ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г.

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

СОДЕРЖАНИЕ

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

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индексы: по каталогу агентства "Роспечать" — **22765**, по Объединенному каталогу "Пресса России" — **39795**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru 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.

УДК 004.414

В. Е. Гвоздев[1], д-р техн. наук, проф., зав. каф.,
Д. В. Блинова[1], канд. техн. наук, стар. препод.,
Н. И. Ровнейко[2], гл. специалист-эксперт,
О. П. Ямалова[3], нач. сектора,
[1] Уфимский государственный авиационный технический университет,
[2] Отделение пенсионного фонда РФ по Республике Башкортостан,
[3] Контрольно-счетная палата Республики Башкортостан,
e-mail: blinova.darya@gmail.com

Системное и структурное моделирование требований к программным продуктам и проектам

Рассматривается процесс формирования требований к программным проектам, описывается процесс формирования рамок требований программных проектов, приведены системная, структурная и иерархическая модели требований.

Ключевые слова: *требования к программным продуктам и проектам, рамки требований, системные и структурные модели требований*

Введение

Одной из главных причин провала или низкой эффективности реализации программных проектов является низкое качество требований к программным системам (ПС) [17, 19] либо некачественный перевод требований к программному продукту в требования к программному проекту [23]. Программный продукт должен обладать требуемыми потребительскими свойствами, программный проект должен обеспечить получение требуемого результата при ограниченных бюджете и сроках реализации. Несмотря на значительное число работ по тематике управления требованиями к программным продуктам и программным проектам, включая выявление правообладателей, извлечение требований, их согласование и структуризацию [4, 23, 26 и др.], задача создания эффективной системы формирования требований к программному продукту и встраивания таких требований в требования к программному проекту далека от окончательного решения.

Требования к программным продуктам и программным проектам обладают всеми признаками сложной системы. При исследовании сложных систем можно выделить признаки, инвариантные к природе системы (общие для сложных систем разных классов), а также признаки, характерные лишь для сложных систем определенного класса. Общими свойствами требований к программным продуктам и программным проектам являются:

- иерархичность организации, что позволяет соподчинять друг другу требования, относящиеся к различным стадиям реализации программных продуктов и программных проектов;
- слабая переносимость опыта, накопленного разными исполнителями при разработке требований, в том числе к программным продуктам аналогичного назначения;
- сложность прогнозирования хода формирования требований на достаточно большом промежутке времени, что обусловлено, во-первых, изменением представлений о ценностях разных правообладателей в ходе реализации программного проекта (явление интерсубъективности [14]), во-вторых, различными сочетаниями состояний внешней и внутренней сред в разных проектах;
- полиморфизм, означающий, что одному проекту в зависимости от целей исследования может быть поставлено в соответствие множество моделей требований, соответствующих основному, вспомогательному и обеспечивающему процессам в составе программного проекта;

- омнипотентность факторов, которая, например, выражается в том, что переход на следующую фазу жизненного цикла меняет цели управления и, соответственно, содержание требований к программному продукту и программному проекту;
- многомерность требований, что обусловлено разным видением программной системы разными правообладателями;
- изменчивость требований в пространстве и времени; изменчивость во времени обусловлена, во-первых, изменением характера работ, связанных с реализацией разных этапов жизненного цикла программных продуктов и различных стадий программного проекта, во-вторых, изменением представлений правообладателей о ценностях в ходе реализации программного проекта; изменчивость в пространстве обусловлена распределенным характером проектирования, т. е. выполнением разных работ в составе проекта на различных рабочих местах специалистами разного профиля;
- гетерогенность требований, что выражается в системном сочетании требований, характеризующих функциональные и нефункциональные свойства программного продукта; необходимость выполнения разных работ, связанных с реализацией основных, вспомогательных и обеспечивающих процессов проекта.

В настоящей статье рассматриваются системные модели, методологической основой построения которых является главный принцип управления сложными системами — принцип комплексности. Предлагаемые модели объединяют в единую систему требования к программным продуктам и программным проектам. Такие требования сформированы в рамках архитектурного подхода к реализации информационных систем [25].

Требования как предмет управления

В теории управления и принятия решений часто используется термин "объект управления". Как отмечается в работе [12], объект по определению не обладает активностью, т. е. способностью принимать решения. Для того чтобы подчеркнуть важное значение субъективной составляющей (потребностей, ожиданий, представлений о ценностях правообладателей) в процессе формирования требований, в дальнейшем используется понятие "предмет управления".

Основу процесса формирования требований как к программным продуктам, так и к программным проектам составляет обмен персональными знаниями [14] между специалистами и группами специалистов, с разных сторон видящих программную систему и процессы, входящие в состав программного проекта. При этом степень формализации действий, реализуемых разными специалистами, различна на разных стадиях проекта. В наименьшей степени она формализована на начальных стадиях (концептуальная стадия), значительно в большей степени — на стадии детального проектирования и конструирования ПС [19].

Управление формированием требований может рассматриваться как управление организационными системами. Такой подход обусловлен тем, что требования к программному продукту являются результатом управляемой дискуссии специалистов в различных предметных областях, задействованных в управлении бизнес-процессами и находящихся между собой в разных отношениях. В то же время, в силу уникального характера каждого проекта [13], механизм управляемого формирования требований к проекту, т. е. совокупность правил, законов, процедур, регламентирующих взаимодействие вовлеченных специалистов (правообладателей), также является уникальным. Наличие совокупности механизмов управления формированием требований к программным продуктам и программным проектам в определенной степени позволяет предсказывать поведение как исполнителей проекта, так и организации, выступающей в роли заказчика ПС. Таким образом, предсказуемость поведения как внешних, так и внутренних правообладателей относится к факторам снижения неопределенности программного проекта на разных стадиях его реализации (конус неопределенности [21]) за счет использования механизмов управления.

В литературе (например, в работе [12]) описаны следующие виды управления в социальных системах.

