ссылается на пару атомов, которые она связывает между собой.

В ходе проектирования целевой системы разработана объектно-реляционная архитектура информационной поддержки. Ее детальное описание может быть отображено на модель MVC (*Model-View-Controller*). При этом можно выделить следующие компоненты:

- *модель (model)*, которая реализует логику управления и содержит ядро функциональных возможностей системы;
- *представление (view)*, которое отображает конечный результат;
- *диспетчер (controller)*, который получает входные и выходные данные, реализует соответствующие вызовы и управляет представлениями.

Схема взаимодействия модели, диспетчера и представления изображена на рис. 6.

Хранение данных осуществляется на сервере баз данных. В основу управления сервером положена реляционная СУБД MS SQL Server, обеспечивающая логическую целостность спроектированной схемы и возможность доступа к информации посредством SQL-запросов.

Представление данных с использованием Windows Forms отнесено на сторону клиента.

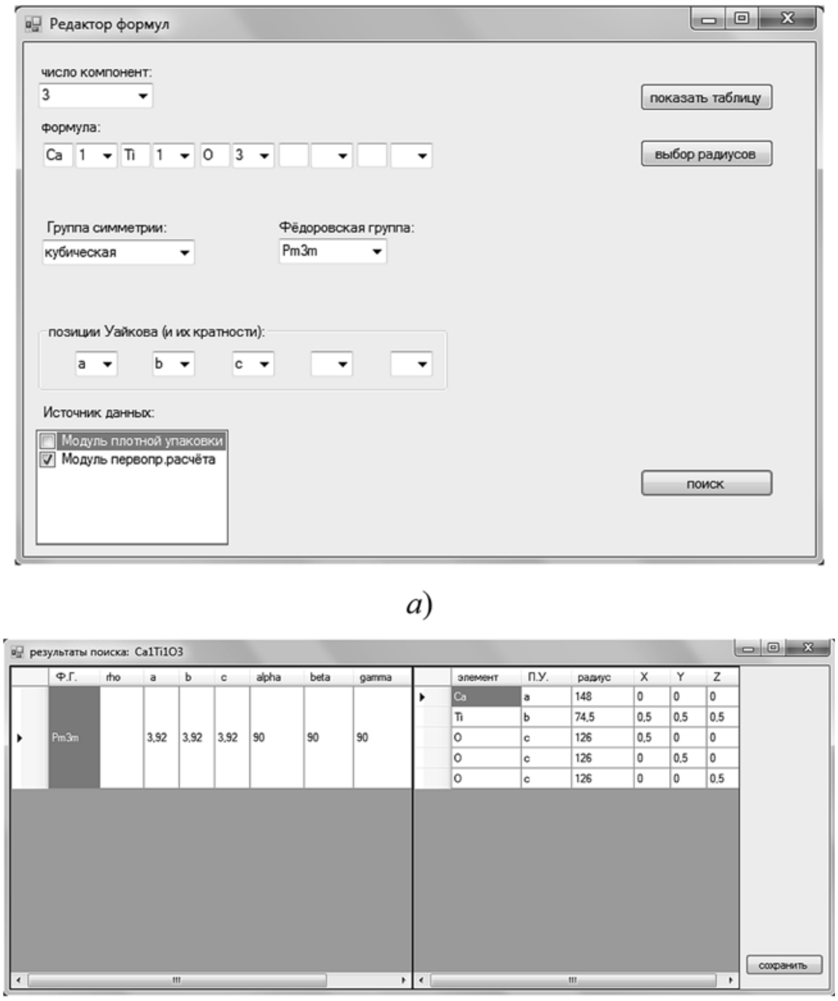
Для уменьшения зависимости архитектуры системы от конкретной СУБД была сделана попытка реализации логики работы с базой данных на основе технологии NHibernate [11, 12]. В отличие от технологий LINQ to SQL и Entity Framework, NHibernate обеспечивает более стабильную поддержку широкого круга СУБД. Существенной особенностью этой технологии является абстрактная инвариантная модель представления данных, которая может быть использована как в слое пользовательского интерфейса, так и для реализации операций манипулирования данными в СУБД. Технология NHibernate позволяет, в отличие от ADO.NET, не писать запросы к СУБД явно, а формировать их автоматически на основе декларативной информации из объектно-реляционного отображения (*mapping*). Кроме этого, технология NHibernate дает возможность кэшировать объекты и результаты выполнения запросов, существенно снижая нагрузку на сервер базы данных, в отличие от ручной реализации в ADO.NET. Пример объектного отображения для подсхемы хранения результатов расчета по монослою приведен на рис. 7.

Примеры работы

Рис. 8 и 9 иллюстрируют разработанные авторами приложения (8, *а* и 9, *а* — заполнение формы; 8, *б* и 9, *б* — результат выполнения запроса).

а) *б)*

Рис. 8. Поиск радиусов элементов соединения для заданных типов связи



а)

б)

Рис. 9. Просмотр хранимых данных для кристаллохимического соединения

Заключение

В ходе выполнения работ по созданию системы компьютерного моделирования многомасштабной схемы расчета многослойных полупроводниковых структур был проведен анализ предметной области, классифицированы и структурированы соответствующие материаловедческие и кристаллографические данные. Проанализированы физические модели, лежащие в основе вычислительных модулей многомасштабной схемы расчета, выявлены информационные потребности вычислительных модулей. Описаны функциональные требования к разрабатываемой системе информационной поддержки. На основании этих требований были построены реляционная модель хранения и правила агрегирования данных, разработана общая архитектура информационной поддержки, а также построено объектно-реляционное отображение модели хранения. Кроме того, был разработан пользовательский интерфейс, позволяющий осуществлять выборку данных согласно заданным критериям.

Разработанная система апробирована на компьютерных моделях однослойных наноструктур с кубической кристаллической решеткой — полупроводники типа $A^{III}B^{V}$ (AlN, GaN, InN), перовскиты ABC_3 ($CaTiO_3$, $BaTiO_3$).

Работа поддержана грантом РФФИ № 13-08-01335 А.

Список литературы

1. **Абгарян К. К.** Применение оптимизационных методов для моделирования многослойных полупроводниковых наносистем // Тр. Института системного анализа Российской академии наук. Динамика неоднородных систем. 2010. Т. 53, № 3. С. 6—9.
2. **Трушин О. С., Викулов П. А., Карим А., Кара А., Рахман Т.** Исследование диффузионных процессов на поверхности металлов методом самообучаемого кинетического Монте-Карло // Математическое моделирование. 2007. Т. 19, № 3. С. 116—126.
3. **Абгарян К. К., Хачатуров В. Р.** Компьютерное моделирование устойчивых структур кристаллических материалов // Вычислительная математика и математическая физика. 2009. Т. 49. № 8. С. 1517—1530.
4. **Kohn W.** Nobel Lecture: Electronic structure of matter — wave functions and density functional // Rev. Mod. Phys. 1999. Vol. 71, N 5. P. 1253—1266.
5. **Kresse G., Furthmüller J.** Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set // Computational Materials Science. 1996, Vol. 6, Issue 1. P. 15—50.
6. **Kresse G., Furthmüller J.** Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set // Physical Review B. 1996. Vol. 54, Issue 16. P. 11169.
7. **Securityзняк В. Е.** Основы вычислительной физики. Москва-Ижевск: НИЦ Регулярная и хаотическая динамика, 2006. 156 с.
8. **Verlet L.** Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules // Phys. Rev. 1967. Vol. 159, N 1. P. 98—103.
9. **Дейт К. Дж.** Введение в системы баз данных, 8-е изд.: Пер. с англ. М.: Вильямс, 2005. 1024 с.
10. **Чен П. П.-Ш.** Модель "сущность—связь" — шаг к единому представлению о данных // Системы Управления Базами Данных. 1995. № 3. URL: http://citforum.ru/database/classics/chen/
11. **Schenker G. N., Cure A.** NHibernate 3 Beginner's Guide. Packt Publishing, 2011, 368 c.
12. **Dentler J.** NHibernate 3.0 Cookbook. Packt Publishing, 2010, 328 c.

K. K. Abgarian, Associate Professor, Head of Department, e-mail: kristal83@mail.ru, **P. A. Sechenykh**, Postgraduate Student, Junior Researcher, e-mail: p-sechenyh@mail.ru, **E. S. Gavrilov**, Senior Teacher, Junior Researcher, e-mail: eugavrilov@gmail.com, Moscow Aviation Institute (National Research University), Institution of Russian Academy of Sciences Dorodnicyn Computing Centre of RAS

The Object-Relational Approach to the Development of a System of Computer Simulation of Multiscale Computational Scheme of Multilayer Semiconductor Nanostructures

The article examines the object-relational system of information support of multiscale computational scheme of the calculation of the multilayer semiconductor nanostructures' (MSNS) properties. Units and devices based on MSNS including layers of AlN, GaN, InN are used in various fields of science and technology: nanoelectronics industry, radio electronics, lasers, LEDs, computer technology. For example, units that use MSNS are widely applied in aviation and space radio devices.

The article provides a description of some mathematical models implemented in computing modules of the previously developed MSNS computational scheme. The paper also gives a comparison of databases and information systems used at problem solving of materials science.

The paper presents data storage relational model which applies to solving resource-intensive and different-scale problems. Created object mapping of data storage relational model. The implementation of the object mapping was based on the technology of NHibernate, with the use of the library FluentNHibernate. Created architecture of the scheduler, which is based on the description of operations with data and specifications of information flows enables the exchange of data between computing modules.

The paper shows the implementation of user interface, which allow criteria-based data retrieving and generate a text file that contains the input data for calculation modules, and save it for later import the calculating properties sequence. To implement the user interface was used technology Windows Forms.

The developed system has been tested on computer models of multilayered semiconductors that actively used in the aerospace industry.

Keywords: *database, object-relational approach, multilayer semiconductor nanostructure, multiscale calculation scheme*

References

1. **Abgarian K. K.** Primenenie optimizatsionnykh metodov dlya modelirovaniya mnogosloinykh poluprovodnikovykh nanosistem *Trudy Instituta sistemnogo analiza Rossijskoj akademii nauk. Dinamika neodnorodnyh system,* 2010, vol. 53, no. 3, pp. 6—9 (in Russian).
2. **Trushin O. S., Vikulov P. A., Karim A., Kara A., Rahman T.** Issledovanie diffuzionnykh protsessov na poverkhnosti metallov metodom samoobuchaemogo kineticheskogo Monte-Karlo. *Matematicheskoe modelirovanie*, 2007, vol. 19, no. 3, pp. 116—126 (in Russian).
3. **Abgarian K. K., Hachaturov V. R.** Komp'yuternoe modelirovanie ustoichivykh struktur kristallicheskikh materialov. *Vychislitel'naya matematika i matematicheskaya fizika*, 2009, vol. 49, no. 8, pp. 1517—1530 (in Russian).
4. **Kohn W.** Nobel Lecture: Electronic structure of matter — wave functions and density functionals, *Rev. Mod. Phys.*, 1999, vol. 71, no. 5, pp. 1253—1266.
5. **Kresse G., Furthmuller J.** Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. *Computational Materials Science*, 1996, vol. 6, issue 1, pp. 15—50.
6. **Kresse G., Furthmüller J.** Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Physical Review B*, 1996, vol. 54, issue 16. pp. 11169.
7. **Zaliznjak V. E.** *Osnovy vychislitel'noj fiziki* (Fundamentals of computing physics), Moskva-Izhevsk, NIC Reguljarnaja i haoticheskaja dinamika, 2006, 156 p. (in Russian).
8. **Verlet L.** Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. *Phys. Rev.* 1967, vol. 159, no. 1, pp. 98—103.
9. **Date C. J.** *An Introduction to Database Systems*, Addison-Wesley, 2004, 1024 p.
10. **Chen P. P.-S**. The Entity-Relationship Model — Toward a Unified View of Data, *ACM Transactions on Database Systems (TODS)*, 1995, vol. 1, pp. 9—36.
11. **Schenker G., Cure A.** *NHibernate 3 Beginner's Guide*, Packt Publishing, 2011, 368 p.
12. **Dentler J.** *NHibernate 3.0 Cookbook*, Packt Publishing, 2010, 328 p.

УДК 621.391

С. А. Благонравов, ст. инженер, **С. Б. Уткин,** начальник НИС 22-03, гл. специалист, Санкт-Петербургское ОКБ "Электроавтоматика" им. П. А. Ефимова,
С. В. Батова, вед. инженер, **П. В. Коновалов,** аспирант, e-mail: olkesomewhere@gmail.com, Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Опыт применения технологии эмуляции процессов при разработке компонентов программного обеспечения авиационных систем

Представлены результаты разработки программного эмулятора аппаратной платформы вычислительной системы, который предназначен для автоматизированной поддержки процесса разработки программного обеспечения. Такой эмулятор позволяет выполнять разработку компонентов программного обеспечения на ранних этапах его проектирования. Разработка целевых программных компонентов осуществляется на инструментальной ЭВМ, а проверка корректности работы программного кода осуществляется на эмуляторе, который сам реализуется на инструментальной ЭВМ. Предложены принципы построения эмулятора. Описаны его функциональные возможности и доступные разработчику программные средства эмулятора. Приведены пример функциональной схемы вычислительной системы и пример рабочего окна эмулятора в режиме отладки программного кода. Описан алгоритм тестирования операционной системы на эмуляторе.

Ключевые слова: *эмуляция, авионика, операционные системы реального времени, интегрированная модульная авионика, ARINC 653*

Введение

Технология эмуляции вычислительных процессов и процессов обмена, протекающих в информационно-управляющих системах подвижных объектов, в настоящее время широко используется при разработке большинства программно-управляемых изделий. Значительные успехи достигнуты исследователями в создании средств эмуляции для отработки аппаратных решений [7]. Наглядными примерами являются среды проектирования и программирования аппаратных средств Xilinx Foundation, Max Plus, LabVIEW, MicroCap и др.

Технология эмуляции позволяет анализировать поведение вычислительного узла (модуля авионики) на предварительных этапах проектирования (эскизно-техническое проектирование, техническое предложение), когда в распоряжении разработчика отсутствует готовое физическое устройство и необходимо оценить целесообразность принятия и последующего внедрения в производство какого-либо проектного решения [3, 8].

Применение технологии эмуляции при разработке компонентов программного обеспечения на языках высокого уровня к настоящему времени находится в зачаточном состоянии и в значительной степени определяется возможностями используемой разработчиками среды проектирования программ. Вместе с тем технология эмуляции функционального поведения вычислительной платформы позволит различным группам разработчиков программного обеспечения (программистам, интеграторам проекта, тест-инженерам и пр.) работать одновременно, уменьшая тем самым суммарное время разработки изделия и способствуя принятию правильного проектного решения. Особенно актуально применение технологии эмуляции при разработке программного обеспечения для систем, к функционированию которых предъявляются повышенные требования по безопасности целевого применения, в частности, к авиационным вычислительным системам и комплексам [1, 6].

Необходимость разработки технологии эмуляции при проектировании комплексов программ на высокоуровневых языках определяется несколькими условиями.

Во-первых, аппаратное обеспечение вычислительной системы авионики на начальных этапах проектирования либо отсутствует (еще не изготовлено), либо изготовлено в единичном (опытном) экземпляре и используется различными группами разработчиков, находящимися в состоянии "взаимной конкуренции" за обладание временным ресурсом использования образца.

Во-вторых, возможности современных отладочных средств, используемых при тестировании аппаратно-программного обеспечения сложных программно-управляемых систем, ограничены, особенно если отладке подлежит низкоуровневое системное программное обеспечение.

В-третьих, проверить выполнение некоторых ветвей программного кода на реальной аппаратной платформе в ряде случаев оказывается проблематично, а иногда и вовсе невозможно. Примером такого случая является фрагмент программного кода, реализующий реакцию вычислительной системы на различные отказы узлов (модулей) аппаратуры, в том числе и отказ самого модуля-вычислителя в составе вычислительной системы или комплекса.

Наконец, исполнение на физической аппаратуре большого объема тест-программ, необходимых для прохождения этапа сертификации программного обеспечения авионики, даже при использовании средств автоматизации, является достаточно трудоемкой процедурой, а в процессе разработки комплексов программ и их сертификации такие эксперименты выполняются многократно.

В связи с этим актуальной является задача разработки технологии эмуляции, ориентированной на исследование проектных решений в рамках создания компонентов программного обеспечения вычислительных систем авиационного применения.

Обоснование разработки программы-эмулятора

Когда в начале 2000-х гг. в Санкт-Петербургском ОКБ "Электроавтоматика" было принято решение о начале разработки новой отечественной операционной системы (ОС) реального времени для изделий интегрированной модульной авионики, соответствующей отраслевому стандарту ARINC 653, разработчики программного обеспечения ощутили острую нехватку готовых аппаратных платформ и отсутствие возможности апробации проектных решений. Коллектив разработчиков располагал единственным экземпляром целевой платформы, для которой разрабатывалась новая ОС, а необходимость написания большого объема низкоуровневого кода, работающего со сложной высокоинтегрированной аппаратурой, при отсутствии аппаратных средств для отладки программного обеспечения (ПО) на целевой платформе предвещала значительные временные затраты на его отладку.

Классическим решением в данной ситуации является использование для разработки ПО эмулятора целевой аппаратной платформы. Однако доступные на тот момент бесплатно распространяемые эмуляторы (коммерческие решения не рассматривались ввиду ограниченного финансирования) имели существенные недостатки [9, 10]. К их числу относятся следующие: во-первых, отсутствие эмуляции уникальных устройств (контроллеров) собственной разработки, входящих в состав заданной целевой платформы; во-вторых, отсутствие удобного графического интерфейса пользователя; в-третьих, отсутствие развитых интеллектуальных средств отладки ПО. Разработчики ОС осознали потребность не просто заменить аппаратный "черный ящик" целевой платформы на программный "черный ящик" эмулируемой целевой платформы. Они вознамерились превратить его в "белый ящик", т. е. получить полную информацию о состоянии системы и ее компонентов, причем в удобном экранном представлении. Таким образом, разработчики ОС пришли к выводу о необходимости разработать специализированную программу-эмулятор целевой аппаратной платформы для отладки и тестирования ОС.

Целевая аппаратная платформа представляла собой базовый вычислительный модуль авионики — процессорную плату, реализованную на основе микропроцессора IDT 79RC64574 архитектуры MIPS4. Функциональная схема базового вычислительного модуля авионики приведена на рис. 1. Состав аппаратных средств модуля:

- цифровой процессор ЦП 1;
- цифровой процессор ЦП 2;
- контроллер-коммутатор памяти, реализованный на базе элементов программируемой логики;
- контроллер RS-232;
- системный таймер;
- аппаратные средства взаимодействия между процессорами (система "почтовых ящиков", обеспечивающая обмен сообщениями между процессорами), реализованные на базе элементов программируемой логики;
- постоянное запоминающее устройство (ПЗУ) общего доступа, в качестве которого использована FLASH-память;
- несколько оперативных запоминающих устройств (ОЗУ), объединенных по схеме независимых "банков ОЗУ";
- ОЗУ общего доступа (ОЗУОД);
- контроллер PCI для процессора ЦП 2, подключенный по системному параллельному интерфейсу CPCI, необходимый для информационного обмена вычислительного модуля в составе многомодульной вычислительной системы;
- контроллер PCI, необходимый для подключения двух плат-мезонинов к вычислительному модулю по системному параллельному интерфейсу PMC;
- управляющий микроконтроллер (МК), имеющий доступ к ПЗУ процессоров через свою внешнюю шину;
- ОЗУ общего доступа ЦП 1 и МК;
- ОЗУ общего доступа ЦП 2 и МК.

Эмулятор должен был представлять собой совокупность программных средств, исполняемых на инструментальной ЭВМ в диалоговом режиме. В задачи эмулятора вычислительного модуля входили:

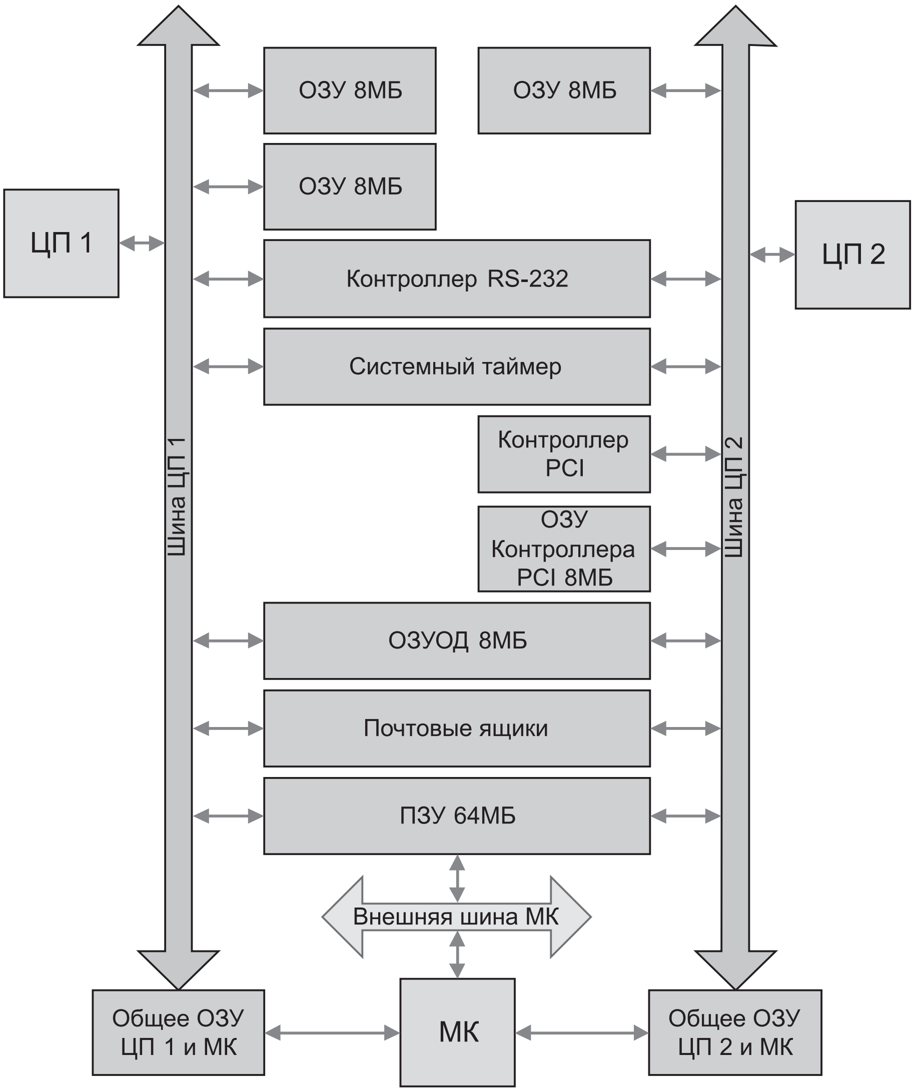


Рис. 1. Функциональная схема базового вычислительного модуля

- эмуляция работы таких аппаратных устройств, как процессоры, шины адрес/данные, ОЗУ (в том числе режим общего доступа для процессоров), ПЗУ, контроллеры RS-232, системные таймеры, аппаратные средства взаимодействия между процессорами;
- эмуляция выполнения ПО на двух процессорах одновременно;
- возможность задания различных режимов работы модуля, независимое управление пуском, остановкой обоих процессоров;
- загрузка файлов в ячейки памяти ОЗУ/ПЗУ эмулируемого модуля и чтения ячеек памяти ОЗУ/ПЗУ с сохранением данных в файле;
- реализация окна консольного терминала интерфейса RS-232;
- дизассемблер содержимого ячеек памяти;
- анализ отладочной информации в загружаемом (выполняемом) файле формата ELF-32 с возможностью отображения на экране инструментальной ЭВМ исходного программного кода и соответствующего ему дизассемблированного, выполняемого программного кода;
- обеспечение режима отладки программ, включающего возможности пуска, паузы, точки останова, пошагового выполнения инструкций процессора, пошагового выполнения строк исходного программного кода с заходом или без захода внутрь вызываемой подпрограммы;
- отображение полного состояния всех устройств модуля и возможность изменения состояния любого устройства во время отладки программного кода;
- возможность установки пользователем точек остановки по событиям, возникающим в процессе работы эмулируемой аппаратуры (исключение, переполнение, изменение, обращение по адресу, обращение к ОЗУ и пр.);
- возможность отслеживания данных в памяти в соответствии с их типами (низкоуровневая отладка), возможность устанавливать точки остановка при изменении данных в памяти;
- обеспечение возможности отладки программного кода при его выполнении с использованием механизма "виртуальной памяти";
- сохранение истории выполнения программы в различных режимах (последовательность вызова подпрограмм, последовательность выполняемых процессором инструкций в пределах выбранной функции или программы в целом);
- сохранение полного состояния конфигурации модуля в "точках отката" с возможностью последующего возобновления выполнения операций с сохраненного места;
- поддержка режимов работы, для которых эмулируемое время выполнения инструкций достаточно близко к реальному времени выполнения на физической аппаратуре, чтобы временная диаграмма работы модуля на эмуляторе соответствовала временной диаграмме работы реальной аппаратуры.

Сравнение программы-эмулятора с существующими аналогами

Эмуляция чаще всего полностью имитирует аппаратную среду. Исходя из перечисленных выше задач разрабатываемого эмулятора были рассмотрены в качестве примеров коммерческий эмулятор фирмы

Сравнение характеристик эмуляторов

Характеристика	ОС ЭОС	QEMU	Wind River
Эмулируемая платформа	Аппаратная: бортовой вычислительный модуль МВ70 (включая уникальные контроллеры собственной разработки) и различные его модификации	Аппаратная: различные виды процессоров и периферия персонального компьютера	Программная: выполнение приложений ARINC 653 под управлением ОС VxWorks
Поддержка отладки приложений	Приложения, соответствующие стандарту ARINC 653 под управлением ОС ЭОС	Нет	Приложения, соответствующие стандарту ARINC 653 под управлением ОС VxWorks
Поддержка разработки ОС	Да (ОС ЭОС)	Нет	Нет
Инструментальная платформа ОС	Windows	Linux	Windows, Linux
Установка	Копирование исполняемого файла на компьютер	Установка и настройка из командной строки	JTAG-эмуляторы Wind River ICE/Wind River Probe, среда разработки и отладки Wind River Workbench и отладочные платы Wind River
Графический интерфейс	Есть	Нет	Есть

Wind River и свободно распространяемый эмулятор QEMU [9, 10]. В таблице представлено сравнение характеристик указанных эмуляторов и эмулятора ОС ЭОС (ЭОС — операционная система, разрабатываемая и поддерживаемая авторами).

При разработке эмулятора были учтены все положительные и отрицательные стороны известных эмуляторов, а также особенности разрабатываемого продукта прикладного характера. Из таблицы видно, что эмулятор существенно упрощает установку продукта и дает возможность поддерживать разработку специфичной ОС. На рис. 2 и 3 (см. третью сторону обложки) приведены скриншоты, дающие представление о графическом интерфейсе пользователя, реализованном эмулятором Wind River Workbench и эмулятором собственной разработки соответственно.

Совместная (взаимосвязанная) разработка ОС и эмулятора

Разработка эмулятора предполагала, во-первых, создание программы, эмулирующей работу функциональной модели вычислительного модуля, во-вторых, дополнение эмулятора функциональными возможностями для отладки ОС ЭОС [4, 5]. Таким образом, на начальном этапе была создана модель эмулятора процессорного модуля, соответствующая функциональной модели вычислительного модуля (см. рис. 1). Назначение программной реализации такой модели — выполнять код разрабатываемой программы и в окне терминала инструментальной ЭВМ [2] выводить программные сообщения, направляемые на консоль. На этом этапе были разработаны несколько отдельных тестовых программ, выполнение которых на эмуляторе и на аппаратуре показало полную идентичность исполнения инструкций и приемлемую близость рассчитываемого на эмуляторе времени выполнения программы к реальному времени выполнения той же программы на физической аппаратуре. Таким образом была подтверждена правильность предложенных разработчиками моделей.

Далее расширение функциональных возможностей эмулятора велось одновременно с разработкой ОС. Код ОС отлаживался на эмуляторе. Одновременно выполнялось тестирование кода самого эмулятора. Аналогичный подход часто используется при разработке компиляторов языков программирования, когда предыдущая версия компилятора используется для компиляции кода следующей версии самого компилятора.

В результате экспериментов за довольно короткий срок (два года) разработчики ОС смогли: реализовать и исследовать несколько вариантов архитектуры ядра ОС (из которых затем была выбрана наилучшая); реализовать интерфейс ОС и приложений, соответствующий стандарту ARINC 653; сформировать и выполнить набор тестов для проверки соответствия ОС этому стандарту — суммарно около 250 тестов. Необходимо отметить, что весь объем работ проводился на эмуляторе практически без использования целевой аппаратной платформы.

Для проверки корректности самого разработанного набора тестов все тесты проверялись на ОС VxWorks (распространенной ОС реального времени, соответствующей стандарту ARINC 653, используемой в качестве "эталонной"). При этом важно отметить, что тестовые испытания также проводились не на реальной целевой платформе, а на эмуляторе этой ОС Wind River Workbench, выполнение которого осуществляется на персональном компьютере. Таким образом, ОС была практически полностью разработана и отлажена в виртуальной аппаратной среде — на эмуляторе. Только на последнем этапе разработки было проведено полное тестирование на целевой аппаратной платформе, выявившее дополнительные ошибки, не обнаруженные при тестировании на эмуляторе и составившие около 5 % от общего числа выявленных ошибок.

Затем значительно расширился набор тестов для ОС, необходимый для выполнения ее сертификации в соответствии с нормативными документами, принятыми в авиационной отрасли. Были разработаны средства автоматического исполнения пакета тестов, а также средства автоматизации анализа результатов тестов. Это позволило существенно сократить сроки проведения тестовых испытаний. Например, прогон набора из 1500 тестов, каждый из которых представляет собой отдельную конфигурацию ПО, загружаемого в целевую платформу, может быть выполнен в автоматическом режиме за несколько часов. Полный прогон тестов и анализ их результатов с использованием специальных средств автоматизации может быть выполнен одним работником за один рабочий день.

При создании тестов разработчики ОС столкнулись с неожиданным препятствием. В отличие от прикладного ПО авионики, которое разрабатывается для конкретного изделия, операционная система, являясь универсальным ПО, должна выполняться на разных аппаратных платформах и обеспечивать работу с разными конфигурациями прикладного ПО и аппаратных средств. Оказалось, что протестировать работу ОС с конфигурациями, в которых некоторые параметры имеют максимальные проектные значения, на физических аппаратных платформах невозможно в принципе в силу недостатка аппаратных ресурсов. Например, вследствие недостатка физической памяти нельзя провести тестирование конфигураций, в которых число объектов некоторого типа равно максимальному значению, заданному в требованиях к ОС. Решение лежало на поверхности и заняло несколько минут — разработчики эмулятора "добавили" в эмулируемый модуль несколько сотен Мбайт оперативной памяти.

Одновременно появилась необходимость портирования ОС на другую целевую платформу. Поскольку процессорные ядра обоих платформ реализуют одну архитектуру MIPS (хотя и разные ее версии) решено было обойтись "малой кровью" и не разрабатывать модель для нового процессора, а разработать модели внешних устройств новой целевой платформы. В результате появилась эмуляция нескольких гибридных модулей, не существующих в природе, каждый из которых позволял отрабатывать отдельные компоненты ПО, предъявляющие специфические требования к аппаратной платформе. Интересной особенностью портирования ОС с одной целевой платформы на другую, ощущаемой программистом, явилась плавность перехода. Она выражается в постепенном замещении устройств одной платформы другими, что позволило выполнить переход поэтапно и с гораздо большим комфортом, чем при выполнении портирования без использования эмулятора. Например, в исходную эмулируемую целевую платформу была добавлена эмуляция контроллера RS-232, входящего в состав новой целевой платформы. Разработчик драйвера нового контроллера получил возможность отлаживать новый программный код, пользуясь старым контроллером для выдачи диагностических сообщений на терминал. Когда отладка драйвера для нового контроллера была завершена, старый контроллер был удален из эмулируемой платформы.

Примерно в это же время функциональные возможности эмулятора были расширены механизмами просмотра внутренних структур данных ОС. Если возможность просмотра переменных отлаживаемой программы, в качестве которой в данном случае выступает ОС, была и ранее, то теперь в эмуляторе появилось средство удобного просмотра сложных структур данных, формируемых ОС динамически на этапе инициализации. Примером такой информации являются структуры данных, используемые ОС для организации виртуальных адресных пространств приложений и управления аппаратными устройствами отображения виртуальной памяти (TLB). Такое расширение функциональных возможностей привело к значительному сокращению времени отладки системного ПО, поскольку устранило необходимость для программиста "вручную" проходить по цепочке указателей, отыскивая нужный объект в памяти ОС.

К моменту окончания разработки первой рабочей версии ОС сложилась определенная технология работы. Весь коллектив программистов, как разработчики самой ОС, так и разработчики тестов для ОС, выполняет написанные программы только в виртуальной аппаратной среде на эмуляторе. Причем эмулируемые целевые платформы у разных разработчиков могут отличаться в зависимости от конкретных задач. Каждый этап разработки ОС завершается регрессионным тестированием — полным прогоном всего набора тестов на эмуляторе и устранением найденных ошибок. Окончательная версия ОС собирается для заданной целевой платформы и тот же полный набор тестов прогоняется на целевых аппаратных средствах. При этом применяется инструментальное ПО, выполняющее пакетный прогон тестов на целевой платформе и использующее те же конфигурационные файлы, что и ПО, выполняющее прогон на эмуляторе. Для анализа протоколов выполнения тестов применяется общее инструментальное ПО. Наличие у каждого программиста эмулятора позволяет распараллелить разработку и отладку отдельных компонентов ОС и тестов. Каждый программист, изменяющий код ОС, имеет возможность самостоятельно выполнить соответствующие тесты и отладить написанный код, а каждый разработчик теста имеет возможность самостоятельно выполнить и отладить написанный или измененный тест еще до финального прогона всего пакета тестов, благодаря чему финальное тестирование обычно не выявляет никаких дополнительных ошибок и служит лишь для получения формального сертификационного зачета. Процесс проведения формального тестирования ОС с использованием эмулятора показан на рис. 4.

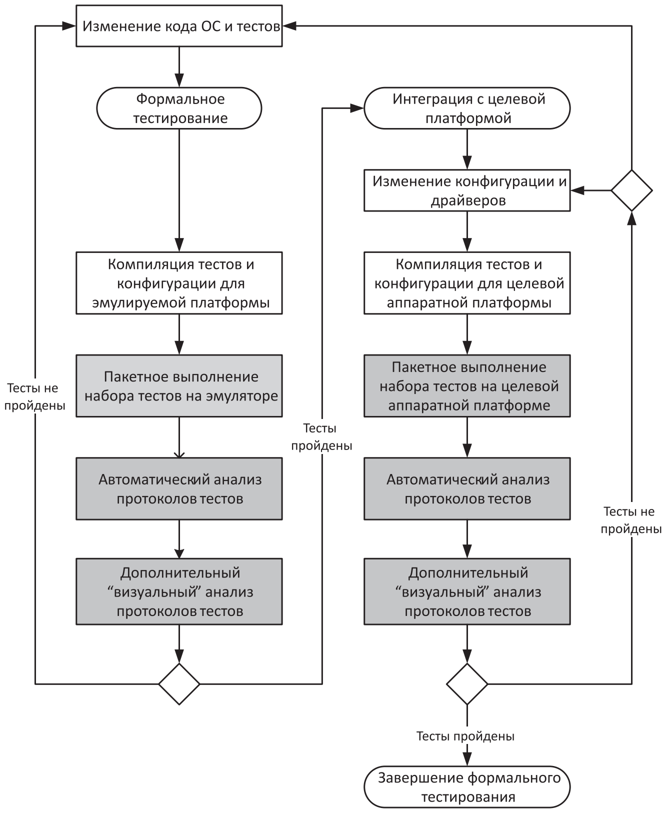


Рис. 4. Технология разработки ОС с использованием программы-эмулятора

Архитектура эмулятора

Рамки данной статьи не позволяют подробно остановиться на описании принципов и внутренней архитектуры эмулятора. Поэтому далее кратко опишем только архитектуру эмулятора с точки зрения реализации заданных функций. На рис. 5 приведена упрощенная диаграмма классов эмулятора, отражающая связь архитектуры эмулятора с реализуемыми функциями.

Базовым классом для всех эмулируемых устройств является класс Device, реализующий интерфейсы, необходимые как собственно для эмуляции выполнения программ, так и для обеспечения выполнения сервисных функций по отладке ПО. Все потомки класса Device подразделяются на три основные группы. Классы, эмулирующие устройства, выполняющие программы, реализуют дополнительный интерфейс ICPU. Классы, эмулирующие функциональные возможности шины данных, реализуют дополнительный интерфейс IBus. Большую часть в иерархии классов, эмулирующих аппаратуру целевой платформы, составляют потомки класса BusDevice, эмулирующие устройства, подключаемые к шине данных. Для этого класс BusDevice реализует дополнительный интерфейс IBusDevice. Потомками этого класса являются классы, эмулирующие память, системный таймер, контроллер RS-232 и др.

Класс Board описывает вычислительный модуль целевой платформы, содержащий заданный набор устройств, реализуемых классами, порождаемыми от класса Device. Класс Simulator отвечает за про-

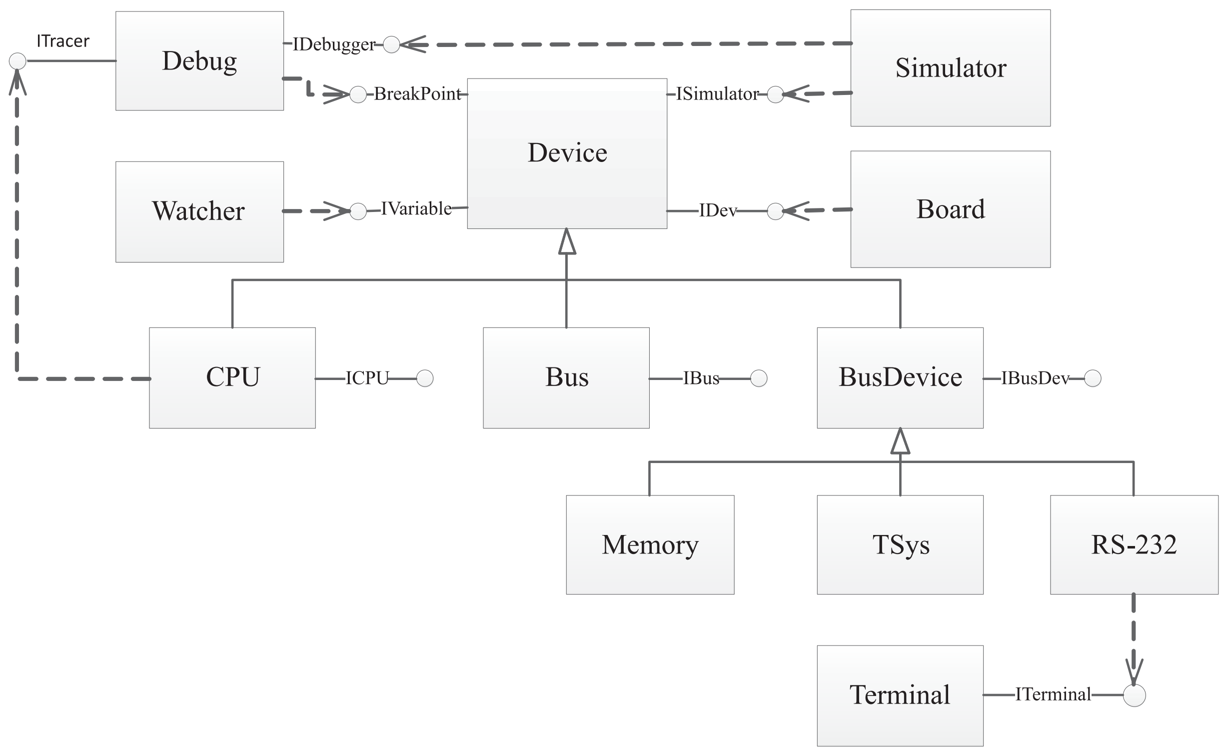


Рис. 5. Диаграмма классов эмулятора

цесс эмуляции выполнения программы, включая сброс и останов процессоров, анализ активности установленных событий отладки и вычисление времени выполнения программы. Класс Debug реализует механизмы трассировки выполнения программ и работы с событиями отладки. Класс Watcher обеспечивает возможность просмотра состояния всех устройств, в том числе предоставляя внутреннюю структуру этих устройств в виде набора переменных различных типов. Наконец, класс Terminal обеспечивает протоколирование и подыгрыш взаимодействия с целевой платформой по интерфейсу RS-232.