- **Управление составом**, применительно к задаче формирования требований — выбор состава правообладателей или, иными словами, субъектов извлечения требований.
- **Управление структурой**, применительно к формированию требований — определение полномочий и близости правообладателей к проекту [10], что определяет характер связей (формальных и неформальных) между правообладателями.
- **Информационное управление**, применительно к формированию требований — ограничения на функциональность и качество ПС и ее компонентов, определяемое средой использования системы, а также существующими системами аналогичного назначения — создаваемый программный продукт должен в каком-то смысле превосходить имеющиеся аналоги.
- **Мотивационное управление**, применительно к формированию требований — нахождение баланса интересов различных правообладателей, которые с разных сторон видят ПС [6, 7]. Например, обеспечение баланса между требованиями к надежности и ресурсами, доступными при разработке ПС [18], либо между областью применения ПС и требованиями к документации [19].
- **Институциональное управление**, по аналогии с работами [27, 28] в настоящей работе применительно к формированию требований под институционализмом понимается реализация положений стандартов (полезных практик), рекомендованных для стандартизированного описания требований к производственным процессам, ПС и их компонентам на разных этапах жизненного цикла.

Перечисленные методы управления присутствуют на всех стадиях жизненного цикла ПС — от обоснования целесообразности разработки продукта до вывода его из эксплуатации. Однако значимость видов управления различна на разных стадиях жизненного цикла программных продуктов и проектов. Так, в работе [19] отмечается, что существующие к настоящему времени международные и национальные стандарты не полностью и неравномерно покрывают объекты и процессы создания и применения ПС и их компонентов. Наиболее творческие и сложные процессы (системный анализ, проектирование, интеграция компонентов и систем, испытания и сертификация) создания и развития сложных и больших ПС почти не поддержаны рекомендациями стандартов вследствие недостаточной унификации и формализации их описания и разнообразия содержания. Вместе с тем мотивационный, информационный и институциональный подходы к управлению испытывают взаимные влияния и, подвергаясь конвергенции, присутствуют во всех системах управления программными проектами. Определяющим является соотношение методов, характерных для каждого подхода, в рамках конкретного проекта.

Рамки требований к программному проекту

Цель определения рамок требований состоит в создании основ построения системы управления формированием требований. Рамки требований определяются ответами на следующие вопросы.

- Что и почему мы хотим получить; как это повлияет на текущую действительность, нашу жизнь в дальнейшем и на какую временную перспективу? В представленной на рис. 1 модели ответы на эти вопросы даются в рамках компонента "предмет программного проекта и среда использования ПС".

Рис. 1. Модель рамок требований к программному проекту

- Какова ожидаемая сложность проекта; каков ожидаемый объем работ; насколько велик риск провала проекта?
- Как мы собираемся управлять реализацией проекта? При этом рассматриваются:
  - тип организации проекта — интегрированная, независимая, матричная;
  - организационная архитектура системы управления проектом — "лоскутное одеяло", сильная интеграция; слабая интеграция [25];
  - подходы к организации мониторинга хода проекта — вехи; параметры; частота и точность их контроля; метрики; ключевые индикаторы деятельности;
  - способы планирования проекта и его стадий; выявления причин отклонения фактических параметров от плановых; подходы к принятию решений и выработке мер по устранению отклонений;
  - подходы к управлению рисками проекта и др. вопросы, связанные с управлением проектом с учетом специфики создаваемого программного продукта, а также сочетание внешней и внутренней сред проекта.
- Какие ресурсы (информационные, технологические, организационные, временные, финансовые и др.) потребуются для реализации проекта?

В результате поиска ответов на перечисленные вопросы правообладатели (внешние и внутренние [10] с привлечением консультантов формируют компромиссные, приемлемые с точки зрения различных правообладателей, требования к функциональным возможностям и показателям качества программного продукта, и, соответственно, к содержанию и качеству результатов отдельных стадий программного проекта.

Исходными данными для определения границ требований являются следующие.

- Потребности, желания (ожидания) заказчика/пользователей. Потребности являются отражением текущего состояния предметной области применения ПС. Ожидания представляют собой прогнозируемое заказчиком/пользователем состояние предметной области после применения программного продукта.
- Существующие у заказчика информационные системы (так называемые "наследуемые системы" [20]), а также функциональные характеристики и характеристики качества существующих ПС аналогичного назначения. Особенностями наследуемых систем является, во-первых, то, что при их создании в недостаточной степени учитывалась возможность перенесения программных продуктов на новые платформы [20]. Во-вторых, они настолько важны для использующей их организации, что она не может отказаться от них [22]. К составными частям наследуемых систем относятся следующие.
  - Технические компоненты, обеспечивающие решение задач информационной поддержки бизнес-процессов организации (в виде аппаратной

и программной составляющих информационной системы).

- Информационный компонент, содержащий историческую информацию о наследуемой системе; персональные знания специалистов, обеспечивающих ее сопровождение. В работе [25] подчеркивается, что данные — самый инерционный компонент IT-архитектуры.
- Технологический компонент, в том числе включающий: профили стандартов [19]; инструментальные средства; процедуры (методики); языки программирования. В литературе [19, 22] подчеркивается неоднозначное влияние стандартов на качество ПС. С одной стороны, стандарты — это система многократно апробированных полезных практик, использование которых позволяет сократить затраты ресурсов и повысить качество ПС. С другой стороны, в силу того, что развитие стандартов отстает от развития технологий, они в какой-то степени являются препятствием к широкому использованию фактически доступных технологических возможностей.
- Организационный компонент, в состав которого помимо плана правообладателей [6] входит описание организационных интерфейсов и связей с внешними правообладателями. Классификация внешних правообладателей приводится, например, в работе [2].

Как отмечалось выше, управление процессом формирования рамок требований к программному проекту опирается на основные подходы к управлению социально-техническими системами, а именно — на институциональный, мотивационный, информационный [12]. Учитывая низкую степень формализации деятельности, связанной с формированием требований, и определяющее значение в этой деятельности творческой составляющей (в особенности на начальной стадии проекта), можно утверждать, что основными подходами к управлению процессом формирования требований являются мотивационный и информационный. Другими словами, основу формирования требований составляют персональные знания и представления правообладателей о ценностях [14].

Правообладатели и консультанты находятся в фокусе модели рамок требований. Факторами, определяющими качество требований, являются: состав правообладателей и консультантов; их компетентность; распределение между правообладателями полномочий по принятию решений, выделению и распределению ресурсов; заинтересованность ключевых правообладателей в успешной реализации проекта. Отличие консультантов от правообладателей состоит в том, что результаты проекта напрямую затрагивают интересы правообладателей, в то время как консультант лишь реализует некоторую функцию и его зависимость от результатов проекта значительно слабее [24].

Стрелки, связывающие компоненты модели (рис. 1), подчеркивают то обстоятельство, что формирование требований (как, впрочем, и реализация остальных видов деятельности в рамках программного проекта) носит итерационный характер. Цели, содержание, механизмы реализации компонентов и связей между ними зависят от этапа программного проекта.

Многомерная модель требований

В многочисленных литературных источниках, посвященных управлению программными проектами, подчеркивается важность построения моделей программных проектов и программных продуктов как одного из основных инструментов поддержки принятия решений, связанных с реинжинирингом системы информационной поддержки управления бизнес-процессами. Наличие целостной системной модели программного проекта, выступающей в качестве платформы взаимодействия правообладателей и разработчиков, является необходимым условием сбалансированного устойчивого развития программного продукта.

Многомерной модели требований ставится в соответствие системная модель следующего вида:

$$\langle\langle T^{(\Pi)}\rangle, \langle T^{(\Pr)}\rangle, \langle R\rangle, \langle M\rangle , \langle I\rangle, \langle S\rangle, \langle STR\rangle, \langle C\rangle\rangle .$$

Компонентами этой модели являются:

$\langle T^{(\Pi)}\rangle$ — система требований к ПС, обладающая свойствами, определенными в стандарте [7];

$\langle T^{(\Pr)}\rangle$ — система требований к программному проекту;

$\langle R\rangle$ — ресурсы, доступные для реализации проекта;

$\langle M\rangle$ — система управления проектом, предназначенная для решения всего комплекса задач, связанных с формированием требований как к программному продукту, так и к частям основного, вспомогательного и обеспечивающего процессов программного проекта;

$\langle I\rangle$ — система информационного обеспечения реализации основных, вспомогательных и обеспечивающих процессов, связанных с формированием требований;

$\langle S\rangle$ — правообладатели (внешние и внутренние), вовлеченные в проект;

$\langle STR\rangle$ — прямые и обратные связи между $\langle T^{(\Pi)}\rangle$, $\langle T^{(\Pr)}\rangle$, $\langle R\rangle$, $\langle M\rangle$, $\langle I\rangle$, $\langle S\rangle$;

$\langle C\rangle$ — множество интервалов времени, соответствующих разным стадиям реализации проекта.

Каждый из компонентов представленной системной модели, в свою очередь, сам является системой. Описанные далее структурные модели являются разными проекциями описанной системной модели. В приведенной модели в качестве отдельного компонента не выделен такой как "риск", хотя риск является неотъемлемым свойством любого проекта [13]. Это обусловлено тем обстоятельством, что риск можно рассматривать лишь применительно к определенному субъекту или объекту. В связи с этим предполагается, что риск является одним из атрибутов каждого компонента системной модели.

Структурные модели требований

Системам требований можно поставить в соответствие множество структур, акцентирующих внимание на различных особенностях программных продуктов и программных проектов. На рис. 2 приведена одна из структурных моделей — "пирамида требований". Особенностью этой модели является то, что в ней подчеркивается ключевая роль человеческого фактора (правообладателей) в формировании требований. "Пирамида требований" является частным случаем более общей модели — "пирамиды проекта", описанной в работе [16].

Состав компонентов модели обусловлен тем, что ключевыми факторами успеха при формировании требований к ПС и проектам являются следующие.

- Идентификации компетентных правообладателей, способных оказать влияние на ход проекта и обладающих адекватными знаниями и опытом: в предметной области (области использования программной системы); проектирования; реализации; испытания; развертывания и сопровождения; исключения ПС из эксплуатации. Создание эффективной системы управления правообладателями, позволяющей в том числе: своевременно выявлять конфликты интересов организаций, групп и субъектов, вовлеченных в проект; создавать условия, обеспечивающие эффективную деятельность правообладателей [11, 3, 1].

- Обеспечение соответствия требований к программному проекту требованиям к программному продукту. Реализация этого условия обеспечивает рыночную эффективность программного продукта, а также ресурсную и операционную эффективности проекта. При этом предполагается, что требования к программному продукту вытекают из целей бизнес-процесса, реализуемого организацией [25].

- Управляемость процесса формирования требований. Реализация этого условия предполагает, во-первых, наличие эффективной системы коммуникаций между правообладателями как инструмента нахождения компромиссных решений по спорным вопросам на основе обмена персональными знаниями [14], а, во-вторых, понимание объективных ограничений на качество продукта, которое может быть достигнуто при предполагаемом масштабе ПС, условиях использования и доступных ресурсах проекта [23, 17, 9, 5].

- Наличие соответствующих целям проекта технологических ресурсов, включая профили стандартов [19], а также инструментальных средств, позволяющих автоматизировать деятельность, связанную с формированием системы требований [26, 2, 8]. Реализация этого условия позволяет, во-первых, использовать полезные практики, описание которых представлено в виде нормативно-методических материалов, во-вторых, сократить время формирования качественных требований и их документирование с детальностью, адекватной целям и ограничениям проекта [19, 17, 7, 5, 4, 9].

Рис. 2. Пирамида требований

На рис. 3 представлена структурная модель, являющаяся проекцией многомерной модели требований и выражающая один из базовых принципов исследования сложных систем — принцип декомпозиции. Применительно к формированию требований этот принцип предписывает поэтапное преобразование требований более высокого уровня иерархии в требования подчиненного уровня иерархии. При этом требования, относящиеся к разным уровням иерархии, связаны по вертикали, а требования одного уровня иерархии — по горизонтали. Горизонтальные и вертикальные связи являются инструментом управления, создающим предпосылки для обеспечения таких свойств требований как трассируемость, полнота, корректность, непротиворечивость [7]. Содержание компонентов системной модели, а также горизонтальных и вертикальных связей меняются в зависимости от стадии проекта.