Перспективы применения эмулятора для отладки функционального ПО

Когда в ОКБ "Электроавтоматика" началась разработка функционального ПО для бортовых систем, построенных по принципам интегрированной модульной авионики и использующих новую ОС, возникло естественное желание предоставить разработчикам прикладного ПО такую же комфортную среду, какой пользовались разработчики самой ОС. Очевидно, что использование эмулятора целевой платформы при разработке функционального ПО может обеспечить те же преимущества, что и при разработке системного ПО. К числу таких преимуществ относятся перечисленные далее.

- Автономность разработчиков, т. е. возможность параллельной разработки и отладки ПО коллективом программистов в чисто виртуальной среде без использования целевой платформы.
- Опережающая разработка ПО, т. е. разработка до появления целевой платформы и даже до выбора конечной конфигурации целевой платформы. По результатам такой опережающей разработки можно сделать оценку необходимых аппаратных ресурсов (числа процессоров, объемов памяти) и сформировать окончательные требования к составу целевой аппаратной платформы.
- Комфортная высокоуровневая отладка приложений с возможностью сохранения "следов" деятельности и функцией повторения выполненных действий.
- Автоматизация тестирования.

Вместе с тем использование эмулятора в качестве инструмента для разработки и отладки функционального ПО связано с необходимостью решения ряда дополнительных задач. Эти задачи обусловлены, во-первых, значительным возрастанием сложности отлаживаемых программных систем, особенно на этапе интеграции компонентов прикладного ПО, во-вторых, большим разнообразием аппаратных средств, для которых разрабатывается прикладное ПО. В результате анализа подходов к решению этих задач разработчики эмулятора пришли к выводу о необходимости дальнейшего расширения функциональных возможностей эмулятора по следующим направлениям:

- ♦ развитие возможностей отладки отдельных приложений, выполняемых под управлением ОС, просмотра состояния объектов, созданных приложением;
- ♦ моделирование внешних связей эмулируемой целевой платформы, включающее создание моделей каналов ввода—вывода, используемых в авионике;
- ♦ реализация возможности гибкого изменения конфигурации эмулируемой платформы самим пользователем, обеспечивающего выбор эмулируемых аппаратных средств;
- ♦ превращение эмулятора в открытую систему с возможностью наращивания функций, добавления эмулируемых устройств и подыгрывающих моделей;
- ♦ дополнение эмулятора средствами динамического анализа выполняемого кода, обеспечивающими анализ глубины вызовов, использования стека, анализ структурного покрытия кода, анализ покрытия условий/ветвлений, определение мертвого кода и т. д.

В случае успешной реализации описанных выше функциональных возможностей эмулятор может превратиться в эффективное средство разработки, отладки и сертификации ПО авиационных комплексов, способное значительно уменьшить трудоемкость процесса разработки, повысить степень его автоматизации. Применение этих мер позволит сократить сроки разработки и улучшить качество создаваемого ПО.

Заключение

Несмотря на высокую изначальную стоимость создания эмулятора, со временем он может стать более финансово выгодным решением, обеспечивающим значительное сокращение трудозатрат при разработке, отладке и сертификации ПО за счет снижения зависимости от наличия целевой аппаратуры, автоматизации процесса разработки, возможности распараллеливания работы, наличия средств анализа и снижения требований к уровню квалификации персонала.

Вместе с тем перманентное расширение сферы применения эмулятора требует постоянного расширения его функциональных возможностей. Некоторые добавляемые функции могут быть реализованы в рамках существующей архитектуры эмулятора без значительных трудозатрат, другие могут потребовать принципиальных изменений. Таким образом, для достижения экономического эффекта при разработке эмулятора большое значение имеет анализ областей его возможного применения и выбор архитектуры, обеспечивающей наращивание функциональных возможностей и доработку на протяжении всего жизненного цикла программы-эмулятора.

Список литературы

1. **Благодатских В. А., Волнин В. А., Поскакалов К. Ф.** Стандартизация разработки программных средств / Под ред. О. С. Разумова. М.: Финансы и статистика, 2005. 288 с.
2. **Богданов А. В., Кирсанова Ю. А., Уткин С. Б., Шек-Иовсепянц Р. А.** Некоторые вопросы создания и использо-

вания виртуальных и физических моделей при разработке аппаратных и программных частей управляющих цифровых комплексов // Мир авионики. 2001. № 1. С. 36—39.

3. **Брауде Э.** Технология разработки программного обеспечения. СПб.: Питер, 2004. 655 с.

4. **Гатчин Ю. А., Жаринов И. О.** Основы проектирования вычислительных систем интегрированной модульной авионики. М.: Машиностроение, 2010. 224 с.

5. **Гатчин Ю. А., Жаринов И. О., Жаринов О. О.** Архитектура программного обеспечения автоматизированного рабочего места разработчика бортового авиационного оборудования // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. 2012. № 2. С. 140—141.

6. **Макконнелл С.** Профессиональная разработка программного обеспечения. Пер. с англ. СПб.: Символ-Плюс, 2006. 240 с.

7. **Парамонов П. П., Жаринов И. О.** Интегрированные бортовые вычислительные системы: обзор современного состояния и анализ перспектив развития в авиационном приборостроении // Научно-технический вестник информационных технологий, механики и оптики. 2013. № 2. С. 1—17.

8. **Хопкрофт Дж., Мотвани Р., Ульман Дж.** Введение в теорию автоматов, языков и вычислений. М.: Вильямс, 2002. 528 с.

9. **QEMU** Emulator User Documentation [Электронный ресурс]. URL: http://qemu.weilnetz.de/qemu-doc.html (дата обращения 17.03.2015).

10. **QEMU** Internals [Электронный ресурс]. URL: http://qemu.weilnetz.de/qemu-tech.html (дата обращения 17.03.2015).

S. A. Blagonravov, Senior Engineer, **S. B. Utkin,** Lead Expert,
SPb Scientific Design Bareu "Electroavtomatica" n. a. P. A. Efimov,
S. V. Batova, Postgraduate Student, **P. V. Konovalov,** Postgraduate Student,
e-mail: olkesomewhere@gmail.com, Saint Petersburg National Research University of Information Technologies, Mechanics and Optics (University ITMO)

Using Emulation Technics in the Development of Avionics Software Components

Development of the software emulation system, designed to support automated system software development process is considered. The emulator hardware platform enables the software components development stages early in the design: the development of software components is carried out on a computer tool, as the code testing, performed using emulator executable environment on the same instrumental machine. The authors suggest the principles of emulator construction. Software development tools used in emulator and functionality of the emulation software are described. Examples demonstrated: an example of the computing system's functional diagram and an example window of the emulator in debug mode. Also presented an algorithm of operating system test process performed on the emulator.

Keywords: *emulation, avionics, real-time operating systems, integrated modular avionics, ARINC 653*

References

1. **Blagodatskikh V. A., Volnin V. A., Poskakalov K. F.** *Standartizatsiya razrabotki programmnykh sredstv* (Standardization of software development), Moscow: Finansy i statistika, 2005, 288 p. (in Russian).

2. **Bogdanov A. V., Kirsanova Yu. A., Utkin S. B., Shek-Iovsepyants R. A.** Some Questions of Creation and Use of Virtual and Physical Models in the Design of Hardware and Software Parts of the Digital Control Systems. *Mir avioniki*, 2001, no. 1, pp. 36—39 (in Russian).

3. **Braude E.** *Tekhnologiya razrabotki programmnogo obespecheniya* (Software Engineering), Saint-Petersburg: Piter, 2004, 655 p. (in Russian).

4. **Gatchin Yu. A., Zharinov I. O.** *Osnovy proektirovaniya vychislitel'nykh sistem integrirovannoi modul'noi avioniki: monografiya* (Basics of Designing Computer Systems for Integrated Modular Avionics: Monograph), Moscow: Mashinostroenie, 2010, 224 p. (in Russian).

5. **Gatchin Yu. A., Zharinov I. O., Zharinov O. O.** Arkhitektura programmnogo obespecheeniya avtomatizirovannogo rabochego mesta razrabotchika bortovogo aviatsionnogo oborudovaniya (Architecture of Software for Automated Workplace of Avionics Developer). *Nauchno-tekhnicheskii vestnik Sankt-Peterburgskogo gosudarstvennogo universiteta informatsionnykh tekhnologii, mekhaniki i optiki*, 2012, no. 2, pp. 140—141 (in Russian).

6. **Makkonnell S.** *Professional'naya razrabotka programmnogo obespecheniya* (Professional software development). Saint-Petersburg: Simvol-Plyus, 2006, 240 p. (in Russian).

7. **Paramonov P. P., Zharinov I. O.** Integrirovannye bortovye vychislitel'nye sistemy: obzor sovremennogo sostoyaniya i analiz perspektiv razvitiya v aviatsionnom priborostroenii (Integrated onboard computing systems: a review of the current state and the analysis of development prospects in avionics of development of aviation instrumentation). *Nauchno-tekhnicheskii vestnik informatsionnykh tekhnologii, mekhaniki i optiki.* 2013. no. 2. pp. 1—17 (in Russian).

8. **Hopkroft J., Motvani R., Ulman J.** *Vvedenie v teoriyu avtomatov, yazykov i vychislenii* (Introduction to Automata, Languages and Computing Theory). Moscow, Vil'yams, 2002, 528 p. (in Russian).

9. **QEMU** Emulator User Documentation, available at: http://qemu.weilnetz.de/qemu-doc.html.

10. **QEMU** Internals, available at: http://qemu.weilnetz.de/qemu-tech.html.

УДК 57.087.1 + 519.165

А. В. Поляков, аспирант, e-mail: andre.levsha@gmail.com,
Московский государственный университет имени М. В. Ломоносова

Алгоритм сравнения отпечатков пальцев на основе структуры созвездий

Введено определение дактилоскопического созвездия и представлен новый алгоритм верификации отпечатков пальцев на основе структуры таких созвездий. Центральной идеей данного подхода является ассоциация топологической конфигурации минуций со структурой созвездий. Вычислительная сложность алгоритма составляет O(n^3) операций при хорошей точности верификации (вероятность ошибки первого рода равна 3 % при вероятности ошибки второго рода в 3 %).

Ключевые слова: *отпечатки пальцев, верификация, минуции, триангуляция Делоне, созвездие, расстояние Левенштейна, венгерский алгоритм*

Введение

Аутентификация личности по отпечаткам пальцев является наиболее распространенной и надежной биометрической технологией. Биометрическая система аутентификации в зависимости от применения может работать в двух режимах: верификации (сравнение один к одному) и идентификации (сравнение один ко многим). Задача верификации состоит в подтверждении личности, в то время как задачей идентификации является установление личности субъекта доступа. В данной статье будем рассматривать задачу верификации.

Пусть даны два отпечатка пальцев I и T. Задача состоит в том, чтобы построить алгоритм и его программную реализацию для сравнения этих отпечатков, которая на выходе выдавала бы меру схожести отпечатков, и на основе этой информации биометрическая система выносила бы решение о принадлежности этих отпечатков одному человеку или разным людям.

К настоящему времени разработано большое число алгоритмов сравнения отпечатков пальцев, которые условно можно разбить на три класса [1]: корреляционное сравнение (вычисление взаимной корреляции двух изображений отпечатка [2]); сравнение минуций (точек обрыва и бифуркаций папиллярных линий [3, 4]); сравнение потоков папиллярных линий и папиллярных узоров [5].

В настоящее время наиболее часто применяется подход, основанный на сравнении минуций. Минуцией (или точкой Гальтона) называется особая точка папиллярной линии, как правило, обрыв линии или раздвоение линии (бифуркация). В этом случае задача верификации отпечатков пальцев сводится к задаче сравнения двух множеств точек, т. е. поиску такого геометрического преобразования (как правило, аффинного), переводящего максимальное число точек из первого отпечатка в точки второго.

Несмотря на большое число существующих биометрических алгоритмов сравнения отпечатков пальцев, задача верификации личности по отпечаткам пальцев остается сложной задачей. Однако она по-прежнему актуальна вследствие большой изменчивости представления отпечатков пальцев. Такая изменчивость обусловлена целым рядом факторов: сдвиги и повороты пальца на сканирующем устройстве; состояние кожи; давление пальца на сканирующее устройство; разрешение сканирующего устройства; шум; ошибки, возникающие при экстракции признаков.

Ежегодно в мире публикуют тысячи статей (по данным Google Scholar, с 2014 г. по настоящее время было опубликовано 8100 статей [6]) по верификации отпечатков пальцев. С 2000 по 2006 г. проводились соревнования Fingerprint Verification Competition (организаторами которого являлись университет Болонии (Италия), Мичиганский государственный университет (США), университет Сан-Хосе (США), Мадридский университет (Испания)) [7, 8].

В настоящей статье введена математическая модель отпечатка пальца, поставлена задача верификации отпечатков пальцев по их шаблонам, даны предварительные сведения и определения, представлен алгоритм верификации отпечатков пальцев, дана оценка его вычислительной сложности по времени; представлены результаты точности работы алгоритма на тестовой базе FVC2002 DB1_b в сравнении с точностью популярного алгоритма верификации отпечатков пальцев NIST MINDTCT/BOZORTH3.

Математическая модель отпечатка пальца

Представим отпечаток пальца в виде нумерованного списка минуций, каждая из которых имеет следующие характеристики: N — номер минуции; координаты $x, y \in \mathbb{N}$ ($\mathbb{N}$ — множество натуральных чисел); угол направления $\theta \in \{0, 2\pi\}$ папиллярной линии в минуциальной точке; тип $\tau \in \{0, 1\}$ (0 — точка обрыва линии; 1 — точка бифуркации линии), вектор гребневого счета **RC** от данной минуции до остальных (гребневый счет между двумя минуциями — это число папиллярных линий, пересекающих отрезок с концами в этих минуциях).

$$I = \{(N_i, x_i, y_i, \theta_i, \tau_i, \mathbf{RC}_i)\}, \quad i = \overline{1, n},$$

где n — общее число минуций в отпечатке.

Постановка задачи

Пусть даны два шаблона отпечатка пальцев $I = \{(N_i, x_i, y_i, \theta_i, \tau_i, \mathbf{RC}_i)\}$, $i = \overline{1,n}$, и $T = \{(N'_j, x'_j, y'_j, \theta'_j, \tau'_j, \mathbf{RC}'_j)\}$, $j = \overline{1,m}$ (m — число минуций в шаблоне T), требуется построить алгоритм сравнения этих отпечатков, который на выходе выдавал бы меру схожести отпечатков $s \in [0, 1]$, и на основе этой информации биометрическая система выносила бы решение о принадлежности этих отпечатков одному человеку или разным людям.

Предварительные сведения и определения

Определение 1. Триангуляцией называется планарный граф, все внутренние области которого являются треугольниками.

Определение 2. Триангуляция Делоне (или граф Делоне) $DT(M)$ — такая триангуляция для заданного множества точек M на плоскости, при которой для любого треугольника все точки из M за исключением точек, являющихся его вершинами, лежат вне окружности, описанной вокруг треугольника.

Впервые триангуляция Делоне была определена в работе [9].

Свойства $DT(M)$:

- Триангуляция Делоне невырожденного множества точек единственна.
- $DT(M)$ максимизирует минимальный угол среди всех построенных треугольников [10].

Определение 3. Пусть S_1 и S_2 — две строки длины m и n соответственно над некоторым алфавитом Σ, тогда расстоянием Левенштейна (расстоянием редактирования) $lv(S_1, S_2)$ называется рекуррентно вычисляемая величина $lv(S_1, S_2) = D(m, n)$:

$$D(i,j) = \begin{cases} 0, & \text{если } i, j = 0; \\ i, & \text{если } j = 0, i > 0; \\ j, & \text{если } i = 0, j > 0; \\ \min(D(i, j-1) + 1, D(i-1, j) + 1, & \\ D(i-1, j-1) + \mathfrak{z}(S_1[i], S_2[j])), & \text{если } i, j > 0, \end{cases}$$

где $\mathfrak{z}(a,b) = \begin{cases} 0, & \text{если } a = b; \\ 1, & \text{иначе} \end{cases}$,

$S_1[i]$ — i-й символ в строке S_1; $S_2[j]$ — j-й символ в строке S_2.

Определение 4. Пусть дано конечное множество точек плоскости $M = \{m_1, ..., m_s\}$, а $DT(M)$ — триангуляция Делоне этого множества. Созвездием с центром в m_i называется подграф $Astr(m_i)$ графа $DT(M)$, $Astr(m_i) = (V, E)$, где V — объединение m_i и множества вершин, смежных с m_i в $DT(M)$, $E = (m_i, m_k)$, $m_k \in V \backslash \{m_i\}$.

Определение 5. Меткой, ассоциированной с созвездием $Astr(m_i)$, называется пара $(m_i, str(m_i))$, где $str(m_i) = s_1 s_2 ... s_t$ — конечное слово над алфавитом $EA = \{a_1, ..., a_{18}\}$, получаемое из созвездия $Astr(m_i)$ по следующему набору правил:

а) перейдем в полярную систему координат с центром в m_i и пронумеруем смежные с m_i вершины по возрастанию полярного угла;

b) представим созвездие $Astr(m_i)$ вектором $\mathbf{AP}(m_i) = \left(m_{neighbor(m_i)_1}, ..., m_{neighbor(m_i)_t}\right)$, где $m_{neighbor(m_i)_l}$ — l-я смежная с m_i минуция; t — число смежных с m_i минуций;

c) заменим все минуции в $\mathbf{AP}(m_i)$ соответствующими значениями их направлений:

$$\mathbf{AP}(\theta_i) = \left(\theta_{neighbor(m_i)_1}, ..., \theta_{neighbor(m_i)_t}\right);$$

d) для каждой компоненты $\theta_{neighbor(m_i)_l}$ вектора $\mathbf{AP}(\theta_i)$ вычислим

$$k_l = ceiling\left(\min\left(\begin{array}{l} \left|\theta_i - \theta_{neighbor(m_i)_l}\right|, \\ 360 - \left|\theta_i - \theta_{neighbor(m_i)_l}\right| \end{array}\right) / 10\right),$$

где функция $celling(x)$ возвращает наименьшее целое число, большее, либо равное x, $l = \overline{1, t}$; в результате получим вектор $\mathbf{AP}(\theta_i) = (k_l, ..., k_t)$;

e) присвоив символам s_l значение a_{k_l}, для всех $l \in \overline{1, t}$ получим слово $str(m_i) = s_1 s_2 ... s_t$.

Определение 6. Метки $str(m_i)$ и $str(m'_j)$ назовем похожими (обозначение $str(m_i) \approx str(m'_j)$, если расстояние Левенштейна $lv(str(m_i), str(m'_j)) < \sigma_{порог}$, где $\sigma_{порог}$ — калибровочный параметр системы.

Определение 7. Звездным расстоянием между минуцией m_i и всеми ее смежными вершинами в графе Делоне назовем вектор $\mathbf{AD}(m_i) = (r_1, ..., r_t)$, где $r_s = \mathbf{RC}(m_i, m_s)$; t — число вершин, смежных с m_i в $Astr(m_i)$, и упорядоченных так же, как в определении 4.