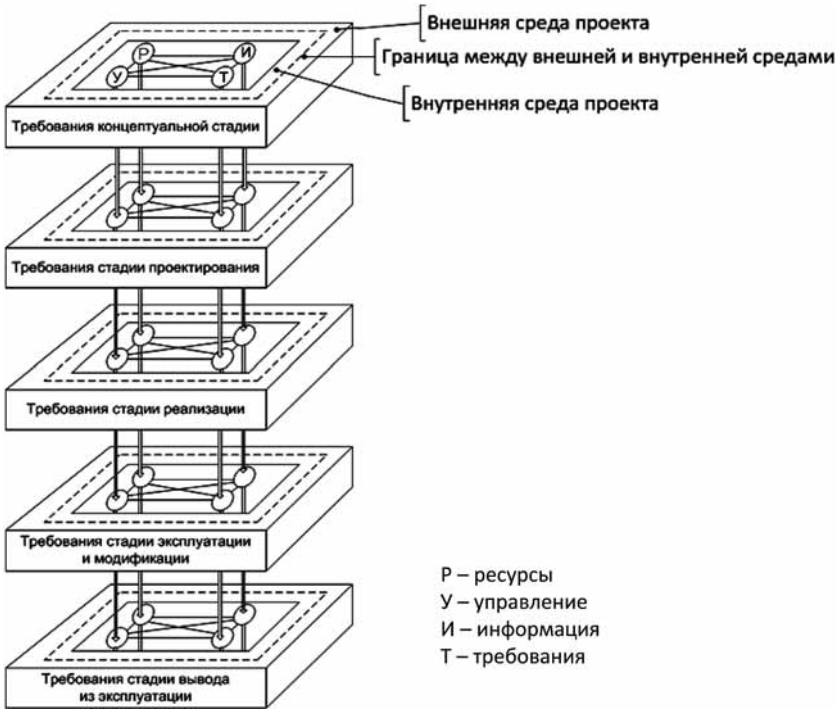


Рис. 3. Иерархическая модель требований

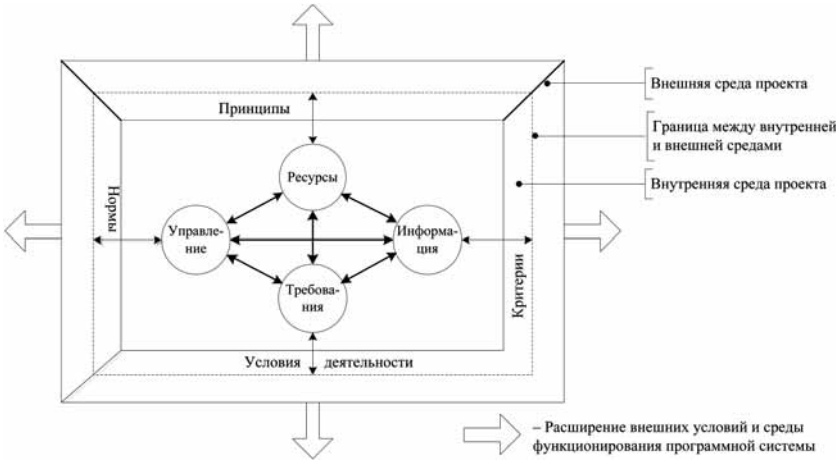


Рис. 4. Сечение иерархической модели требований

На рис. 4 представлено сечение иерархической модели требований. Эта модель подчеркивает взаимосвязь внешней и внутренней сред проекта. В ней (по аналогии с определением, данным в работе [15]) под ***внешней средой*** понимается совокупность внешних по отношению к программному проекту объектов (технических средств и ПС) и субъектов (внешних правообладателей, деловых процессов, нормативных документов и организаций), не входящих в границы проекта, изменения свойств и/или поведения которых влияет на проект; а также тех объектов и субъектов, свойства или поведение которых меняются в зависимости от результатов проекта.

Под ***внутренней средой*** понимается совокупность объектов (технических, инструментальных средств и иных ресурсов), субъектов (внутренних правообладателей [2]) и организаций, которые они представляют, объединенных в рамках выполнения деловых процессов, направленных на создание требований к программному продукту. Под организацией, следуя [12], понимается объединение людей, совместно реализующих некоторую программу или цель (в нашем случае — формирующих требования к программному продукту и программному проекту на разных стадиях их жизненного цикла) и действующих на основе определенных процедур и правил.

Границы между внутренней и внешней средами определяются перечисленными далее атрибутами.

- Критерии, которые, с одной стороны, характеризуют качество программной системы с точки зрения внешних правообладателей; с другой стороны — качество ПС с точки зрения внутренних правообладателей (примером нормативного документа, содержащего перечень таких критериев качества, является ГОСТ Р ИСО/МЭК 9126).

- Нормы, принятые в настоящее время в IT-индустрии. К нормам относятся стандарты, "де-юре" и "де-факто" аккумулирующие в себе полезные практики реализации сложных ПС. Сюда же относятся все правила, оформленные в виде официальных документов, а также "неписаные" правила, обеспечивающие решение эргономических, санитарных, правовых и других вопросов и задач, воз-

никающих в ходе реализации программных систем.

- Условия деятельности, а именно:
  - производственные (бизнес-процессы — внешние и внутренние);
  - информационные (внешняя и внутренняя информация, необходимая для организации и реализации проекта);
  - технологические (инструментальная и документальная поддержки проекта);
  - организационные (внутренняя упорядоченность и согласованность процессов и объектов проекта; направленность на получение требуемого результата при рациональных затратах ресурсов как внешних, так и внутренних правообладателей; объединение правообладателей, вовлеченных в проект и действующих на основе определенных процедур и правил).
- Принципы формирования требований. Целесообразно выделить три класса принципов: общесистемные принципы; принципы формирования требований к программным продуктам; принципы формирования требований к программным проектам. Выделение этих классов обусловлено следующим: во-первых, требования к программным продуктам и программным проектам являются разновидностью сложных систем, и, следовательно, имеют признаки, объединяющие их со сложными системами других классов; во-вторых, это отдельный класс сложных систем, следовательно, должны существовать признаки, отличающие требования к программным продуктам от требований к ПС, принадлежащим к другим классам; в-третьих, необходимым условием получения программного продукта с требуемыми характеристиками качества является управляемая реализация программного проекта; другими словами, должна быть сформирована система принципов реализации программного проекта, связанных с требованиями к реализуемому в его рамках программному продукту.

Рис. 5. Структура процесса определения границ требований

Структуры процессов формирования требований

На рис. 5 и 6 в качестве примеров представлены две структурные модели процессов формирования требований.

На рис. 5 представлена линейная структура процесса определения границ требований. В ней особое место занимают вопросы извлечения требований правообладателей, что служит основой сбалансированного учета интересов основных групп правообладателей, вовлеченных в проект.

В составе компонента "планирование общения с правообладателями" предполагается решение следующих задач:

- формирование целей, для достижения которых необходимо изучить мнения правообладателей;
- определение целей встреч правообладателей и критериев достижения цели обсуждений;
- определение контрольных дат встреч.

В составе компонента "изучение профиля правообладателей" предполагается решение следующих задач:

- определение ключевых правообладателей;
- оценка истинной и декларируемой мотивации вовлеченности их в проект;
- изучение структуры правообладателей.

В составе компонента "формирование тем и вопросов для обсуждения" предполагаются ответы на следующие вопросы:

- какая информация действительно нужна для реализации проекта?
- каким образом формулировать вопросы?
- каким образом оценивать значимость ответа?

В составе компонента "определение состава команды[1] извлечения требований" предполагается решение следующих задач:

- определение субъекта, управляющего обсуждением;
- определение длительности обсуждений;
- обучение участников обсуждений.

[1] Отличие команды от группы заключается в том, что группы выполняют четкие функциональные обязанности, а члены команды действуют сообща в целях достижения результатов, к которым стремится каждый из них [13].

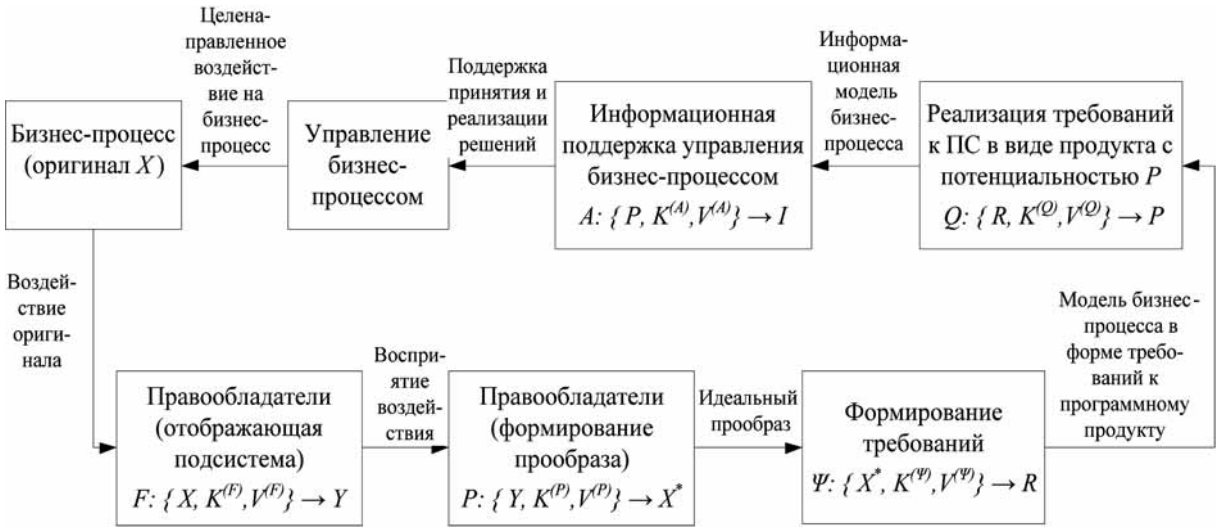


Рис. 6. Формирование и реализация требований к программным продуктам в системе управления бизнес-процессами

В состав компонента "подготовка к обсуждению" входят следующие действия:

- согласование сроков проведения обсуждений;
- согласование способа фиксации информации, получаемой в результате обсуждений.

В составе компонента "обсуждение целей продукта и проекта правообладателями" предполагается:

- изучение потребностей, пожеланий и ожиданий правообладателей;
- систематизация и интеграция информации, полученной всеми способами;
- формирование системы требований;
- формирование отчета и рассылка его ключевым правообладателям.

В составе компонента "встраивание требований в границы проекта" предполагается решение следующих задач:

- определение ключевых факторов успеха проекта;
- планирование интеграции ключевых факторов успеха в процессы и проектные продукты.

Процесс формирования требований можно рассматривать с позиций отображения оригинала (бизнес-процесса, протекающего в объекте управления) в программный продукт [25]. В основу структурной модели, представленной на рис. 6, положены следующие известные положения:

- цели создания ПС вытекают из целей бизнес-процессов, информационную поддержку управления которым она должна обеспечить;
- потенциальность ПС должна быть достаточной для решения основных задач, связанных с управлением бизнес-процессами;
- степень соответствия требований к ПС реальным потребностям системы управления бизнес-процессом определяется квалификацией (знаниями и компетентностью), а также представлениями о ценностях правообладателей, вовлеченных в программный проект.

В качестве отображающих системы выступают правообладатели, которые на основе анализа воздействий на них оригинала (бизнес-процесса) X формируют его образ Y. В результате сопоставления образа Y с ранее накопленными знаниями, информацией и данными, выявляются сходства и различия образа данного бизнес-процесса от образов иных оригиналов, наблюдавшихся в прошлом. Это служит основой обратного отображения X^*, т. е. формирования идеального образа бизнес-процесса. Идеальность образа состоит в том, что он содержит только существенные с точки зрения управления оригиналом сведения о бизнес-процессе, а также об ограничениях, обусловленных внешней и внутренней средами объекта управления. Идеальный образ служит основой для формирования требований к программному продукту. Символами $K^{(\bullet)}$ и $V^{(\bullet)}$ на рисунке обозначены знания и информированность правообладателей, задействованных на разных стадиях построения системы информационной поддержки.

Заключение

Рост сложности и масштабов программных проектов, тяжелые последствия стратегических ошибок, которые допускаются на ранних стадиях проекта и обусловлены низким качеством требований, необходимость учета противоречивых интересов разных групп правообладателей, вовлеченных в программный проект, ограниченные ресурсы программных проектов вынуждают разрабатывать эффективные системы управления программными проектами. Такая работа в том числе определяет необходимость разработки методологических, методических, модельных, инструментальных, информационных основ управления формированием качественных требований. Одной из задач построения системы управления формированием требований является разработка системных моделей,

позволяющих с единых позиций описывать требования разных групп правообладателей. Подобный подход открывает возможности сравнительного анализа требований, соотнесенных с разными процессами, входящими в состав программных проектов. Кроме того, унификация подходов к описанию требований делает возможным сравнение требований, относящихся к разным стадиям программного проекта. Разработка структурных моделей является предпосылкой построения математических моделей, что, в свою очередь, открывает возможности оптимизации процессов. Все это в совокупности увеличивает возможность формирования полных, сбалансированных, непротиворечивых, реалистичных, устойчивых требований.