Определение 8. Минуции $m_i \in I$ и $str\, m'_j \in T$ назовем совпадающими, если выполняются следующие условия:

1) $str(m_i) \approx str(m'_j)$;

2) $\tau_i = \tau'_j$;

3) Для всех $w = \overline{1, \min\left(length(str(m_i)), length(str(m'_j))\right)}$,

$$|\mathbf{AD}(m_i)[w] - \mathbf{AD}(m'_j)[w]| < \rho_{порог},$$

где $\mathbf{AD}(m_i)[w]$, $\mathbf{AD}\left(m'_j\right)[w]$ — w-е компоненты векторов $\mathbf{AD}(m_i)$ и $\mathbf{AD}\left(m'_j\right)$ соответственно; $\rho_{\text{порог}}$ — комбинированный порог системы.

Определение 9 (задача о назначениях). Пусть дан двудольный граф $B = (V, U, E)$, представленный матрицей смежности $\mathbf{X} = (x_{ij})$, $i = \overline{1, n}$, $j = \overline{1, m}$, $n \leqslant m$, $x_{ij} \in \{0, 1\}$. Каждое ребро, соединяющее i-ю вершину с j-й вершиной, имеет вес $c_{ij} \in \mathbb{Z}^+$ ($\mathbb{Z}^+$ — множество неотрицательных целых чисел). Тогда задача о назначениях формулируется следующим образом:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \to \max \text{ при условиях } \sum_{i=1}^{n} x_{ij} = 1, \ \sum_{j=1}^{m} x_{ij} = 1.$$

Для решения задачи о назначениях на практике применяется венгерский алгоритм, разработанный Гарольдом Куном в 1955 г. На рисунке приведен пример программной реализации венгерского алгоритма, разработанной А. Лопатиным (г. Санкт-Петербург).

Алгоритм работает описанным далее образом [11].

На вход алгоритму дается матрица **a** размера ($n \times m$), где $n \leqslant m$.

Массивы $u[0...n]$ и $v[0...m]$ хранят веса. Сначала они нулевые, что верно для матрицы, состоящей из нуля строк (отметим, что для данной реализации не важно, имеются или нет в матрице **a** отрицательные числа).

```
vector<int> u (n+1), v (m+1), p (m+1), way (m+1);
for (int i=1; i<=n; ++i) {
    p[0] = i;
    int j0 = 0;
    vector<int> minv (m+1, INF);
    vector<char> used (m+1, false);
    do {
        used[j0] = true;
        int i0 = p[j0],  delta = INF,  j1;
        for (int j=1; j<=m; ++j)
            if (!used[j]) {
                int cur = a[i0][j]-u[i0]-v[j];
                if (cur < minv[j])
                    minv[j] = cur,  way[j] = j0;
                if (minv[j] < delta)
                    delta = minv[j],  j1 = j;
            }
        for (int j=0; j<=m; ++j)
            if (used[j])
                u[p[j]] += delta,  v[j] -= delta;
            else
                minv[j] -= delta;
        j0 = j1;
    } while (p[j0] != 0);
    do {
        int j1 = way[j0];
        p[j0] = p[j1];
        j0 = j1;
    } while (j0);
}

vector<int> ans (n+1);
for (int j=1; j<=m; ++j)
    ans[p[j]] = j;

int cost = -v[0];
```

Пример программной реализации венгерского алгоритма

Массив $p[0...m]$ содержит паросочетание: для каждого столбца $i = \overline{1, m}$ он хранит номер соответствующей выбранной строки $p[i]$ (или 0, если пока ничего не выбрано). При этом $p[0]$ для удобства реализации полагается равным номеру текущей рассматриваемой строки.

Массив $minv[1...m]$ содержит для каждого столбца j вспомогательные минимумы, необходимые для быстрого пересчета весов:

$$minv[j] = \min_{i \in Z_1} \{a(i, j) - u[i] - v[j]\},$$

где $Z_1 \subset V$ — множество вершин первой доли графа B, которые были посещены обходом венгерского алгоритма.

Массив $way[1...m]$ содержит информацию о том, где эти минимумы достигаются, чтобы впоследствии можно было восстановить увеличивающую цепочку. На первый взгляд кажется, что в массиве $way[]$ для каждого столбца нужно хранить номер строки, а также завести еще один массив — для каждой строки запомнить номер столбца, из которого в нее пришли. Однако вместо этого можно заметить, что алгоритм Куна всегда попадает в строки, проходя по ребру паросочетания из столбцов, поэтому номера строк для восстановления цепочки всегда можно взять из паросочетания (т. е. из массива $p[]$). Таким образом, $way[j]$ для каждого столбца j содержит номер предшествующего столбца (или 0, если такого нет).

Сам алгоритм представляет собой внешний цикл по строкам матрицы, внутри которого происходит добавление в рассмотрение i-ой строки матрицы. Внутренняя часть представляет собой цикл "do-while ($p[j_0]! = 0$)", который работает, пока не будет найден свободный столбец j_0. Каждая итерация цикла помечает посещенным новый столбец с номером j_0 (посчитанным на прошлой итерации, а изначально равным нулю — т. е. первый шаг с фиктивного столбца), а также новую строку i_0 — смежную ему в паросочетании (т. е. $p[j_0]$; а изначально при $j_0 = 0$ берется i-я строка). В силу появления новой посещенной строки i_0 нужно соответствующим образом пересчитать массив $minv[]$ и одновременно найти минимум в нем — $delta$, а также номер столбца j_1, в котором этот минимум был достигнут (заметим, что при такой реализации значение $delta$ могло оказаться равным нулю, что означает, что на текущем шаге вес можно не менять, так как существует новый достижимый столбец). После этого проводится пересчет весов $u[]$, $v[]$, а затем — соответствующее изменение массива $minv[]$. По окончании цикла "do-while" получаем увеличивающую цепочку, оканчивающуюся в столбце j_0, "раскрутить" которую можно, пользуясь массивом предков $way[]$. Константа INF — это "бесконечность", т. е. некоторое число, заведомо большее всех возможных чисел во входной матрице **a**. Стоимость найденного паросочетания можно взять как вес нулевого столбца (взятый с противоположным знаком). В самом деле, $-v[0]$ содержит в себе сумму всех значений $delta$, т. е. суммарное изменение веса.

Хотя при каждом изменении веса изменяться могли сразу несколько значений $u[i]$ и $v[j]$, суммарное изменение веса в точности равно *delta*, поскольку пока нет увеличивающей цепи, число достижимых строк ровно на единицу больше числа достижимых столбцов (только текущая строка i не имеет себе "пары" в виде посещенного столбца).

Алгоритм верификации отпечатков пальцев на основе структуры созвездий (*Astr*-алгоритм)

Алгоритм верификации отпечатков пальцев на основе структуры созвездий представляется следующим образом.

Вход:

$I = \{(N_i, x_i, y_i, \theta_i, \tau_i, \mathbf{RC}_i)\}$, $i = \overline{1,n}$ — реализация математической модели образца отпечатка пальца, полученного со сканирующего устройства;

$T = \left\{\left(N'_j, x'_j, y'_j, \theta'_j, \tau'_j, \mathbf{RC}'_j\right)\right\}$, $j = \overline{1,m}$ — реализация математической модели шаблона отпечатка пальца, взятого из базы данных.

Выход: $s \in [0, 1]$.

Шаг 1. Процедура конструирования $DT(I)$, $DT(T)$ — графов Делоне I и T соответственно.

Шаг 2. Для $DT(I)$ и $DT(T)$ составляем списки созвездий $List_I(Astr_i)$ и $List_T(Astr_j)$ соответственно, которые представлены в виде векторов вида

$$\mathbf{List}_I(Astr_i) = \{(m_i, neighbor(m_i)_1, ..., neighbor(m_i)_s)\}$$

$$\mathbf{List}_T(Astr_j) = \{(m_j, neighbor(m_j)_1, ..., neighbor(m_j)_r)\},$$

где $neighbor(m_i)_1, ..., neighbor(m_i)_s$ — вершины созвездия $Astr_i$ с центром m_i, упорядоченные по возрастанию полярного угла (при этом вершина m_i находится на месте первой координаты в векторе).

Шаг 3. Кодирование созвездий метками.

Из списков созвездий $List_I(Astr_i)$ и $List_T(Astr_j)$ формируем соответствующие списки меток $Wordlist_I$ и $Wordlist_T$.

Шаг 4. Для каждой метки из $Wordlist_T$ формируем множество меток, получающихся из данной циклическими перестановками, и добавляем его в $Wordlist_T$.

Шаг 5. Процедура построения декартова произведения

$$CartProd = Wordlist_I \times Wordlist_T.$$

Шаг 6. Процедура вычисления расстояния Левенштейна между всеми парами слов из *CartProd*.

Выбираем подмножество $CartProd_\sigma \subset CartProd$ — множество пар центров созвездий с расстояниями Левенштейна, меньшими $\sigma_{порог}$.

Шаг 7. Процедура голосования.

За каждую пару $(m_i, m'_j) \in CartProd_\sigma$ отдается число голосов, равное $|\min(length(str(m_i)), length(str(m'_j))) - lv(str(m_i), (str(m'_j))|$, где $length(str)$ — длина строки str.

Шаг 8. Венгерский алгоритм поиска максимального паросочетания в двудольном графе.

$CartProd_\sigma$ представляется в виде взвешенного двудольного графа $G = (V \cup W, E)$, где минуции из I принадлежат V, а минуции из T принадлежат W, а веса ребер определяются числом голосов, отданных за данную пару минуций.

С помощью венгерского алгоритма проводим поиск максимального паросочетания в двудольном графе максимального веса.

Выход: Список $Hungarian\_List = \{(m_i, m'_j)\}$ соответствующих друг другу центров созвездий.

Шаг 9. Пусть *Condition 1* и *Condition 2* — условия 2 и 3 определения 8 соответственно.

Цикл

```
Counter ← 0
A ← NULL
FOR (i=1 to length(Hungarian _ List)) DO {
IF ((Condition1(Hungarian _ List[i])==TRUE)&&
(Condition2(Hungarian _ List[i]) = = TRUE))
    {
A ← A∪Hungarian _ List[i]
Counter++
    }
}
```

Для всех элементов максимального паросочетания, полученных на предыдущем шаге, проверяем условия 2 и 3 определения 8 и составляем множество совпадающих пар минуций A мощности *Counter*.

Шаг 10. Добавление соседей.

Для каждой пары из множества A добавляем вершины — соседей, которые "голосовали" за пару на предыдущем этапе.

Если в результатах голосования среди победителей оказались пары точек, не получившие голосов соседей, то такие пары удаляются.

Выход: Список пар *Winners*, $|Winners| = K$.

Шаг 11. Вычисление степени похожести отпечатков

$$s = \frac{2K}{m+n}.$$

Утверждение. Пусть $m < n$. Тогда вычислительная сложность *Astr*-алгоритма составляет $O(n^3)$ операций.

Доказательство.

На шаге 1 построения триангуляции Делоне $DT(I)$ и $DT(T)$) требуется $O(n\log n)$ операций.

На шаге 2 требуется проверка смежности всех вершин графа. Вычислительная сложность составляет $O(n)$ операций.

Шаг 3 требует $O(n)$ операций.

На кодирование также необходимо $O(n)$ операций.

Шаг 4 требует $O(n^2)$ операций

Шаг 5 требует $O(n^2)$ операций.

Шаг 6 требует $O(n^2)$ операций.

Шаг 7 требует $O(n)$ операций.

Шаг 8 требует $O(n^3)$ операций.

Шаг 9 требует $O(n)$ операций.

Шаг 10 требует $O(n)$ операций.

Шаг 11 требует $O(1)$ операций.

Таким образом, общая вычислительная сложность *Astr*-алгоритма составляет $O(n^3)$ операций.

Утверждение доказано.

Экспериментальные результаты

Для программной реализации алгоритма были выбраны структурная парадигма разработки программного обеспечения и метод разработки "сверху вниз".

Программная реализация алгоритма была написана автором на языке R в свободно распространяемой среде RStudio. Выбор языка программирования обусловлен кроссплатформенностью (Windows/Unix/Mac OS), простотой реализации алгоритма благодаря поддержке R широкого спектра статистических и численных методов, а также хорошей расширяемостью с помощью пакетов. Пакеты представляют собой библиотеки для работы специфических функций или специальных областей применения. В частности, для построения триангуляции Делоне использовалась функция delaunayn() из пакета "geometry", а для реализации венгерского алгоритма была взята функция solve_LSAP () из пакета "clue".

Был проведен эксперимент на базе отпечатков FVC2002 DB1_b (всего 80 отпечатков, 10 пальцев по 8 изображений каждого пальца). Эта база данных была специально составлена для международных соревнований разработчиков алгоритмов верификации и является одной из основных баз данных отпечатков пальцев, использующихся для тестирования алгоритмов верификации отпечатков пальцев.

Характеристики используемой ЭВМ: Intel Core i5-2500 CPU @3.30 GHz 3.30GHz, 4 Гбайт ОЗУ, ОС Windows 7.

Было проведено 3160 сравнений (из них 280 сравнений для своих пальцев и 2880 — для чужих).

Для сравнительной оценки результатов работы *Astr*-алгоритма был выбран алгоритм NIST MINDTCT/BOZORTH3, основанный на сравнении минуций и используемый в свободно распространяемом программном обеспечении SourceAFIS 1.7 [12].

Результаты работы *Astr*-алгоритма и алгоритма NIST MINDTCT/BOZORTH3 при использовании базы данных FVC2002 DB1_b представлены в таблице. EER (*equal error rate*) — значение, при котором вероятность ошибки первого рода равна вероятности ошибки второго рода.

Сравнение *Astr*-алгоритма с алгоритмом NIST MINDTCT/BOZORTH3 по точности

Алгоритм	EER, %
NIST MINDTCT/BOZORTH3	3,6
Astr-алгоритм	3

Заключение

Представленный в статье алгоритм может быть включен в качестве компонента в состав программно-аппаратного комплекса, предназначенного для биометрической верификации личности. Отметим, что данный алгоритм имеет следующие ограничения, присущие классу алгоритмов, основанных на сравнении минуций: во-первых, предполагается, что оба шаблона отпечатков содержат достаточно большое число минуций (т. е. сканируется достаточно большая область поверхности обоих пальцев), во-вторых, предполагается, что не существует двух людей, отпечатки пальцев которых имеют "похожую" минуциальную структуру. Наконец, все алгоритмы, основанные на анализе минуций, чувствительны к качеству изображений: в случае плохого качества могут некорректно определяться типы минуций, что влечет ошибки при построении шаблона. В следующей работе будут более подробно рассмотрены ограничения на работу класса алгоритмов, основанных на сравнении минуций.

Список литературы

1. **Maltoni D., Maio D., Jain A. K.** Handbook of fingerprint recognition. New York: Springer-Verlag, 2003.
2. **Bazen A. M., Verwaaijen G. T. B., Gerez S. H.** et al. A Correlation-Based Fingerprint Verification System // Proc. of ProRISC 2000 Workshop on Circuits, Systems and Signal Processing. November 2000. Veldhoven, The Netherlands. 2000. P. 205—213.
3. **Starink J., Backer E.** Finding point correspondences using simulated annealing // Pattern Recognition. 1995. Vol. 28, N 2. P. 231—240.
4. **Bebis G., Deaconu T., Georgiopoulos M.** Fingerprint Identification Using Delaunay Triangulation // Proc. of IEEE International Conference on Information Intelligence and Systems. 1999. P. 452—459.
5. **Ito K., Morita A., Aoki T.** et al. Fingerprint Recognition Algorithm Combining Phase-Based Image Matching and Feature-Based Matching // Proc. of International Conference on Biometrics. 2006. LNCS 3832. P. 316—325.
6. **Список** статей, посвященных верификации по отпечаткам пальцев, в системе "Google Scholar". URL: https://scholar.google.ru/scholar?as_ylo = 2014&q = fingerprint+verification&hl = ru&as_sdt = 0,5 (дата обращения 08.04.2015).
7. **Система** тестирования алгоритмов верификации "FVC-onGoing" лаборатории биометрических систем Болонского университета (Италия), URL: https://biolab.csr.unibo.it/FVCOnGoing/UI/Form/Home.aspx (дата обращения 08.04.2015).
8. **Dorizzi B., Cappelli R., Ferrara M.** et al. Fingerprint and On-Line Signature Verification Competitions at ICB 2009 //

Proc. of International Conference on Biometrics (ICB). June 2009. Alghero, Italy. Springer, 2009. P. 725—732.

9. **Делоне Б. Н.** О пустоте сферы // Изв. АН СССР. ОМЕН. 1934. № 4. С. 793—800.

10. **Скворцов А. В.** Триангуляция Делоне и ее применение. Томск: Изд-во Том. гос. ун-та, 2002. 128 с.

11. **Портал** MAXimal: алгоритмы и олимпиадное программирование. Венгерский алгоритм решения задачи о назначениях. URL: http://e-maxx.ru/algo/assignment_hungary (дата обращения 08.04.2015).

12. **Программное** обеспечение для верификации отпечатков пальцев. URL: http://sourceforge.net/projects/sourceafis/

A. V. Poliakov, Postgraduate Student**,** e-mail: andre.levsha@gmail.com,
Lomonosov Moscow State University

Fingerprint Matching Algorithm Using a Constellation Structure

Fingerprint matching is a key problem in research of an automatic fingerprint identification system. This paper presents a novel algorithm to fingerprint verification using constellation structure based on Delaunay triangulation. Central to the proposed approach is the idea of associating a topological structure with the fingerprint minutiae using constellation structure.

The proposed fingerprint matching algorithm is insensitive to fingerprint image distortion, scale and rotation. The proposed approach considers O(n^3) operations. The proposed algorithm has been tested on a database FVC2002 DB1b of 80 fingerprints (8 fingerprints from 10 persons), demonstrating good performance.

Keywords: *fingerprint matching algorithm, Delaunay triangulation, constellation, minutia, biometric system, Hungarian algorithm*

References

1. **Maltoni D., Maio D., Jain A. K.** Handbook of fingerprint recognition. New York: Springer-Verlag, 2003.

2. **Bazen A. M., Verwaaijen G. T. B., Gerez S. H., Veelenturf L. P. J., van der Zwaag B. J.** A Correlation-Based Fingerprint Verification System. *Proc. of ProRISC 2000 Workshop on Circuits, Systems and Signal Processing,* November 2000. Veldhoven, The Netherlands, 2000, pp. 205—213.

3. **Starink J., Backer E.** Finding point correspondences using simulated annealing. *Pattern Recognition*, 1995, vol. 28, no. 2, pp. 231—240.

4. **Bebis G., Deaconu T., Georgiopoulos M.** Fingerprint Identification Using Delaunay Triangulation. *Proc. of IEEE International Conference on Information Intelligence and Systems*, 1999. pp. 452—459.

5. **Ito K., Morita A., Aoki T., Nakajima H., Kobayashi K. and Higuchi T.** A Fingerprint Recognition Algorithm Combining Phase-Based Image Matching and Feature-Based Matching. *Proc. of International Conference on Biometrics*, 2006, LNCS 3832, pp. 316—325.

6. **List** of papers devoted to fingerprint verification according "Google Scholar". available at: https://scholar.google.ru/scholar?as_ylo=2014&q=fingerprint+verification&hl=ru&as_sdt=0,5 (date of access 08.04.2015).

7. **FVC-onGoing:** on-line evaluation of fingerprint recognition algorithms, Biometric System Laboratory, University of Bologna, Italy, available at: https://biolab.csr.unibo.it/FVCOnGoing/UI/Form/Home.aspx (date of access 08.04. 2015).

8. **Dorizzi B., Cappelli R., Ferrara M. Maio D., Maltoni D., Houmani N., Garcia-Salicetti S., Mayoue A.** Fingerprint and On-Line Signature Verification Competitions at ICB 2009. *Proc. of International Conference on Biometrics (ICB)*, 2—5 June 2009, Alghero, Italy. Springer, 2009, pp. 725—732.