Список литературы

1. **Boatright J. R.** What's Wrong — and What's Right — with Stakeholder Management // Journal of Private Enterprise. 2006. Vol. XXI. N 2. URL: http://www.apee.org/pdf/BoatrightSpec.pdf
2. **Bourne L.** Project Relationships and Stakeholder Circle^{TM}, PMI Research Conference, 2006. URL: http://www.stakholder-mangment.com/lopers/P010_Project_Relationships.pdf
3. **Capability** Maturity Model Integration (CMMISM), Version 1.1 CMMISM for Software Engineering Staged Representation CMU/SEI-2002-TR-029 SC-TR-2002-029, 2002.
4. **Christel M. G., Kang K. C.** Issues in Requirements Elicitation. Technical Report CMU/SEI − 92-TR-012 ESC-TR-92-012, September 1992.
5. **CobiT 4.I.** Российское издание. М.: Аудит и контроль информационных систем, 2008. 234 с.
6. **IEEE** Guide to the Software Engineering Body of Knowledge. SWEBOK, 2004.
7. **IEEE** Recommended Practice for Software Requirements Specifications IEEE STD 830-1998, June 1998.
8. **Olander S., Landin A.** Evaluation of stakeholder influence in implementation of construction projects // International Journal of Project Management, 23 (4): 321. doi 10.1016/j.ijp roman.2005.02.002.
9. **System** Engineering for Intelligent Transportation Systems / An introduction for Transportation Professionals. January 2007, 109 p.
10. **Walter D., Shelley A., Bourne L.** Influence, Stakeholder Mapping and Visualization. URL: http://journalsonline.tangf.co.uk/
11. **Yang J., Qiping Shen G.** Exploring critical success factors for stakeholder management in construction projects // Journal of civil engineering and management. 2009. № 15(4). P. 337—348.
12. **Бурков В. Н., Коргин Н. А., Новиков Д. А.** Введение в теорию управления организационными системами. М.: Книжный дом "ЛИБРОКОМ", 2009. 264 с.
13. **Бэгьюли Ф.** Управление проектом. Пер. с англ. В. Петрашек. М.: ФАИР ПРЕСС, 2002. 208 с.
14. **Виттих В. А.** Организация сложных систем. Самара: Самарский научный центр РАН, 2010. 66 с.
15. **Волкова В. Н., Денисов А. А.** Основы теории систем и системного анализа. СПб.: Изд-во СПб ГТУ, 1999.
16. **Гвоздев В. Е., Ильясов Б. Г.** Пирамида программного проекта // Программная инженерия. 2011. № 1. С. 16—24.
17. **Липаев В. В.** Анализ и сокращение рисков проектов сложных программных средств. М.: СИНТЕГ, 2005. 224 с.
18. **Липаев В. В.** Надежность программных средств. Серия "Информатизация России на пороге XXI века". М.: СИНТЕГ, 1998. 232 с.
19. **Липаев В. В.** Процессы и стандарты жизненного цикла сложных программных средств. М.: СИНТЕГ, 2006. 276 с.
20. **Липаев В. В.** Технико-экономическое обоснование проектов сложных программных средств. М.: СИНТЕГ, 2004. 284 с.
21. **Макконнелл С**. Сколько стоит программный проект. СПб.: Питер, 2007. 297 с.
22. **Мацяшек Л. А.** Анализ требований и проектирование систем. Разработка информационных систем с использованием UML. М.: Вильямс, 2002. 432 с.
23. **Милошевич Д.** Набор инструментов для управления проектами. М.: ДМК Пресс, 2008. 729 с.
24. **Томпсон А. А., Стрикленд А. Дж.** Стратегический менеджмент. Концепции и ситуации для анализа. М.: Вильямс, 2007. 928 с.
25. **Учебник 4CIO.** Версия 1.0. М.: 4CIO, 2011. 383 с.
26. **Халл Э., Джексон К., Дик Д.** Разработка и управление требованиями. Пер. с англ. И. Корнипаева. Telelogic, 2005. URL: http://inethub.olvi.net.ua/ftp/pub/kpi/selena/documentation/techicallibrary/programming/разработка и управлениетребованиями.pdf
27. **www.aup.ru/books/m 172/4_1.htm**
28. **www.bibliotekar.ru/istoria-economicheskih-ucheniy-3/23.htm**

ИНФОРМАЦИЯ

14—17 октября 2013 г. в г. Ярославль состоится

XV Всероссийская научная конференция RCDL'2013
"Электронные библиотеки: перспективные методы и технологии, электронные коллекции"

Серия Всероссийских научных конференций RCDL, труды которых представлены на сайте http://rcdl.ru, нацелена на формирование российского корпуса международного сообщества ученых, развивающих это научное направление, подробно описанное на сайте конференции.

Совместно с конференциями традиционно проводятся сопутствующие диссертационные семинары, на которых авторам работ, отобранных на основе предварительного рецензирования, предоставляется возможность изложить текущие результаты своих исследований, а также обсудить их сильные и слабые стороны с более опытными коллегами.

Труды конференции буду опубликованы в виде сборника текстов принятых полных статей, кратких статей и тезисов стендовых докладов, а также в электронном виде в европейском репозитории трудов конференций CEUR Workshop Proceedings. Лучшие статьи, представленные на конференцию, будут рекомендованы к публикации в изданиях, признанных ВАК, таких как "Информатика и ее применения", "Программная инженерия", "Системы и средства информатики".

Подробности — на сайте конференции: http://rcdl2013.uniyar.ac.ru

УДК 004.8

М. А. Колосовский, аспирант, e-mail: maxim.astu@gmail.com,
Е. Н. Крючкова, канд. физ.-мат. наук, проф., e-mail: kruchkova_elena@mail.ru,
Алтайский государственный технический университет им. И. И. Ползунова, г. Барнаул

Настройка параметров алгоритма сегментации изображений QuickShift

Данная работа посвящена алгоритму сегментации изображений QuickShift. Цель исследования — выявить, как влияют значения параметров этого алгоритма на результаты сегментации. В результате экспериментов была вычислена оценка качества сегментации при различных значениях этих параметров, было выявлено, в каких случаях лучше всего работает выбранное значение параметра и составлен ряд рекомендаций по установке параметров алгоритма.