9. **Delaunay B.** Sur la sphère vide. A la mémoire de Georges Voronoï. *Bulletin de l'Académie des Sciences de l'URSS, Classe des sciences mathématiques et naturelles*, 1934, no. 6, pp. 793—800.

10. **Skvortsov A. V.** Triangulatsiya Delone i ee primenenie (Delaunay triangulation and its application). Tomsk: Tomsk university press, 2002. 128 p. (in Russian).

11. **MAXimal**: algorithms and Olympiad programming, "Hungarian algorithm for assignment problem", available at: http://e-maxx.ru/algo/assignment_hungary (date of access 08.04.2015).

12. **Fingerprint** recognition library for .NET and experimentally for Java, available at: http://sourceforge.net/projects/sourceafis/

УДК 004.912

И. С. Кононенко, науч. сотр., e-mail: irina_k@cn.ru, **Е. А. Сидорова**, канд. физ.-мат. наук, ст. науч. сотр., e-mail: lena@iis.nsk.su, Новосибирский государственный университет, Институт систем информатики им. А. П. Ершова СО РАН, г. Новосибирск

Жанровые аспекты классификации веб-сайтов

Проанализированы особенности представленной авторами двухуровневой жанровой модели веб-сайтов. На макроуровне жанр веб-сайта описывается с помощью прагматических параметров, характеризующих различные аспекты интернет-деятельности. На микроуровне жанру текстовой страницы сайта соответствует характерная логико-композиционная (жанровая) структура, которая складывается как последовательность содержательных текстовых блоков. Границы блоков идентифицируются с помощью лексикона жанровых маркеров. Описывается архитектура системы классификации сайтов, реализующей предложенную модель.

Ключевые слова: *веб-жанр, праксиологический параметр, коммуникативный параметр, жанровая структура текста, жанровый маркер, лексико-грамматический шаблон, жанровая модель веб-сайта, классификация веб-сайтов*

Введение

Современные исследования общения уделяют все большее внимание особенностям коммуникативных практик, которые реализуются в электронной среде на многочисленных веб-страницах и сайтах. Эти явления служат объектом изучения для новых направлений исследований, таких как лингвистика Интернета и виртуальное жанроведение [1—3]. В бесконечном разнообразии сайтов существуют общие черты, относительно устойчивые типы, которые определяются спецификой конкретной сферы общения и формируются благодаря сходству тематического содержания, единству стиля и композиционного построения. Все эти черты соответствуют классическому определению жанра речевого произведения, сформулированному М. М. Бахтиным [4]. По Бахтину, жанр — это типовая модель построения речевого целого. Каждому типу сайта должна соответствовать жанровая модель, представляющая его "типичную воспроизводимую жанровую форму".

Решение многих прикладных задач, связанных с исследованием процесса коммуникации в электронной среде, таких как автоматическая классификация веб-ресурсов, извлечение из них релевантной информации или автоматическая генерация элементов структуры и дизайна, неизбежно приводит исследователей к необходимости рассмотрения проблематики веб-жанров. При этом основной вопрос, требующий разрешения, состоит в определении самого понятия веб-жанра и исчислении жанровых типов. По некоторым данным, число жанров, которые можно выделить в разнообразии веб-страниц и сайтов, доходит до 6000 (см. работу [5], где этот вопрос решается в рамках экспериментального корпусного исследования).

Разнообразные современные подходы к определению веб-жанров представлены в сборнике [6]. При обсуждении веб-жанров во вступительной статье [7] отмечены недостатки классификации веб-ресурсов по чисто тематическому принципу. Представления, например, одной и той же темы, а именно вопросов имплантации, на сайте стоматологического центра и на форуме стоматологов кардинально различаются. Жанровые категории должны быть максимально тематически нейтральными, хотя на практике различные жанры связаны с тематикой в разной степени. В каталоге Яндекса (yaca.yandex.ru) намечен путь к учету различий тематики и жанра. В нем, кроме обширного тематического дерева, используется типовая классификация, которая подразделяет все сайты на пять типов (сайты с предложениями товаров и услуг, советы и инструкции, справочные издания, форумы, мероприятия). В работе [8] жанровая проблематика рассмотрена в тесной связи с классификацией функциональных стилей, отражающих сферы человеческой деятельности. Отмечена неполнота известных классификаций. Авторы работы [9] усматривают природу жанра в многоаспектной природе коммуникативного акта.

При всей сложности понятия веб-жанра в нем прослеживаются функциональная, формальная (композиционная и лексико-грамматическая) и содержательная стороны [10], которые соотносятся с различными уровнями физической репрезентации (сайт в целом, страница, отдельные составляющие страниц). Жанровые характеристики могут рассматриваться по крайней мере для двух уровней представления ресурса: сайт в целом (макроуровень) и страница сайта (микроуровень). При этом сайт представляет собой целостное речевое произведение, фиксирующее акт коммуникативного взаимодействия автора сайта и его целевой аудитории. Как и любой акт коммуникации, сайт погружен в контекст некоторого акта совместной деятельности, поэтому жанровая репрезентация сайта наряду с прагматическими параметрами, относящимися к собственно коммуникации, должна учитывать общий деятельностный контекст.

В данной работе изложен подход к созданию двухуровневых жанровых моделей веб-сайтов на базе многоаспектной прагматической классификации ресурсов и их структурных составляющих (раздел 1). Прагматика сайта отражается в логико-композиционной структуре текста веб-страниц, которая выявляется с помощью лексикона жанровых маркеров (раздел 2). В разделе 3 представлена архитектура системы распознавания жанра веб-сайта и его страниц.

1. Модель веб-сайта

В основу предлагаемой классификации легли два типа прагматических параметров. Праксиологические параметры позволяют дифференцировать виды деятельности и необходимы для представления деятельностного контекста коммуникации. Коммуникативные параметры предназначены для описания коммуникативных задач и особенностей коммуникативной ситуации, которые связаны с использованием электронной среды (сети Интернет) в качестве канала коммуникации[1].

Как было указано выше, модели веб-сайтов могут рассматриваться по крайней мере для двух уровней представления ресурса: сайт в целом (макроуровень) vs. страница сайта (микроуровень).

Определим формальную модель сайта как систему вида

$$M = \langle G, T, A, P, CT, RT, M_P \rangle,$$

где G — жанр сайта;

T — тематика (тема или множество тем) сайта из заданного тематического классификатора[2];

$A \subseteq O_A$ — множество поддерживаемых сайтом видов деятельности, где O_A — онтология деятельности общего вида;

P — множество праксиологических параметров сайта;

CT — множество коммуникативных задач, решаемых с помощью сайта;

RT — множество реактивных задач, выполняемых целевой аудиторией сайта;

M_P — множество моделей страниц, составляющих веб-сайт.

Взаимосвязь всех характеристик сайта определяется следующим набором функций.

F_A: $2^P \to O_A$ — сочетаемостная функция праксиологических параметров, которая по подмножеству праксиологических параметров P определяет вид деятельности $a \in O_A$.

F_T: $2^{RT} \to CT$ — функция, которая по подмножеству низкоуровневых реактивных задач определяет коммуникативную задачу сайта.

F_G: $O_A \times CT \to G$ — сочетаемостная функция, которая по виду деятельности и решаемой посредством сайта коммуникативной задаче определяет жанр веб-сайта.

Далее рассмотрим модель страницы, относящуюся к микроуровневому представлению веб-сайта. Под страницей как единицей микроуровня подразумевается структурная составляющая сайта, которая физически может быть представлена не только веб-страницей, но и разделом (группой веб-страниц) или блоком в рамках одной веб-страницы.

Формально модель страницы описывается системой вида

$$M_P = \langle g_p, T_p, P_c, S \rangle,$$

где G_p — жанровый тип страницы;

T_p — тематика страницы;

P_c — множество коммуникативных параметров, которые включают функцию страницы и реализованные в ней свойства канала коммуникации;

S — логико-композиционная структура текста страницы.

Взаимосвязь коммуникативных параметров страницы сайта с ее жанром определяется отображением вида F_{Gp}: $P_c \to 2^{Gp}$, где каждому коммуникативному параметру $pc \in P_c$ ставится в соответствие подмножество жанров страниц $G'_p \subseteq G_p$, обладающих данным свойством.

Рассмотрим подробнее базовые параметры предложенной модели.

1.1. Праксиологические параметры

Деятельность (A) принято рассматривать как целенаправленное взаимодействие субъекта с объектом, включающее такие компоненты, как продукт (реализованная цель), средства, процесс и условия.

С онтологической точки зрения все понятия, имеющие отношение к A, можно рассматривать как сущности, подразделяемые на Объектные и Процессные, с одной стороны, и Агентивные и Неагентивные; с другой стороны. Процесс определяет совокупность ролей (ролевых отношений) участников ситуации, каждая из которых представляет определенный аспект Процесса предполагает ограничения на возможные заполнения соответствующими сущностями-носителями ролей. Агентивные сущности, в отличие от Неагентивных, имеют волю, осуществляют целеполагание, а значит могут высту-

[1] Традиционно принято противопоставлять устный и письменный каналы передачи информации (модусы). Дополнительно выделяют жестовый модус, основанный на визуальном взаимодействии, и ряд субмодусов, связанных с развитием технологий. В последнее время значение коммуникации в электронной среде, особенно в сети Интернет, настолько велико, что говорят о появлении электронного модуса, наряду с устным и письменным [8].

[2] Анализ вопросов, связанных с тематикой сайта, выходит за рамки данного исследования, поэтому тематическая составляющая модели представлена здесь схематично. Более детально представлены параметры, которые относятся к жанровой составляющей модели.

пать в роли Субъекта деятельности. Помимо роли Субъекта, Агентивным сущностям соответствуют такие роли, как Объект, Контрагент и Бенефициант. Неагентивные сущности представлены Физическими, Социальными, Ментальными объектами/процессами, Темпоральными и Локативными объектами. Им соответствуют следующие роли: Объект, Продукт, Бенефициант, Время, Место, Среда и т. п. Онтологические концепты допускают уточнение в рамках некоторой родовидовой иерархии. Любой концепт, как сущностный, так и ролевой, может характеризоваться атрибутом, описывающим свойства данного понятия.

Понятия онтологии позволяют сформулировать множество праксиологических параметров, которые могут служить основой для многоаспектной классификации деятельности. Рассмотрим некоторые параметры и примеры соответствующих им жанровых различий (рис. 1).

Субъект: {*Индивид, Группа, Организация, Государство*}. *Группа* может уточняться как *Социальная группа* (в частности, *Семья*) или *Сообщество* (группа по интересам). Противопоставление значений данного параметра имеет рефлексы в жанровой форме сайта: в текстах страниц встречаются местоимения первого лица единственного числа, характерные для персональной странички (персональное общение) и неприемлемые на сайте организации (институциональное общение), где для определенных видов деятельности (коммерческая, рекламная) частотны местоимения первого лица множественного числа. Заметим, что в официальном стиле, характерном для сайтов государственных органов, местоимения первого лица отсутствуют.

Рис. 1. Праксиологические параметры веб-сайтов

Бенефициант: {*Индивид, Группа, Организация, Социум*}. Бенефактивное отношение связывает деятельность *A* с тем участником, который пользуется ее результатом (Продуктом). Параметр позволяет различить продукты, потребляемые индивидуально, коллективно или обществом в целом, и соответствующие им виды деятельности.

Продукт.Отделимость: {*Да, Нет*}. Данный параметр позволяет различить такие виды деятельности, как *производство товаров* и *услуг*: услуга — это деятельность, результаты которой реализуются и потребляются в процессе деятельности. В связи с этим обстоятельством в схеме сайта услуг появляется содержательный блок "Запись на прием".

Продукт.Форма: {*Вещественная, Невещественная*}. Многие продукты, не имея вещественной формы, представляют собой духовные и физические свойства личности, неотчуждаемые от их носителя (здоровье, знания и т. п.).

Продукт.Вид: {*Материальный, Финансовый, Информационный, Духовный, Витальный*}. Данный параметр взаимодействует с предыдущим, позволяя выделить разновидности продуктов, например:

<*Вещественный, Материальный*> — продукты *материального производства* (оборудование, здания, мебель) и *материальных услуг* (коммунальное и бытовое обслуживание, ремонт, транспортные услуги);

<*Невещественный, Витальный*> — состояние физического и психического здоровья как результат *медицинских* и *рекреационных услуг*;

<*Вещественный, Духовный*> — результаты *духовной* деятельности в ее вещественном воплощении, включая произведения художественной литературы и искусства, научную и учебную литературу, социокультурные мероприятия.

Вещественным по форме продуктам услуг дизайна сопутствует духовная эстетическая составляющая, которая имеет рефлексы на лексическом уровне (частотность эмоционально-экспрессивной оценочной лексики, в отличие от рациональной оценки материальных продуктов), наличие содержательных блоков, характеризующих (разумеется, с положительной тональностью) результаты деятельности с точки зрения некоторой оценочной шкалы (награды, участие в престижных творческих конкурсах).

Продукт.Тип: {*Цифровой, Физический*}. Этот параметр позволяет отличить продукты, распределение которых возможно в электронной форме, в том числе информационные файлы (текстовые, графические, звуковые), программные продукты, финансовые инструменты (чеки, ценные бумаги) и т. п., от продуктов, имеющих реальную физическую форму.

Среда: {*Виртуальная, Реальная*}. Значение *Виртуальная* выделяет деятельность, которая осуществляется в электронной среде с использованием сети Интернет или иных электронных средств. Сочетание значений этого параметра и параметра Продукт. Тип дифференцирует различные виды *A* по степени использования электронных средств. Так, в сфере электронной торговли оперируют электронные магазины, предлагающие *Физический* продукт, и электронные библиотеки, поставляющие *Цифровой* продукт. Соответствующие различия проявляются в структуре сайта на уровне наличия/отсутствия блоков, описывающих локативные и темпоральные условия *A* (*Реальная*), способы доставки *Физического* продукта в отличие от *Цифрового* продукта.

Возможные комбинации значений праксиологических параметров образуют вектор в пространстве признаков, характеризующий деятельностный контекст сайта. Вклад конкретных значений этих признаков проявляется на уровне структуры сайта, состава, содержания и языкового оформления составляющих его страниц и блоков.

1.2. Коммуникативные параметры

Веб-сайт рассматривается как коммуникативный акт, осуществляемый владельцем (автором) сайта в рамках объемлющей деятельности *A*, субъектом которой является сам автор, а контрагентом — целевая аудитория (адресат). Автор передает адресату определенную информацию, которая является отражением тематики сайта *T* и деятельности *A*.

1.2.1. Сайт нацелен на выполнение определенных *коммуникативных задач*, которым соответствуют *реактивные задачи* для целевой аудитории.

Выделяют три типа коммуникативных задач: фатическую, иллокутивную и перлокутивную. Фатическая задача автора состоит в том, чтобы наладить общение с адресатом. Иллокутивная задача состоит в передаче адресату информации о *T* и/или *A* (в частности, с целью получить от адресата информацию, необходимую для реализации деятельности *A*). Перлокутивная задача заключается в том, чтобы обеспечить желаемую реакцию адресата, т. е. выполнение им *Реактивной задачи*.

Предлагаемая типология реактивных задач включает *фатическую, когнитивную, иллокутивную и прагматическую* задачи. Рассмотрим типы и взаимосвязи реактивных задач подробнее.

- *Фатическая* задача — задача, реализация которой означает, что адресат становится участником общения (вступает в коммуникацию и поддерживает ее).
- *Когнитивная* задача соответствует процессам, связанным с переработкой информации, полученной от автора. Рассматривают два вида таких процессов и, соответственно, два вида когнитивных подзадач:
  - ◇ *информационную*, когда адресат усваивает информацию о *T* и/или *A*, следствием чего может быть выполнение следующей когнитивной подзадачи и/или переход к выполнению прагматической задачи;
  - ◇ *оценочную*, когда адресат активирует или вырабатывает определенное отношение к *T* и/или *A* (при наличии прагматической задачи автору необходима позитивная оценка адресата относительно *A*, следствием чего может быть переход к выполнению прагматической задачи).
- *Иллокутивная* задача связана с передачей информации от адресата к автору для реализации следующих подзадач:
  - ◇ *информационная*, когда адресат передает автору информацию (в частности, сведения о себе), необходимую для реализации коммуникативной задачи автора;
  - ◇ *оценочная*, когда адресат передает автору свое отношение к *T* и/или оценку *A* (в частности, продукта *A*).
- *Прагматическая* задача состоит в том, что адресат совершает определенное некоммуникативное действие *A*′, связанное с *T* и встроенное в структуру *A* (заказ/покупка, пожертвование, подача заявки на участие и т. п.).

Коммуникативные задачи выстраиваются в иерархию, в которой реализация задачи более низкого уровня является предпосылкой выполнения задачи более высокого уровня. Наличие низкоуровневых реактивных задач (фатической, когнитивной) в иерархии коммуникативных задач характерно для любого коммуникативного акта. В зависимости от структуры задач, определяющей конечные цели и мотивы, все сайты по их назначению можно разделить на следующие три жанровые группы.

К первой группе относят ***информационные сайты***. Структура коммуникативных задач, которая характеризует сайты этой группы, включает фатическую и когнитивную реактивные задачи. Эти сайты нацелены на изменение знаний, мнений адресата по тематике *T*, исходя из предварительной общей гипотезы об интересе адресата к *T*. При этом корректировки представления о целевой аудитории не происходит, что сближает такой сайт с монологическим изложением.

Сайты для общения составляют вторую группу. Структура коммуникативных задач, которые решаются в этом случае, включает фатическую, когнитивную и иллокутивную реактивные задачи. Такие сайты направлены на организацию общения, что типично для блогов и форумов. Технологически они в наибольшей степени обеспечивают возможность диалогового общения в стиле устной коммуникации. Это позволяет участникам общения в реальном времени достраивать и корректировать свои представления друг о друге.

К третьей группе относят ***деловые сайты***. Структура коммуникативных задач в этой группе включает необходимость решения реактивных задач всех перечисленных типов. Такие сайты ориентированы на организацию совместной деятельности. Они занимают промежуточное положение в отношении адекватности представлений об адресате. В целом деловые сайты имеют монологический характер, но обеспечивают обратную связь с целевой аудиторией для уточнения некоторых аспектов, необходимых для успешной реализации целей деятельности *A*, которую обслуживает данный сайт. Как и в первом случае, образ адресата весьма обобщен, вследствие чего коммуникативный процесс характеризуется высокой степенью стереотипности. Особенностью деловой коммуникации является нацеленность не просто на информирование, но на создание условий

<table>
<tr><th colspan="5">Интерактивность</th></tr>
<tr><td colspan="2">Интерактивная</td><td colspan="3">Неинтерактивная</td></tr>
<tr><td colspan="2">Форум
Вопросы и Ответы
Форма поиска
Корзина заказов
Регистрационная форма</td><td colspan="3">FAQ [3]
About (страница/раздел о сайте, о Субъекте)
Главная страница
Контакты (Субъекта)
Статьи</td></tr>
<tr><th colspan="5">Динамичность</th></tr>
<tr><td colspan="2">Постоянный контент</td><td colspan="3">Изменчивый контент</td></tr>
<tr><td colspan="2">About
Главная страница
Контакты (Субъекта)
Статьи</td><td colspan="3">Форум
Вопросы и ответы
Комментируемые страницы
Новости</td></tr>
<tr><th colspan="5">Модальность</th></tr>
<tr><td>Текст</td><td>Статическое изображение</td><td>Видео</td><td colspan="2">Аудио</td></tr>
<tr><td>Статьи
Текст-портфолио</td><td>Фотогалерея
Фото до и после</td><td>Видеогалерея</td><td colspan="2">Аудио-библиотека</td></tr>
<tr><th colspan="5">Функциональность</th></tr>
<tr><td>Презентационная (Субъект)</td><td>Информативная (Тематика)</td><td>Информативная (Продукт)</td><td>Контактная</td><td>Директивная</td></tr>
<tr><td>About
Главная страница</td><td>Статьи
Галерея
FAQ
Новости
Специалисты</td><td>Каталог
Товар (Описание)
Услуга (Описание)
Прайс-лист</td><td>Контакты
Регистрационная форма</td><td>Вопросы и ответы
Корзина заказов
Регистрационная форма
Форма поиска</td></tr>
</table>

Рис. 2. Коммуникативные параметры жанровой классификации компонентов сайта

для реализации прагматической задачи. Именно этому подчинены самые разные компоненты структуры делового сайта.

1.2.2. Структурными составляющими сайта являются страницы, разделы (группы страниц), а также блоки в рамках одной страницы. Жанровая типизация блока/страницы/раздела возможна на базе таких коммуникативных параметров, как интерактивность, динамичность, преобладающая модальность, функциональность. На рис. 2 представлены коммуникативные параметры и их значения, а также приведены примеры соответствующих этим значениям жанровых типов страниц делового сайта.

Рассмотрим в качестве примера жанровый тип *Главная (Домашняя) страница*, присутствующий в составе большинства деловых сайтов. С точки зрения используемых свойств канала коммуникации страницы этого жанра отличаются постоянным контентом и, как правило, отсутствием интерактивных блоков. Страница может использовать различные сенсорные модальности канала коммуникации с преобладанием текста и статического изображения. Что касается параметра функциональности, то следует отметить, что многие компоненты сайтов полифункциональны. Так, *Главная (Домашняя) страница* представляет разные аспекты деятельности *A* без детализации, выделяя самые важные и обязательные (Субъект, Потребность), и дает ссылки на их детальные описания. Преобладающая презентационная функция страницы складывается из функций отдельных блоков.

2. Жанровая структура и жанровые маркеры

Жанру сайта соответствуют варианты структуры сайта, определяющие набор, расположение и гипертекстовые связи его компонентов (разделов, страниц и блоков). На уровне жанра отдельной текстовой страницы/блока можно говорить о наличии логико-композиционной структуры текста, соответствующей ее функциональности.

Логико-композиционная структура текста веб-страницы описывается как последовательность блоков, каждый из которых является текстовым фрагментом, представляющим некоторый *аспект содержания* — относительно самостоятельную часть текста, обладающую смысловым единством и соответствующую прагматике сайта *A* и/или его тематике *T*. Например, для сайтов жанра <*Деловой*, *Услуги*, *Институциональный*> прагматически ориентированная презентационная страница *About* (описание Субъекта деятельности *A*) может выглядеть как последовательность блоков[4] (рис. 3).

Каждый содержательный блок характеризуется собственной структурой, например, блок *Преимущества* часто представляется списочной структурой, элементы которой носят описательный характер, а блок *История* по композиции близок к рассказу и носит характер повествования.

При моделировании жанровой структуры текста веб-страницы авторы опираются на опыт работ в рамках так называемого индикаторного метода [11], разработанного для квазиреферирования текстов научно-технических статей — наиболее изученного канонического жанра. В соответствии с этим методом выявление аспектов содержания опирается на специальные словари неключевой внетематической лексики. В нашем подходе такая лексика описывается с помощью жанровых маркеров.

[3] Блок FAQ строится в виде множества вопросно-ответных структур, но, в отличие от блока *Вопросы и Ответы*, не обладает интерактивностью, так как создается разработчиками сайтов на основе предварительного анализа возможных информационных потребностей целевой аудитории.

[4] В работе [8] предложено строить лингвистическое определение жанров и их классификацию на основе исследования типов "пассажей" (т. е. блоков жанровой структуры текста).

```
< Page PageGenre = "About" >
    <Block type = "Вводная_часть" />
    <Block type = "История" />
    <Block type = "Преимущества" />
    <Block type = "Лицензии" />
    <Block type = "Кода (заключение)" />
</ Page >
```

Рис. 3. Логико-композиционная структура текста страницы *About*

Разрабатываемый лексикон *жанровых маркеров* содержит характеристические слова, словосочетания, шаблонные фразы, текстовые клише, определяющие тот или иной аспект содержания и границы представляющего его текстового блока. Например, аспект *Преимущества*, описывающий свойства *Субъекта услуг*, выгодно отличающие его от других, может быть представлен на странице делового сайта (*About* или *Главной*) с помощью маркеров "почему мы...", "наши преимущества...", "пять причин..." (рис. 4).

В общем случае маркер формализуется как лексико-грамматический шаблон, аналогичный описанным в работе [12]. Позиции шаблона заполняют словоформы, знаки, лексемы или лексико-семантические группы. Возможны позиции факультативные (квадратные скобки), вариативные (фигурные скобки), позиции, связанные грамматическими ограничениями с переменными определенного семантиче-

```
<Marker Term="почему мы">
    <Pattern> почему {мы [лучше] / именно мы / именно (наш X)}</Pattern>
    <UseVariable name="X" SemType="company_type" Morph="Сущ(им,ед)"/>
    <Condition type = "ЗАГ/АБ_нач" FragType = "СП_сл/Т_сл" />
    <Genre Parameter = "Деловой&Институциональный&Услуги " />
    <PageGenre Type="About/Главная" Block = "Преимущества " />
</Marker>
<Marker Term="наши преимущества-1">
    <Pattern> [наши] преимущества </Pattern>
    <Condition type = "TIT_вн" FragType = "T" />
    <Genre Parameter = "Деловой&Институциональный&Услуги" />
    <PageGenre Type = "About" Block = "Преимущества" />
</Marker>
<Marker Term="наши преимущества-2">
    <Pattern>[в чем] {наши преимущества/преимущества {работы/сотрудничества} с
                {нами/(наш X)}}
    </Pattern>
    <UseVariable name="X" SemType="company_type" Morph="Сущ(тв,ед)"/>
    <Condition type = "ЗАГ/АБ_нач/АБ_вн" FragType = "СП_сл/Т_сл" />
    <Genre Parameter = "Деловой&Институциональный&Услуги" />
    <PageGenre Type="About/Главная" Block = "Преимущества" />
</Marker>
<Marker Term="пять причин">
    <Pattern> (N причина) […] {купить/покупать} [именно] {у нас/в (наш X)}</Pattern>
    <UseVariable Name = "N" SemType =  "number" />
    <UseVariable name="X" SemType="company_type" Morph="Сущ(пр,ед)"/>
    <Condition type = " ЗАГ/АБ_нач/ЗАГ_вн/АБ_вн " FragType = "СП_сл/Т_сл" />
    <Genre Parameter = "Деловой&Институциональный&Торговля" />
    <PageGenre Type="About/Главная" Block = "Преимущества" />
</Marker>
```

Рис. 4. Жанровые маркеры аспекта содержания *Преимущества*

Рис. 5. Общая схема жанровой классификации интернет-сайтов

ского класса (для которых также могут быть указаны значения грамматических характеристик, согласование с определенными лексемами). Так, переменные класса *company_type* определяются такими лексемами, как *компания, фирма, центр, салон, магазин,* и должны быть согласованы по роду, числу и падежу с лексемой *наш*.

Каждому маркеру поставлены в соответствие презентационные условия: тип маркера (заголовок, абзац, титул страницы и др.) и позиция в нем (начало, конец, внутри); тип извлекаемого фрагмента, представляющего блок аспекта (список, следующий за маркером; текст, следующий за маркером до конца абзаца; текст, следующий за маркером до следующего маркера; весь текст страницы и др.).

Взаимосвязь жанровой модели веб-сайта с маркерами выражается посредством правил сопоставления маркеров (единичного маркера, набора или последовательности маркеров) элементам той или иной модели. Так, в приведенном примере маркерам соответствуют прагматические параметры сайтов, жанровые типы страниц и структурные блоки страниц.

3. Архитектура системы жанровой классификации веб-сайтов

Методы моделирования веб-страниц с помощью жанровых маркеров позволяют решать как задачу жанровой классификации интернет-ресурсов, так и задачу автоматизации генерации веб-страниц, например, в системах визуальной разработки веб-сайтов. В соответствии с принятым подходом, система жанрового анализа веб-сайтов опирается на текстовый контент веб-страниц, составляющих веб-сайт, а также на присущую html-формату представления веб-страниц разметку.

На рис. 5 представлена общая схема процесса жанрового анализа сайтов, которая включает четыре основных этапа:

- предварительный анализ сайта;
- словарный (лексический) анализ текстового контента сайта;
- жанровая типизация страниц сайта;
- жанровая типизация сайта.

Сайт поступает на вход системы в виде структурированного набора веб-страниц в html-формате (другие элементы сайта игнорируются). Модуль предварительного анализа осуществляет очистку страниц от рекламных элементов и формирует текстовое представление с сохранением необходимых для анализа элементов разметки, таких как название страницы, пункт меню, заголовок, ссылка, выделение текста, элемент списка.

Словарная подсистема осуществляет лексический анализ текста страниц сайта и поиск жанровых шаблонов на основе словаря маркеров. Найденные маркеры позволяют выделить жанровые фрагменты и осуществить анализ логико-композиционной структуры текста веб-страницы. Вывод (предположение) о жанре страницы делается на основе связанных с данным маркером значений параметров.

Жанровый рубрикатор хранит систему прагматических (праксиологических и коммуникативных) параметров, онтологию деятельности и описание моделей страниц и жанров сайтов.

Модуль жанровой классификации осуществляет анализ веб-сайта на макроуровне и на основе найденных маркеров выявляет прагматические параметры, описывающие жанр сайта.

Таким образом, результатом работы системы жанровой классификации является логико-композиционная структура текста каждой веб-страницы, входящей в состав сайта, а также набор параметров, характеризующих жанровые типы страниц и жанр сайта в целом.

Заключение

В статье предложен подход к формированию многоаспектной жанровой классификации веб-сайтов и их компонентов на основе прагматических параметров. К числу признаков, релевантных для идентификации жанра веб-сайта, отнесены значения параметров деятельностного контекста и структура коммуникативных задач. Дополнительные признаки, связанные с функциональностью и особенностями канала коммуникации, позволяют уточнить жанр структурной составляющей сайта (раздела, страницы или блока).

Жанру текстовой страницы соответствует характерная логико-композиционная (жанровая) структура, выявляемая с помощью лексикона жанровых маркеров. Метод основан на идентификации функциональных словесных клише, штампов, шаблонных фраз в исходных текстах корпуса веб-страниц и может быть применен как для выявления композиционной структуры текстов страниц различных жанровых типов, так и для генерации такой структуры, с ее последующим развертыванием и наполнением необходимой текстовой информацией.

Представленный подход к описанию интернет-коммуникации позволяет реализовать систему жанровой классификации сайтов на основе нетематической лексики с опорой на особенности представления интернет-текста. Результат жанровой классификации может быть использован для решения типичных задач анализа и синтеза текстового контента веб-сайтов: информационный поиск, тематическая классификация и фильтрация, извлечение информации, генерация сайтов и т. п.

Подход на основе жанровых маркеров реализован для задачи поиска интеллектуальных интернет-ресурсов [13]. Апробация предлагаемой системы классификации в более полном объеме осуществляется на материале сайтов деловой направленности и основывается на жанрово-тематическом рубрикаторе, включающем 108 тематических рубрик с группировкой по жанровым категориям (Услуги, Торговля и т. п.). Планируется расширить жанровую составляющую классификатора в целях охвата всех представленных в модели прагматических аспектов интернет-коммуникации.

Работа выполнена в Новосибирском государственном университете при финансовой поддержке Министерства образования и науки Российской Федерации (договор № 02.G25.31.0054).

Список литературы

1. **Herring S. C.** A Faceted Classification Scheme for Computer-Mediated Discourse // Language@Internet. 2007. Vol. 4. URL: http://www.languageatinternet.org/articles/2007/761 (дата обращения 15.03.15).
2. **Щипицина Л. Ю.** Жанры компьютерно-опосредованной коммуникации. Архангельск: Поморский университет, 2009. 238 с.
3. **Горошко Е. И., Жигалина Е. А.** Виртуальное жанроведение: устоявшееся и спорное // Ученые записки Таврического национального университета им. В. И. Вернадского. Серия "Филология. Социальные коммуникации". 2011. Т. 24 (63), № 1. Ч. 1. С. 105—124.
4. **Бахтин М. М.** Проблема речевых жанров // Эстетика словесного творчества. М.: Искусство, 1986. С. 250—296.
5. **Sorokin A., Katinskaya A., Sharoff S.** Associating Symptoms with Syndromes. Reliable Genre Annotation for a Large Russian Webcorpus // Computational Linguistics and Intellectual Technologies. Papers from the Annual International Conference "Dialogue". 2014. Iss. 13 (20). P. 646—658.
6. **Mehler A., Sharoff S., Santini M.** (eds.). Genres on the Web. Computational Models and Empirical Studies. Dordrecht: Springer, 2010. 362 p.
7. **Santini M., Mehler A., Sharoff S.** Riding the Rough Waves of Genre on the Web // Genres on the Web. Computational Models and Empirical Studies / A. Mehler, S. Sharoff, M. Santini (eds.). Dordrecht: Springer, 2010. P. 3—31.
8. **Кибрик А. А.** Модус, жанр и другие параметры классификации дискурсов // Вопросы языкознания. 2009. № 2. С. 3—21.
9. **Crowston K., Kwa'snik B., Rubleske J.** Problems in the Use-Centered Development of a Taxonomy of Web Genres // Mehler A., Sharoff S., Santini M. (eds.). Genres on the Web. Computational Models and Empirical Studies. Dordrecht: Springer, 2010. P. 69—84.
10. **Dong L., Watters C., Duffy J., Shepherd M.** An Examination of Genre Attributes for Web Page Classification // Proc. of the 41st Hawaii International Conference on System Sciences. 2008. P. 133—143.
11. **Блюменау Д. И., Гендина Н. И., Добронравов И. С.** и др. Формализованное реферирование с использованием словесных клише (маркеров) // Научно-техническая информация. Сер. 2. Информационные процессы и системы. 1981. № 2. С. 16—20.
12. **Большакова Е. И., Васильева Н. Э., Морозов С. С.** Лексико-синтаксические шаблоны для автоматического анализа научно-технических текстов // Десятая Национальная конференция по искусственному интеллекту с международным участием КИИ-2006, Т. 2. М.: Физматлит, 2006. С. 506—524.
13. **Загорулько Ю. А., Боровикова О. И., Сидорова Е. А., Ахмадеева И. Р.** Сбор онтологической информации для интеллектуальных научных интернет-ресурсов // Системная информатика. 2014. № 3. С. 13—23.

I. S. Kononenko, Researcher, e-mail: irina_k@cn.ru, **E. A. Sidorova**, Senior Researcher, e-mail: lena@iis.nsk.su, Novosibirsk State University, A. P. Ershov Institute of Informatics Systems, SB RAS

Genre Aspects of Websites Classification

The paper proposes an approach to the development of two-level genre models of communication on the Web. Genre category of a website as a whole (macro-level) corresponds to genre types of website constituents (pages and structural blocks) on the micro-level. Two types of pragmatic parameters are used to represent genres on both levels of consideration. Praxeological parameters, such as activity subject, beneficiary, product and environment, represent human activity that underlies communication and shows itself in the site structure, content and form of site constituents. Communicative parameters concern the hierarchy of communicative tasks, functionality of site constituents, and the affordances of communication channel. The type of anticipated reaction of the target audience in the hierarchy of communicative tasks forms the basis of website genre classification on the macro-level. Functions of site constituents together with channel features (interactivity, multimodality, and dynamics of content) determine genre types of web pages. The type of a textual page corresponds to genre schematic structure composed of content blocks. Identification of genre schemata as a succession of compositional blocks is possible due to a lexicon of genre markers (cue words and constructions) that are formalized as lexico-grammatical patterns provided with format conditions. The proposed genre models are realized in the website classification system.

Keywords: *web genre, website, web page, praxeological parameter, communicative parameter, genre schematic structure, genre marker, lexico-grammatical pattern, website genre model, website classification*

References

1. **Herring S. C.** A Faceted Classification Scheme for Computer-Mediated Discourse, *Language@Internet*, 2007, vol. 4, available at: http://www.languageatinternet.org/articles/2007/761 (accessed 15.03.2015).
2. **Shchipicina L. Ju.** *Zhanry komp'juterno-oposredovannoj kommunikacii* (Genres of Computer-mediated Communication), Arkhangelsk, Pomorskij university, 2009, 238 p. (in Russian).
3. **Goroshko E. I., Zhigalina E. A.** Virtual'noe zhanrovedenie: ustojavsheesja i spornoe. *Uchenye zapiski tavricheskogo nacionalnogo universiteta im. V. I. Vernadskogo. Seriya "Filologiya. Socialnye kommunikacii"*, 2011, vol. 24 (63), no. 1, part 1, pp. 105—124 (in Russian).
4. **Bahtin M. M.** Problema rechevyh zhanrov. *Jestetika slovesnogo tvorchestva*, Moscow: Iskusstvo, 1986, pp. 250—296 (in Russian).
5. **Sorokin A., Katinskaya A., Sharoff S.** Associating Symptoms with Syndromes. Reliable Genre Annotation for a Large Russian Webcorpus. *Computational Linguistics and Intellectual Technologies. Papers from the Annual International Conference "Dialogue"*, 2014, pp. 646—658.
6. **Mehler A., Sharoff S., Santini M**. (eds.) *Genres on the Web. Computational Models and Empirical Studies*, Dordrecht: Springer, 2010, 362 p.
7. **Santini M., Mehler A., and Sharoff S.** Riding the Rough Waves of Genre on the Web. *Genres on the Web. Computational Models and Empirical Studies* / Ed. A. Mehler, A. Sharoff, M. Santini. Dordrecht: Springer, 2010, pp. 3—31.
8. **Kibrik A. A.** Modus, zhanr i drugie parametry klassifikatsii diskursov. *Voprosy jazykoznanija*, 2009, no. 2, pp. 3—21 (in Russian).
9. **Crowston K., Kwa'snik B., Rubleske J.** Problems in the Use-Centered Development of a Taxonomy of Web Genres. *Genres on the Web. Computational Models and Empirical Studies*, Dordrecht: Springer, 2010, pp. 69—84.
10. **Dong L., Watters C., Duffy J. Shepherd M.** An Examination of Genre Attributes for Web Page Classification, *41st Hawaii International Conference on System Sciences*, 2008, pp. 133—143.
11. **Bljumenau D. I., Gendina N. I., Dobronravov I. S., Lahuti D. G., Leonov V. P., Fedorov E. B.** Formalizovannoe referirovanie s ispol'zovaniem slovesnyh klishe (markerov). *Nauchno-tehnicheskaja informatsija. Ser. 2. Informatsionnye protsessy i sistemy*, 1981, no. 2, pp. 16—20 (in Russian).
12. **Bol'shakova E. I., Vasil'eva N. E., Morozov S. S.** Leksiko-sintaksicheskie shablony dlya avtomaticheskogo analiza nauchno-tekhnicheskikh tekstov. *Materialy Desjatoj Nacional'noj konferencii po iskusstvennomu intellektu s mezhdunarodnym uchastiem KII-2006*, vol. 2, Moscow, Fizmatlit, 2006, pp. 506—514 (in Russian).
13. **Zagorul'ko Ju. A., Borovikova O. I., Sidorova E. A., Ahmadeeva I. R.** Sbor ontologicheskoi informatsii dlya intellektual'nykh nauchnykh Internet-resursov. *Systemnaya Informatica,* 2014, no. 3, pp. 13—23 (in Russian).

УДК 004.04

П. П. Олейник, канд. техн. наук, доц., зав. каф., e-mail: xsl@list.ru, Шахтинский филиал Московского государственного университета информационных технологий, радиотехники и электроники, Шахтинский институт (филиал) Южно-Российского государственного политехнического университета (НПИ) им. М. И. Платова, г. Ростов-на-Дону,
Н. Е. Бородина, студент, **Э. Г. Галиаскаров**, канд. хим. наук, доц.,
Ивановский государственный химико-технологический университет

Разработка информационной системы для проведения научных конференций

Проанализирован процесс реинжиниринга информационной системы, которая предназначена для проведения научных конференций и использовалась авторами многократно при проведении международных научно-практических конференций "Объектные системы" (objectsystems.ru). Методом обратного инжиниринга существующего программного кода получена модель имеющейся системы. В ходе ее анализа выявлены ошибки проектирования и предложены способы их устранения. С учетом выявленных недостатков и дополнительных требований разработана новая формальная модель системы, на основе которой с использованием платформы, поддерживающей подход Model Driven Architecture, реализована новая система. Все имеющиеся данные были сохранены и импортированы в новую систему. Представленный в статье процесс разработки информационной системы на основе модификации системы, используемой ранее, показал свою эффективность на практике.