Ключевые слова: *сегментация изображений, QuickShift, суперпиксели*

Введение

Сегментация изображения означает его разбиение на множество областей. Цель многих задач анализа изображений заключается в сегментации на области, с которыми связана существенная для данной задачи информация. Например, при обработке спутниковой фотографии областями интереса могут быть области городских построек, лесных массивов и сельскохозяйственных полей [1]. Задача усложняется, когда нужно не просто понимать тип местности, но и выделять конкретные объекты: здания; автомобили; людей и др. Такая сегментация, выделяющая целые объекты на изображении, называется *семантической.*

Для выполнения семантической сегментации удобно перейти от пиксельного представления к более высокоуровневым структурам — суперпикселям. *Суперпикселями* называют области изображения, объединяющие пиксели с похожими признаками (интенсивность, цвет, текстура и др.). Таким образом, осуществляется переход к более компактному и удобному представлению изображения. Представление будет тем компактней, чем более крупные суперпиксели выделяются и, следовательно, меньше будет их общее число. Вместе с тем качественное разделение на суперпиксели не должно приводить к объединению в один суперпиксель фрагментов нескольких объектов, что ограничивает размеры суперпикселей сверху. Таким образом, хорошая сегментация — это компромисс между учетом всех деталей изображения и компактностью представления. Метод выделения суперпикселей (пересегментация) как начального этапа семантической сегментации, впервые предложенный в работе [2], широко используется в настоящее время [3, 4].

На данный момент известно множество алгоритмов выделения суперпикселей. Однако с какой бы стороны не рассматривалась задача сегментации, существует необходимость настройки параметров алгоритма в зависимости от изображений. Рассмотрим как это происходит при различных подходах к сегментаци:

- **Методы на основе краев [5—7].** При имеющейся карте краев изображения остается лишь найти связные области, однако карта краев не определяется однозначно для данного изображения. В общем случае нужно задавать порог, определяющий, насколько сильные изменения изображения считать краем. Если порог слишком мал, то карта краев будет зашумлена несущественными границами, а если слишком высок, то не все края объектов найдутся. Кроме того, алгоритмы обнаружения краев обычно требуют предварительного применения различных фильтров, параметры которых также зависят от изображения.
- **Эвристические методы.** К ним относятся:
  - разрастание регионов [1] (требует задания условий продолжения разрастания региона);

- метод разделения и слияния [8] (необходимо задать условие неоднородности региона (при котором регион разделяется));
- алгоритм водораздела [9] (необходимо предварительное применение фильтров к изображению).

- **Методы на графах**, такие как Normalized Cut [10] и метод Felzenszwalb & Huttenlocher [11], разбивают изображение, используя более сложный критерий, чем эвристические методы. Однако в них также определяется порог, при котором следует остановить разбиение области.
- **Методы на основе кластеризации.**
  - метод *k*-средних вынуждает оценивать количество кластеров, на которые разобьется множество пикселей. Кроме того, результат может сильно варьироваться в зависимости от начальной инициализации центров кластеров.
  - методы MeanShift [12], MedoidShift [13] не содержат параметров чувствительных к содержанию изображения.
  - метод QuickShift [14] требует задания максимальной разности между признаками разделяемых областей.
- **Энергетические методы:** Snakes [15]; методы уровня [16]; метод соревнования областей [17]; TurboPixels [18]. В этих методах нужно задавать число семян (из которых будут сформированы регионы), а также регулировать параметры схождения контуров.

Резюмируя перечисленное выше можно выделить следующие общие проблемные вопросы, которые возникают при применении алгоритмов сегментации:

- критерий необходимости разбиения региона и метод нахождения границы;
- критерий для объединения региона;
- предположение о числе регионов;
- неоднозначность карты краев;
- предварительное применение фильтров.

Эти вопросы в разной степени характерны для всех алгоритмов сегментации. В данной работе в качестве примера адаптации алгоритмов сегментации рассмотрим алгоритм QuickShift, что объясняется следующим. В настоящее время одним из лучших алгоритмов сегментации является MeanShift [28], но модификация этого алгоритма QuickShift почти не уступает MeanShift по качеству и работает на порядок быстрее [14]. При проведении экспериментов использована конкретная реализация QuickShift [19], распространяемая в открытой библиотеке VLFeat [20].

QuickShift относится к методам кластеризации на основе плотности вероятности (*probability density function*). Центрами кластеров являются локальные максимумы плотности, а точки, "поднимающиеся" к данному пику, составляют кластер. Плотность оценивается методом окна Парзена (*Parzen or Parzen-Rosenblatt window*), также называемым ядерной оценкой плотности (*kernel density estimation*). Плотность точки пространства — это сумма вкладов $K(d_i)$ от элементов выборки, где K — функция, называемая ядром, d_i — расстояние от точки до i-го элемента. Учитываются элементы, находящиеся ближе заданного порога (*ширины окна*) [21].

Алгоритм требует эмпирического задания ряда параметров (констант, функций, наборов признаков), необходимых ему для его работы. Введение таких параметров вызвано желанием ускорить работу этого алгоритма, производного от более медленного и более точного алгоритма MeanShift за счет принятия некоторых предположений о работе алгоритма. Так как зачастую предположение содержит некий количественный параметр, максимальное эффективное значение которого невозможно строго доказать, эти предположения появляются в алгоритме в виде параметризованных эвристик. Отрицательной стороной такого ускорения алгоритма является сильная зависимость качества работы алгоритма от этих параметров, наилучшие значения которых могут варьироваться в зависимости от конкретного изображения. Рассматриваемая реализация QuickShift имеет следующие явные параметры:

- компактность цветов суперпикселя — параметр *ratio*;
- размер окна Парзена для оценки плотности распределения пикселей — *kernelsize*;
- максимальная длина ребра в дереве — *maxdist.*

Задачей рассматриваемой в настоящей статье работы является исследование зависимости поведения алгоритма от различных параметров. Кроме явных параметров алгоритма, авторами были выявлены и другие точки возможного изменения алгоритма, текущее значение которых не является однозначно оптимальным, а именно:

- используемые признаки пикселя (пространственные, цветовые, текстурные и т. д.) — в описываемой реализации используются координаты пикселя на изображении и цветовые компоненты в пространстве *Luv*;
- метод разделения дерева в лес суперпикселей — удаление всех ребер длиннее чем параметр *maxdist*;
- функция ядра оценки плотности — гауссово ядро;
- метрика расстояния в пространстве признаков — евклидова.

Был проведен ряд экспериментов с различными значениями перечисленных параметров, выявлены

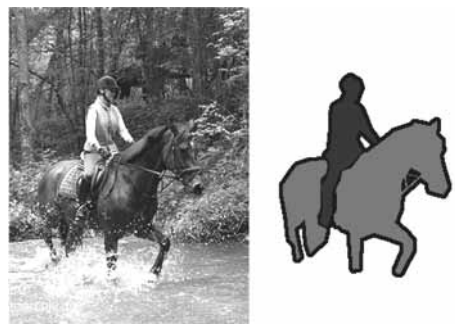


Рис. 1. Изображение и его эталонная сегментация

зависимости поведения алгоритма от этих параметров. Эксперименты проводились на базе изображений Pascal VOC 2007 [22]. Эта коллекция содержит 422 изображения, для которых имеется эталонная сегментация, где каждый объект и фон обозначены своим цветом, а также выделена граница между областями (рис. 1).