Ключевые слова: *множественное наследование, реинжиниринг, UML, диаграмма деятельности, диаграмма классов, MDA, объектное проектирование, объектное моделирование, базы данных*

Введение

Одним из основных путей обмена научными знаниями, способом демонстрации научных достижений является их публикация в научных изданиях, представление на научных конференциях и семинарах. Наиболее доступным, динамичным и эффективным способом является участие именно в публичных научных событиях (симпозиумах, конференциях, совещаниях). На настоящее время существует огромное число научных конференций. Перед их организаторами возникает потребность эффективного управления процессами приема и регистрации присылаемых на конференцию научных публикаций, обработки большого объема дополнительных вспомогательных сведений организационного характера, включая данные о рецензии, оплате, отправленных сборниках и т. п. Удовлетворение этих потребностей требует разработки и использования информационных систем, автоматизирующих перечисленные процессы.

Данная статья отражает опыт авторов по разработке новой информационной системы каталогизации научных работ, основанной на реинжиниринге ее предшествующей версии, которая многократно использовалась при проведении конференции "Объектные системы" (objectsystems.ru). Далее под каталогизацией будем понимать весь жизненный цикл статьи: от получения первичной версии от автора и до принятия всех исправлений в соответствии с замечаниями рецензентов. В статье описаны недостатки предшествующей системы, которые стали причиной разработки новой системы, свободной от них. Полученная система уже была успешно протестирована при проведении последних сессий конференции "Объектные системы".

Недостатки существующей системы

В ходе подготовки к проведению первой сессии конференции "Объектные системы" возникла необходимость в использовании системы каталогизации научных работ. Изучив потребности в реализуемых ею механизмах, авторы сформулировали основные требования, которым должна удовлетворять такая система. Анализ ряда доступных на тот момент решений показал [1—5], что отсутствует система, которая бы полностью удовлетворяла предъявленным к ней требованиям. По этой причине, а также, с учетом того обстоятельства, что авторы имеют достаточный опыт в создании успешных программных систем, было принято решение о самостоятельной разработке. В результате было создано программное решение, получившее название SharpArchitect Scientific Conference Manager, функциональные возможности и отдельные модули которого подробно описаны в работах [6—8]. На протяжении нескольких лет предыдущая система достаточно успешно использовалась и решала возложенные на нее задачи. Вместе с тем в ходе эксплуатации системы был обнаружен ряд серьезных недостатков, затрудняющих ее дальнейшее использование.

Рассмотрим ключевые недостатки SharpArchitect Scientific Conference Manager, которые, по мнению авторов, присутствуют в предыдущей системе.

Во-первых, система SharpArchitect Scientific Conference Manager разрабатывалась в тот период, когда не были ясны все стоящие перед ней задачи, возникающие при проведении конференции. Не было ясно, какие функциональные возможности информационной системы потребуются в будущем. Основная задача, которая в то время решалась разработчиками, заключалась в оповещении вузов о предстоящих конференциях. В России более 2000 вузов, многие из них имеют профильные кафедры, которые необходимо оповестить. Именно сбор и сохранение информации об организациях и структурных подразделениях являлись наиболее важными задачами на тот момент. Чтобы удовлетворить соответствующую потребность в информационной системе был реализован программный модуль, позволяющий сохранять информацию об авторах и присланных на конференции статьях.

Во-вторых, следующим разработанным программным модулем стал модуль рецензирования. Отметим, что до начала проведения конференций у авторов статьи отсутствовал опыт организации процесса рецензирования. По этой причине было реализовано решение, которое в настоящее время имеет ряд недостатков. Присланная на конференцию статья может быть передана нескольким рецензентам, т. е. выполняется параллельное рецензирование. При этом каждый рецензент может не только написать рецензию, но и изменить исходный присланный автором файл со статьей и/или рисунками путем внесения корректировок. В результате возникает потребность учета этих изменений и формирования итогового файла на основании исправлений рецензентов и авторов. В текущей версии системы имеется возможность учета только одной рецензии по каждой статье. Поэтому невозможно отразить процесс параллельного рецензирования, что не соответствует текущему бизнес-процессу.

В-третьих, оказались важными функциональные возможности, позволяющие анализировать состояние статей ("на рецензии", "отклонена", "принята", "оплачена"). Реализация в предыдущей системе не отражала особенностей, связанных с наличием номинаций и возможностей получения авторами и членами комитетов скидок при оплате оргвзноса.

В-четвертых, при проведении конференции очень важна полнота и непротиворечивость вводимых данных. Это значит, что в момент ввода информации необходимо выполнить ее валидацию. Например, нельзя в систему добавить автора, не указав его фамилии, имени, отчества. Однако такую функцию можно реализовать в программном коде в момент сохранения данных с помощью написания набора из нескольких операторов if выбранного языка программирования. Для небольшой предметной области, для которой необходимо реализовать 25—30 валидационных правил, это наиболее простое с точки зрения программной реализации решение, и оно использовалось в предыдущей версии. По мере эксплуатации имеющейся системы число подобных ограничений значительно возросло, поэтому в программном коде накопилось большое число операторов if и вспомогательных методов. Добавление новой проверки приводило к значительному усложнению кода. Выходом стала реализация модуля декларативной проверки валидационых правил в новой системе. Такой подход позволяет описать требования к данным в виде строк с логическими предикатами. Он проще имеющегося в предыдущей версии и позволяет легко найти ошибки в созданных условиях. Поэтому реализация модуля валидации на настоящее время — приоритетная задача.

В-пятых, при эксплуатации предыдущей системы оказалось, что визуальное выделение строк данных по определенному критерию необходимо для упрощения анализа имеющейся в системе информации. Например, часто возникает потребность просмотреть уже принятые к публикации статьи, а также статьи, находящиеся на рецензии. На настоящее время эти функциональные потребности не реализованы.

Для разработки предыдущей версии системы использовалась среда Visual Studio. Поэтому добавление новых функциональных возможностей требовало перекомпиляции всего исполняемого файла. Для распространения изменений необходимо закрытие всех открытых ранее приложений и копирование очередной версии файла. Такой процесс разработки усложняет исправление найденных в программе ошибок. Гораздо удобнее иметь возможность модифицировать функциональные возможности программы в момент выполнения приложения без использования отдельной среды разработки. Именно такой подход использован в новой версии системы каталогизации научных работ, которая разработана в унифицированной среде быстрой разработки приложений баз данных SharpArchitect RAD Studio.

SharpArchitect RAD Studio — платформа реализации нового приложения

Используемая при разработке платформа SharpArchitect RAD Studio [10] основана на архитектуре, управляемой метамоделью объектной системы. Эта среда разработки не зависит от прикладной предметной области, а сам процесс реализации приложения заключается в корректном построении объектно-ориентированной модели выбранной предметной области. Проектирование начинается с анализа бизнес-процессов и выделения различных типов сущностей, которые могут быть отображены в виде диаграммы классов языка UML. При этом имеется соответствующий профиль, позволяющий помечать классы, атрибуты и ассоциации специальными стереотипами и проектировать в понятиях метамодели объектной системы.

Суть последующей разработки приложений в описанной среде заключается в создании экземпляров метаклассов. Метаклассы позволяют описать различные типы сущностей в зависимости от того, являются они сохраняемыми или нет, используются для вспомогательных целей или в виде параметров различных методов, позволяющих выполнить определенные вычисления. Добавление экземпляров метаклассов — это визуальный процесс, т. е. разработчику предоставляется множество графических форм, позволяющих вносить только допустимые значения. Затем в момент запуска приложения выполняется генерация программного кода на основе значений, присутствующих в метамодели, и регистрация полученных динамических библиотек, содержащих классы сущностей предметной области. Метаинформация используется также при генерации графического интерфейса пользователя. В результате используемая платформа имеет следующие преимущества:

- возможность автоматического создания интерфейса пользователя на основе модели приложения и настройки конечным пользователем;
- возможность с помощью редактора модели вносить изменения, а именно описывать классы, связывать их между собой в соответствии с UML-диаграммами и задавать сигнатуру методов в момент выполнения приложения, при этом для вступления изменений достаточно перезапустить приложение одного конкретного пользователя (остальные могут продолжить работу);
- возможность декларативного описания валидационных правил, позволяющих проверять корректность и непротиворечивость данных в момент сохранения их в базе данных;
- возможность декларативного описания визуализационных правил, позволяющих выделить с помощью изменения цвета различные графические элементы на форме, а также управлять видимостью и доступностью элементов.
- возможность с помощью множества системных (встроенных) классов упростить реализацию поведения часто используемых видов сущностей (например, в системе имеется внутренняя поддержка древовидных структур, которые используются при сохранении информации о территориальных объектах); а также возможность поддержки значений, которые актуальны на определенный период времени (например, оргвзнос может меняться при проведении различных конференций в разные годы).

В любом приложении баз данных основные типы сущностей, такие как Автор, Статья, Организация и т. п., являются сохраняемыми, поэтому создается экземпляр метакласса Класс предметной области (DomainClass). При генерации программного кода используется язык программирования C# и его синтаксическая конструкция "интерфейс" (*interface*). Таким образом, в некотором смысле поддерживается множественное наследование, что позволит упростить реализацию новой системы каталогизации и избавиться от имеющихся архитектурных недостатков, присутствующих в предшествующей реализации и проявляющихся в дублировании фрагментов кода.

Резюмируя представленные выше соображения, можно сделать вывод, что предшествующая система не отвечала современным требованиям. По этой причине появилась необходимость в разработке нового программного продукта, в качестве целевой платформы которого имело смысл выбрать SharpArchitect RAD Studio и выполнить реинжиниринг в понятиях метамодели объектной системы. В целях максимального сохранения информации наследуемой системы был проведен обратный инжиниринг и получена полная модель классов предшествующей системы. Чтобы не допустить ошибок проектирования было принято решение выполнить анализ бизнес-процессов и построить соответствующую диаграмму языка UML, позволяющую описать жизненный цикл присланной авторами статьи.

Анализ бизнес-процессов предметной области

Для получения ясной и полной картины того, что необходимо реализовать в новой информационной системе, следует проанализировать бизнес-процессы исследуемой предметной области. При проведении научной конференции можно выделить следующие три основных типа участников (роли) таких процессов.

- **Организатор конференции.** Является основным действующим лицом и пользователем системы. В его обязанности входит решение следующих задач:
  - ◇ регистрация публикаций;
  - ◇ назначение рецензента;
  - ◇ проверка исправлений в соответствии с замечаниями рецензента;
  - ◇ проверка оплаты;
  - ◇ верстка сборника;
  - ◇ отправка сборников и сертификатов авторам статей.
- **Автор.** Без него работа конференции невозможна, так как он пишет статью и присылает ее на конференцию. В обязанности автора также входит доработка статьи в соответствии с замечаниями рецензента на статью и, при необходимости, оплата организационного взноса.
- **Рецензент.** Он проверяет статью автора и оценивает ее качество. Рецензирование включает в себя: написание рецензии на статью, где указывают замечания и дают советы по ее улучшению; определение результата рецензирования — принять статью к печати, отклонить или отправить на доработку. В ходе верстки сборника конференции рецензентами присуждаются номинации лучшим статьям, присланным на конференцию. При этом в общем случае рецензентов несколько. Однако при графическом отображении роли на диаграмме этот факт не имеет значения и не выделен отдельно.

Процесс публикации статьи на конференции от момента ее подачи до момента получения сборника автором представлен в виде диаграммы деятельности UML на рис. 1.

На диаграмме изображены зоны ответственности (*partition*) для каждой выделенной выше роли участника. Такое представление позволяет наглядно показать передачу управления и продемонстрировать механизмы взаимодействия между участниками про-

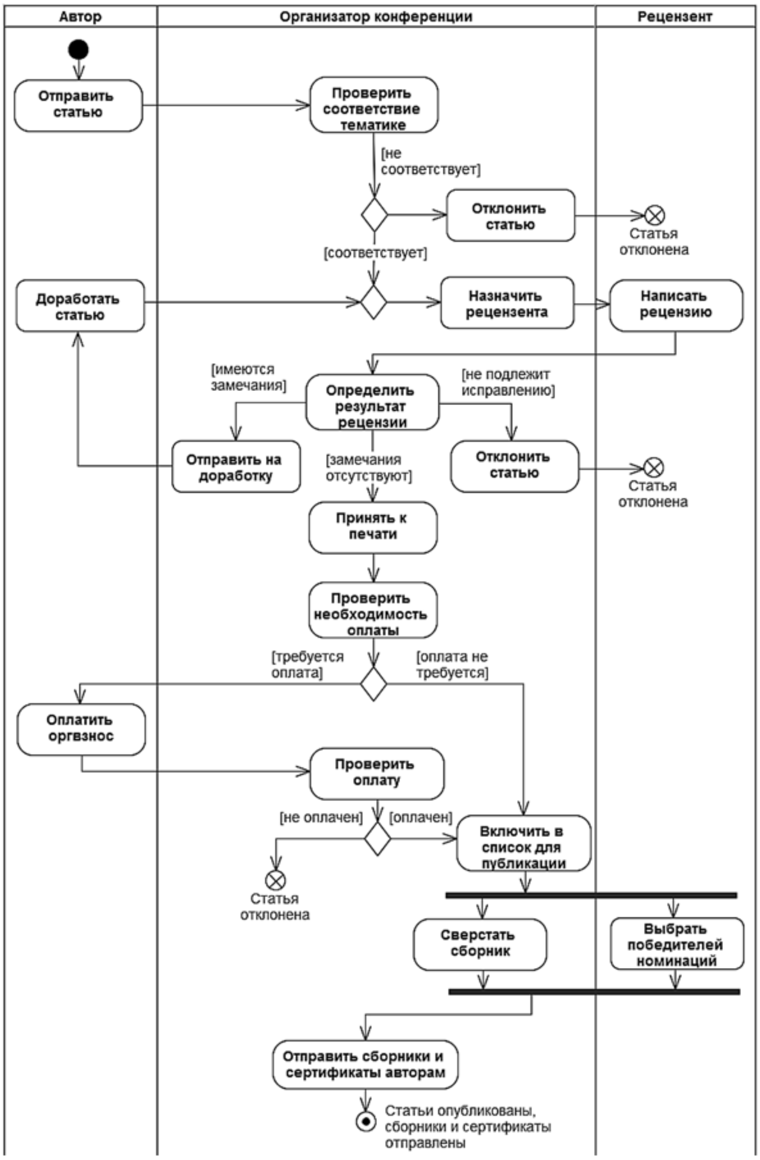


Рис. 1. UML-диаграмма деятельности предметной области

цесса. После выделения основных ролей необходимо описать функциональные требования, предъявляемые к разрабатываемому программному продукту.

Функциональные требования, предъявляемые к разрабатываемой системе

Функциональные требования определяют круг задач, которые необходимо автоматизировать. Эти требования частично были описаны ранее для предыдущей системы, а частично добавлены после проведения нескольких сессий конференции "Объектные системы". Для целевой системы были сформулированы следующие критерии, в краткой форме описывающие функциональные требования (ФТ):

ФТ1: реализация возможности сохранения информации о различных типах изданий с учетом того, что издание может быть собственным (сборником конференции, проводимой самостоятельно) или внешним (сборником конференции, проводимой сторонней организацией);

ФТ2: обеспечение возможности сохранения информации об авторах статей с указанием ученых степеней, занимаемых должностей в различных организациях;

ФТ3: реализация возможности сохранения информации об организациях, в том числе их типе, и подразделений организаций, в которых работают авторы;

ФТ4: предоставление возможности сохранения информации о рецензиях на статьи и их результатах;

ФТ5: реализация возможности сохранения информации о выпусках различных типов изданий, а также предоставления информации об этих выпусках пользователю;

ФТ6: реализация механизма сохранения информации о написанных определенными авторами статьях;

ФТ7: реализация возможности сохранения информации о персональных составах комитетов (программном и организационном) конференции;

ФТ8: реализация возможности сохранения информации о номинациях авторов статей;

ФТ9: представление возможности отслеживать рекомендации статей в другие издания, в частности, в журналы, рекомендованные ВАК РФ для публикации результатов диссертаций на соискание ученых степеней кандидата и доктора наук;

ФТ10: реализация в системе модуля аналитики, выводящего информацию о статьях, находящихся в определенных состояниях, отправленных письмах со сборниками и сертификатами, а также выпусках для участников комитетов;

ФТ11: реализация модуля декларативного описания валидационных правил, позволяющих предотвратить ввод некорректных данных;

ФТ12: предоставление возможности декларативного описания правил визуального выделения данных, удовлетворяющих определенным критериям.

Реинжиниринг предыдущей системы

Для исследования и анализа недостатков предыдущей системы каталогизации научных работ методом обратного инжиниринга на основе исходного кода была получена модель в виде набора взаимосвязанных диаграмм классов. Для этой цели использовался CASE-инструментарий проектирования программного обеспечения Enterprise Architect. Детально проанализировав полученную модель, авторы выявили перечисленные далее основные ее недостатки.

- В системе возможно сохранение информации только двух типов изданий, а именно — сборниках конференций и номерах журналов, подразделяющихся, в свою очередь, на собственные (выпускаемые собственным вузом) и внешние (выпускаемые сторонней организацией). При этом иерархия классов организована нерационально. По существу, имеются два вида издания, но в системе они представлены четырьмя классами: собственное и чужое издания, статьи конференции и журнала.

- Наличие большого числа программных реализаций одних и тех же интерфейсов, которые приходилось использовать для эмуляции множественного наследования, не поддерживаемого с помощью классов языка C#.
- Дублирование одинаковых атрибутов в разных базовых классах, которое объясняется отсутствием множественного наследования классов.

С учетом отмеченных недостатков было принято решение о реализации новой версии программы, ядро которой — архитектура MDA (*Model Driven Architecture*). Эта архитектура позволяет создать приложение, управляемое моделью. Одним из стандартов, используемым в MDA, является графический унифицированный язык моделирования UML, который изначально позиционировался как независящий от платформ и используемых технологий. Именно поэтому он используется для проектирования различных аспектов информационных систем. В контексте целей рассматриваемой системы было решено использовать только диаграммы классов, так как они позволяют отразить структуру приложения. При этом для программной реализации новой информационной системы использована унифицированная среда быстрой разработки корпоративных информационных систем SharpArchitect RAD Studio, созданная одним из авторов настоящей статьи [10].

Полученная модель иерархии классов нового программного продукта разбита на две части и представлена на рис. 2 и 3. На рис. 2 изображена диаграмма классов иерархии предоставляемых конференцией услуг. На рис. 3 (см. четвертую сторону обложки) представлена основная диаграмма классов.

При рассмотрении диаграмм видно, что в модели присутствуют классы с постфиксом Object. Классы Object — это абстрактные базовые несохраняемые классы, которые использованы в качестве контейнеров для представления атрибутов различных типов. Для обозначения абстрактного класса в языке UML наиболее распространено написание его имени курсивом. Так как построение диаграммы было выполнено в Microsoft Visual Studio 2012, а абстрактные классы реализованы в виде интерфейсов, то они отображаются регулярным шрифтом (без наклонов). Данную реализацию можно объяснить тем, что используется множественное наследование, а в среде C# это отношение можно эмулировать только интерфейсами.

Всего в модели используется 14 абстрактных классов. Каждый такой класс хранит в себе информацию о каком-то определенном атрибуте. При этом возможно наследование одного класса Object от другого.

Так как свойства и признаки атрибута определяются в абстрактном классе, его использование позволяет сократить время и облегчить процесс реализации производных классов. При таком подходе отсутствует дублирование кода. Отсюда следует, что при необходимости внести какие-либо изменения или при обнаружении ошибки в новой системе исправление будет происходить в одном определенном месте программы (в базовом классе).

В соответствии с функциональными требованиями, представленными ранее, а именно с ФТ1, а также по выявленным недостаткам предыдущей системы в данной модели был выделен новый класс EditionKind, определяющий тип издания. Его использование позволяет хранить в системе любые виды изданий (журналы, труды конференций, монографии и т. п.), не создавая при этом новых таблиц в базе данных, т. е. без модификации программного продукта. Вся информация об издании при этом хранится в одном классе BaseEdition. Так как издания могут быть собственными (производимыми самостоятельно) и внешними,

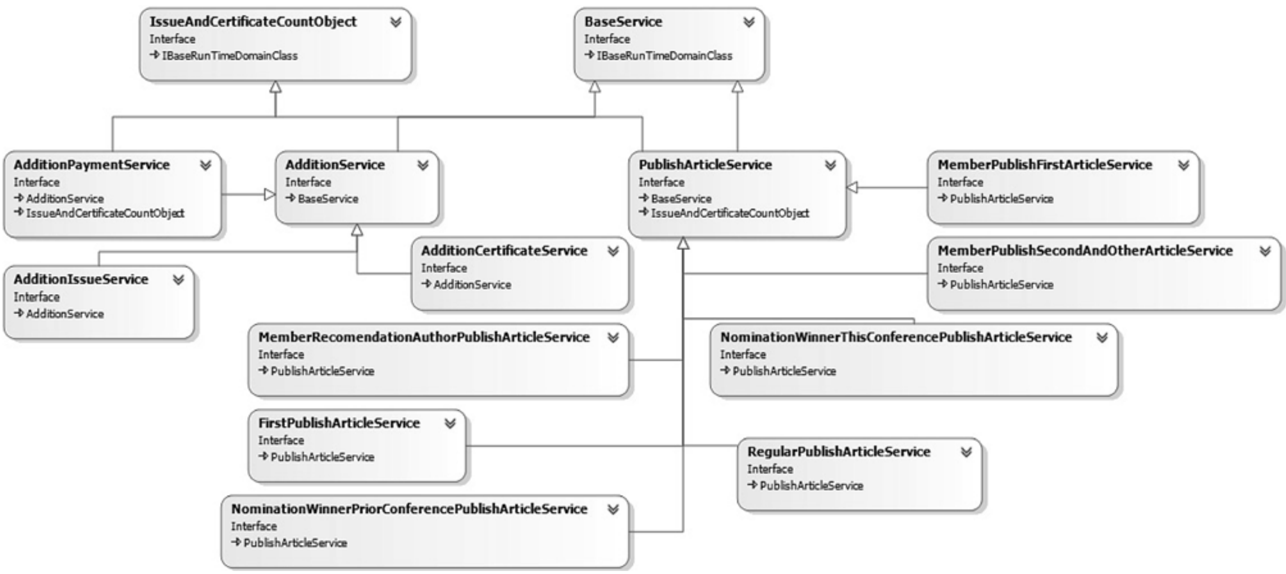


Рис. 2. UML-диаграмма классов иерархии предоставляемых услуг конференции

то существуют два класса, производных от базового класса — OwnEdition и ForeignEdition соответственно.

С отмеченными выше классами имеется связь у классов ForeignIssue и OwnIssue. Данные классы представляют собой сущности, хранящие информацию о выпусках изданий (номера журнала, сборники конференции). Аналогичным образом представлены внешний и собственный выпуски. Данная реализация обеспечивает выполнение ФТ5.

Как известно, в выпусках печатаются статьи, информация о которых также важна для пользователей системы (см. ФТ6). Для этого в модели реализованы такие классы, как ForeignArticle и OwnArticle.

Статьи на конференцию присылают авторы, поэтому в соответствии с ФТ2 следует хранить о них определенную информацию. Для выполнения данной возможности реализован класс Scientist — ученый. Для удобства хранения и использования в системе организованы классы должностей — Degree и ученых степеней — ScienceRank.

Для полноты информации об авторе необходимо также знать его места работы. Именно с этой целью реализован класс WorkingPlace — рабочее место. Данный класс получен в результате организации n-арной ассоциации. С его помощью легко находятся связи между автором (Scientist), организацией (Organization), ее структурным подразделением (Department) и должностью (Post). Для удобства работы с организациями выделен специальный класс OrganizationKind, хранящий информацию о типе организации. Данный класс также реализован в новой системе для удобства работы с ней. Дело в том, что организационно-правовая форма не уникальна в пределах одной организации, и каждый раз вводить одно и то же значение утомительно. Вся данная схема удовлетворяет ФТ3.

Рецензии на статьи выполняют ученые, входящие в определенные типы комитетов, что отражают ФТ4 и ФТ7. Для того чтобы узнать информацию о рецензии и ее авторе в системе реализованы классы Member — участник комитета и Review — рецензия. Данные классы связаны между собой, и таким образом организована связь рецензии и рецензента. Также для удобства использования организованы классы-перечисления, в которых предоставлены возможные варианты результата рецензии (ReviewResult), типа комитета (CommitteeType) и должности в комитете (CommitteePost).

За опубликованные статьи авторы могут быть признаны победителями в определенных номинациях, что соответствует требованию ФТ8. Для хранения данной информации в системе предусмотрен класс Nomination.

В соответствии с ФТ9, который гласит, что система должна предоставлять возможность отслеживания рекомендаций статей в другие издания, реализован класс RecommendationInSupportEdition. Указанный класс содержит в себе информацию о результате рекомендации, ее текст, а также дату сохранения результата.

При проведении конференции важно знать сводную аналитическую информацию, касающуюся статей, находящихся в определенных состояниях ("на рецензии", "отклонена", "рекомендована к печати"), а также отправленных письмах и выпусках, предназначенных участникам комитетов. Для удовлетворения ФТ10 в системе организован специальный модуль "Аналитика", который предоставляет пользователю всю отмеченную выше информацию.

Представленная схема организации иерархий новой системы и детальное описание присутствующих в ней классов подтверждают, что все перечисленные функциональные требования соблюдены.

После разработки новой модели было выполнено ее сравнение с моделью существующей системы. Необходимо было проверить, все ли классы отражаются в новой модели. Проверка показала, что реинжиниринг выполнен корректно и в полном объеме.

Так как данные, которые используются предшествующей системой, хранятся в реляционной базе данных, следующим этапом стала проверка новой базы данных на наличие эквивалентных таблиц, соответствующих таблицам старой базы данных, и присутствие в них всех необходимых столбцов.

Импорт данных

Разработка новой системы не означает, что работа в ней начнется с чистого листа, и данные, существующие в предшествующей системе, останутся только в ней. По этой причине следующим важным шагом стал импорт данных из старой системы во вновь разработанную.

Простое копирование данных было невозможно, так как возникли следующие сложности.

- Предшествующая система в качестве объектного идентификатора использовала значение типа guid, а новая — int.
- Модель новой системы предполагает наличие различных типов изданий, подразделяя их только на внешние и внутренние. В то время как в предыдущей системе было разделение на два типа изданий — журналы и материалы конференций, которые в свою очередь могут быть изданы как собственным вузом, так и сторонней организацией.

Учитывая последние сложности, перенести данные нужно было так, чтобы не потерять связи первичных/внешних ключей и соответствия между записями. Для выполнения этой задачи было написано большое число многотабличных сложных запросов на языке SQL. Заполнение таблиц базы данных происходило последовательно, начиная с простых таблиц, которые не ссылались на другие таблицы, с переходом затем к более сложным, содержащим отношения (связи) между таблицами.

Бизнес-логика и проверка расчетных значений

Выполнив SQL-запросы и добавив таким образом данные из предыдущей системы в новую, необходимо было проверить все ли записи успешно импортированы.

Для таблиц, значения которых хранятся в базе данных, решение данной задачи не составило особого труда. Проверка осуществлялась путем сравнения числа записей соответствующих таблиц различных систем.

Реализация вычисляемых атрибутов, используемых для расчета значений, потребовала написания бизнес-логики на языке C#. Ее правильность проверялась сопоставлением определенных значений предыдущей и новой систем. Так, если значения выбранных записей совпадали, следовал вывод о том, что реализация выполнена корректно.

Каждый класс состоит из определенного числа атрибутов. При этом каждый из них имеет свою степень значимости. Так, отсутствие значения одного атрибута никак не повлияет на функциональные возможности системы, а отсутствие другого наоборот сделает ее работу некорректной.

Таким образом, чтобы не допустить возникновения ошибочных противоречивых ситуаций при работе пользователя с системой, были определены следующие бизнес-правила.

- Обязательный атрибут. Данное правило определено для атрибутов, значения которых важны для корректной работы системы. Так, пользователь не сможет сохранить запись, если значение данного атрибута будет отсутствовать.
- Уникальное значение. Данное правило определяет "единственность" значения атрибута или комбинации значений. Система не позволит сохранить запись, если в определенных полях присутствуют значения, которые уже сохранены ранее для другой записи.
- Проверка соответствия значения или набора значений определенному предикату (логическому выражению).
- Отсутствие правила означает, что наличие значения данного атрибута возможно, но необязательно. Их отсутствие в системе никак не повлияет на ее работу.

Все типы описанных выше бизнес-правил были реализованы в новой системе в виде валидационных правил, описанных декларативно строками, содержащими предикаты.

Заключение

Представленный в статье процесс разработки информационной системы каталогизации научных работ, основанный на модели существующей системы, продемонстрировал свою эффективность на практике. Это достигнуто благодаря получению методом обратного инжиниринга модели с последующей ее трансформацией и реализацией в новой системе с использованием подхода MDA. Время, затраченное на разработку системы, с учетом неполной занятости трех участников проекта составило не более одного месяца. Новая система прошла апробацию в ходе проведения X Международной научно-практической конференции "Объектные системы — 2015", которая проводилась в мае 2015 г. в Ростове-на-Дону.

Дальнейшим развитием системы будет доработка бизнес-правил, необходимых для правильного функционирования системы, а также настройка графических форм пользовательского интерфейса. Новая система реализована в собственной среде разработки SharpArchitect RAD Studio, и добавление новых функциональных возможностей будет выполняться с помощью соответствующих диалоговых окон, что упростит и ускорит процесс разработки. В будущем планируется реализация web-приложения рассматриваемой системы и расширение числа взаимодействующих с ней лиц.

Список литературы

1. **Ackenhusen J. G.** Database management system for conference program preparation // IEEE Transactions on Professional Communication. 1989. Vol. 32, N 3. P. 194—199.
2. **Deponte J., Muller H., Pietrek G., Schlosser S., Stoltefuss B.** Design and Implementation of a System for Multimedial Distributed Teaching and Scientific Conferences // International Conference on Virtual Systems and MultiMedia, VSMIM'97. 10—12 Sep. 1997. Geneva, Switzerland, 1997. P. 156—165.
3. **Tu Yong, Peng Jie, Liu Runda, Zhao Hui.** Developing a Registration System for the Scientific Content Using DOI // Fourth International Conference on Cooperation and Promotion of Information Resources in Science and Technology, 21—23 November 2009. Beijing, China, 2009. P. 65—70.
4. **Gong Zhiping, Yang Xiaosong, Wang Shuo, Yang Yanhong.** Research on security structure of scientific and technical information system // 2012 (ICSAI), 2012 International Conference on Systems and Informatics. ICSAI 2012. 19—20 May 2012, Yantai, China, 2012. P. 2660—2663.
5. **Васенин В. А., Голомазов Д. Д., Ганкин Г. М.** Архитектура, методы и средства базовой составляющей системы управления научной информацией "ИСТИНА — Наука МГУ" // Программная инженерия. 2014. № 9. С. 3—12.
6. **Олейник П. П.** Опыт применения инструментов объектно-реляционного отображения при разработке информационной системы каталогизирования научных работ // Информационные технологии в науке, экономике и образовании: материалы 5-й Всероссийской научно-практической конференции. 2—3 сентября 2010 г. Алт. гос. техн. ун-т, БТИ. / Под. ред. О. Б. Кудряшовой. Бийск: Изд-во Алт. гос. техн. ун-та, 2010. С. 102—105.
7. **Олейник П. П., Игумнов Е. А., Свечкарёв Е. А.** Опыт проектирования информационной системы для каталогизирования научных работ при проведении международных конференций // Объектные системы — 2010: материалы II Международной научно-практической конференции. Россия, Ростов-на-Дону, 10—12 ноября 2010 г., Ростов-на-Дону, 2010. С. 48—51. URL: http://www.objectsystems.ru/files/Sertificates2010_2/Object_Systems_2010_Winter_Session_Proceedings.pdf
8. **Олейник П. П., Игумнов Е. А., Свечкарёв Е. А.** Реализация модуля рецензирования в информационной системе проведения научных конференций // Объектные системы — 2011: материалы III Международной научно-практической конференции. 10—12 мая 2011 г. Ростов-на-Дону / Под общ. ред. П. П. Олейника. Ростов-на-Дону, 2011. С. 26—29. URL: http://objectsystems.ru/files/Object_Systems_2011_Proceedings.pdf
9. **Бородина Н. Е., Олейник П. П., Галиаскаров Э. Г.** Опыт выполнения реинжиниринга объектной модели на примере информационной системы каталогизирования научных статей при проведении международных конференций // Объектные системы — 2014 (Зимняя сессия): материалы IX Международной научно-практической конференции, 10—12 декабря 2014 г. Ростов-на-Дону / Под общ. ред. П. П. Олейника. Ростов-на-Дону, 2014. С. 17—23. URL: http://objectsystems.ru/files/2014WS/Object_Systems_2014_Winter_session_Proceedings.pdf
10. **Олейник П. П.** Программа для ЭВМ "Унифицированная среда быстрой разработки корпоративных информационных систем SharpArchitect RAD Studio", свидетельство о государственной регистрации № 2013618212 от 04 сентября 2013 г.

P. P. Oleynik, Associate Professor, Head of Department, e-mail: xsl@list.ru, Shakhty Branch of Moscow State University of Information Technologies, Radioengineering and Electronics, Shakhty Institute (branch) of Platov South Russian State Polytechnic University (NPI), Rostov-on-Don, **N. E. Borodina**, Student, **E. G. Galiaskarov**, Associate Professor, Ivanovo State University of Chemistry and Technology, Ivanovo

Development of the Information System for Scientific Conferences

The article describes the process of developing a new information system that is used in International scientific-practical conference "Object systems" (objectsystems.ru). Model of the existing system was obtained by reverse engineering. Analysis of the resulting models allowed to reveal design issues and to determine how to resolve them. There were defined the optimality criteria which are the requirements for system functionality: the ability to store information about different types of editions; about the authors; about the organizations; about the review of the article and their results; about the issues of various types of editions; about the articles; about the committees and the authors who are committee participants; about the nominations of articles; the ability to track the recommendations of the publication of articles in other editions; analytics module that displays information about the articles that are in a certain state, about the packages with collections and certificates sent to authors and about the issues for committee participants. Given the detected issues and the additional requirements a new formal model has been developed. With platform Sharp Architecture RAD Studio, using the MDA approach, the new model has been transformed into a finished system. All available data were saved and imported into the new system. The information system development process discussed in the article has demonstrated its effectiveness and practicality.

Keywords: *multiple inheritance, reengineering, UML; activity diagram, class diagram, MDA, object design, object modeling, database*

References

1. **Ackenhusen J. G.** Database management system for conference program preparation. *IEEE Transactions on Professional Communication,* 1989, vol. 32, no. 3, pp. 194—199.
2. **Deponte J., Muller H., Pietrek G., Schlosser S., Stoltefuss B.** Design and Implementation of a System for Multimedial Distributed Teaching and Scientific Conferences. *International Conference on Virtual Systems and MultiMedia, VSMIM'97.* 10—12 Sep. 1997, Geneva, Switzerland, 1997, pp. 156—165.
3. **Tu Yong, Peng Jie, Liu Runda, Zhao Hui**. Developing a Registration System for the Scientific Content Using DOI. *Fourth International Conference on Cooperation and Promotion of Information Resources in Science and Technology.* 21—23 November 2009, Beijing, China, 2009, pp. 65—70.
4. **Gong Zhiping, Yang Xiaosong, Wang Shuo, Yang Yanhong**. Research on security structure of scientific and technical information system. *2012 International Conference on Systems and Informatics, ICSAI 2012*, 19—20 May 2012, Yantai, China, 2012, pp. 2660—2663.
5. **Vasenin V. A., Golomazov D. D., Gankin G. M.** Arkhitektura, metody i sredstva bazovoy sostavlyayushchey sistemy upravleniya nauchnoy informatsiey "ISTINA — Nauka MGU" (Architecture, Methods and Tools of the Intellectual System of Thematic Analysis of Scientific Information "ISTINA — Science in MSU" Core). *Programmnaya ingeneria*, 2014, no. 9, pp. 3—12 (in Russian).
6. **Oleynik P. P.** Opyt primeneniya instrumentov ob''ektno-relyatsionnogo otobrazheniya pri razrabotke informatsionnoy sistemy katalogizirovaniya nauchnykh rabot (Using of Object-Relational Mapping Tools for Development of Scientific Conference Managment Information System). *Informatsionnye tekhnologii v nauke, ekonomike i obrazovanii: materialy 5-y Vserossiyskoy nauchno-prakticheskoy konferentsii.* 2—3 September 2010, Alt. gos. tekhn. un-t, BTI / Ed. O. B. Kudryashova. Biysk: Izd-vo Alt. gos. tekhn. uni-ta, 2010, pp. 102—105. (in Russian).
7. **Oleynik P. P., Igumnov E. A., Svechkarev E. A.** Opyt proektirovaniya informatsionnoy sistemy dlya katalogizirovaniya nauchnykh rabot pri provedenii mezhdunarodnykh konferentsiy (Designing Scientific Conference Managment Information System). *Ob''ektnye sistemy — 2010, materialy II Mezhdunarodnoy nauchno-prakticheskoy konferentsii.* 10—12 November 2010, Rostov-na-Donu / Ed. P. P. Oleynik. Rostov-na-Donu, 2010. pp. 48—51, available at: http://www.objectsystems.ru/files/Sertificates2010_2/Object_Systems_2010_Winter_Session_Proceedings.pdf (in Russian).
8. **Oleynik P. P., Igumnov E. A., Svechkarev E. A.** Realizatsiya modulya retsenzirovaniya v informatsionnoy sisteme provedeniya nauchnykh konferentsiy. (Implementing of a Review Module in the Scientific Conference Managment Information System). *Ob''ektnye sistemy — 2011: materialy III Mezhdunarodnoy nauchno-prakticheskoy konferentsii*. 10—12 May 2011, Rostov-na-Donu / Ed. P. P. Oleynik. Rostov-na-Donu, 2011, pp. 26—29, available at: http://objectsystems.ru/files/Object_Systems_2011_Proceedings.pdf (in Russian).
9. **Borodina N. E., Oleynik P. P., Galiaskarov E. G.** Opyt vypolneniya reinzhiniringa ob''ektnoy modeli na primere informatsionnoy sistemy katalogizirovaniya nauchnykh statey pri provedenii mezhdunarodnykh konferentsiy. *Ob''ektnye sistemy — 2014 (Zimnyaya sessiya): materialy IX Mezhdunarodnoy nauchno-prakticheskoy konferentsii.* 10—12 December 2014, Rostov-na-Donu / Ed. P. P. Oleynik. Rostov-na-Donu, 2014, pp. 17—23, available at: http://objectsystems.ru/files/2014WS/Object_Systems_2014_Winter_session_Proceedings.pdf (in Russian).
10. **Oleynik P. P.** Programma dlya EVM "Unifitsirovannaya sreda bystroy razrabotki korporativnykh informatsionnykh sistem SharpArchitect RAD Studio" (Computer program "The Unified Environment of Rapid Development of Corporate Information Systems SharpArchitect RAD Studio"), the certificate on the state registration № 2013618212, 04 September 2013 (in Russian).

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

Сдано в набор 04.06.2015 г. Подписано в печать 17.07.2015 г. Формат 60×88 1/8. Заказ PI815
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**