Оценка качества сегментации

Оценка качества сегментации — вопрос крайне неоднозначный, в связи с чем существует колоссальное число метрик сегментации: на основе пересечения регионов, совпадения границ, статистические методы и метрики теории информации [23—25]. В данной работе воспользуемся метрикой, производной от индекса Ранда [26], измеряющего похожесть двух разбиений на кластеры и применяемого, в частности, для оценки семантической сегментации [24, 27, 28]. Непосредственное применение этого индекса к оценке построенных суперпикселей одинаково штрафует разбиение объекта на несколько суперпикселей и пересечение суперпикселем нескольких объектов, что будет ухудшать точность выделения контуров. Предлагается уделить большее внимание точности выделения контуров, т. е. второй компоненте, так как выделение объектов целиком не предусматривается на этом этапе. Представляемая метрика будет считать ошибку сегментации. Кроме того, вычисление индекса Ранда существенно более затратно с вычислительной точки зрения. Отрицательной стороной такого перехода к новой метрике является то, что она не учитывает компактность представления, т. е. число суперпикселей. Поэтому одновременно с метрикой вычисляется *средний размер суперпикселей*. Таким образом, если с увеличением качества сегментации не уменьшается размер суперпикселей, то такая сегментация действительно лучше.

Оценивать качество сегментации будем числом пикселей, относящихся к разным объектам, но попавшим в один суперпиксель (рис. 2). Штраф для j-го суперпикселя F_j равен:

$$F_j = \Sigma \frac{n_i}{n}(n - n_i),$$

где n_i — число пикселей, попавших в j-й суперпиксель и относящихся к i-му объекту, $n = n_1 + n_2 + ... + n_k$. Иными словами, параметр F_j считает число пар пикселей, которые попали в один суперпиксель, но принадлежат разным объектам на эталоне. Упомянутый индекс Ранда, говоря упрощенно, считает число пар не по суперпикселям в отдельности, а по всем парам пикселей на изображении. Пиксели, отмеченные в эталоне как граничные (черный цвет на рис. 2), не учитываются.

Рис. 2. Пример суперпикселя, в который объединились части сразу трех объектов из рис. 1 (наездника, лошади и фона)

Ошибка сегментации всего изображения в целом вычисляется по формуле

$$T = \frac{1000}{hw}\Sigma F_j,$$

где h, w — высота и ширина изображения в пикселях соответственно. Таким образом, получим оценку количества ошибочных пикселей на 1000 пикселей изображения, которую будем называть *ошибкой сегментации*.

В ходе выполнения работы проведены пять блоков экспериментов, в каждом из которых исследовались отклонения одного из параметров. Изменения различных параметров выполнялись независимо, для того чтобы понять, как каждый из них влияет на сегментацию. Отклонения во всех блоках происходят от одной и той же базовой конфигурации — *baseline*:

- *ratio* = 0,5;
- *kernelsize* = 6;
- *maxdist* = 15;
- цветовое пространство *Luv*;
- Гауссово ядро.

Разные изображения могут потребовать разного числа суперпикселей для качественной сегментации, поэтому было бы неправильно вычислять средний штраф и средний размер суперпикселей по коллекции изображений. Действительно, некоторые изображения сегментируются со штрафом порядка 5, другие — порядка 50. В связи с этим при запуске алгоритма на нескольких значениях исследуемого параметра, значение этого параметра отклонялось от базового (*baseline*) и вычислялось отклонение метрик (ошибка сегментации и размер суперпикселя) в процентах относительно *baseline*. После вычислялось среднее значение отклонений по коллекции изображений.

Компактность цветов суперпикселя

В рассматриваемой реализации QuickShift в качестве признаков пикселей используются не только значение цвета пикселя, но и его координаты на изображении. Благодаря этому кластеры в таком пространстве признаков должны образовывать непрерывные области и на изображении. Без рассмотрения пространственных признаков пиксели одного кластера (даже если кластер получился по цветовым признакам очень компактным) могут быть "разбросаны" по всему изображению, что порождает большое число мелких суперпикселей и противоречит цели сегментации. Заметим также, что и с учетом пространственной компоненты получаемые кластеры все-таки могут образовывать несвязные области, что требует дополнительной постобработки для использования результатов сегментации. Без использования пространственной ком-

понеты на одном и том же изображении может получиться несколько тысяч суперпикселей, вместо нескольких сот суперпикселей с использованием этой компоненты.

Данная реализация позволяет также регулировать компактность цветов пикселей, объединяемых в один суперпиксель, за счет использования параметра *ratio* в условии объединения пикселей:

$$(da^2 + db^2 + dc^2)ratio + dx^2 + dy^2 \leqslant maxdist,$$

где *da*, *db*, *dc* — расстояние между пикселями по компонентам цвета (их может быть и меньше трех); *dx*, *dy* — разность координат сравниваемых пикселей. При большом значении *ratio* в суперпиксели будут объединяться только пиксели с близкими значениями цвета, и чем больше будет важность этой компоненты, тем меньшие суперпиксели можно будет строить, тем больше регионов будет получаться. Вместе с тем при слишком малом значении в один суперпиксель будут попадать не очень похожие по цвету пиксели, что как и излишнее число регионов не отвечает целям сегментации изображения (рис. 3). Получается, что нужен компромисс в определении цветовой компактности суперпикселей.

Результаты численного измерения качества сегментации при различных значениях параметра *ratio* представлены в табл. 1. Данные показывают, что с увеличением параметра *ratio* качество сегментации улучшается. Однако это достигается за счет объединения в суперпиксели все более похожих по цвету пикселей, что увеличивает число суперпикселей. Более того, при больших значениях *ratio* генерируется больше очень мелких суперпикселей. Причина в том, что от больших суперпикселей отщепляются мелкие области, а если суперпиксель имел изменчивую текстуру, то при увеличении параметра он может вообще "развалиться" во множество мелких суперпикселей (рис. 4). Независимость от текстуры является большим недостатком используемого пространства признаков. Так, например, дорожный асфальт может иметь структуру со светлыми вкраплениями на темном фоне, что будет рассматриваться цветовыми признаками как совершенно различные объекты.

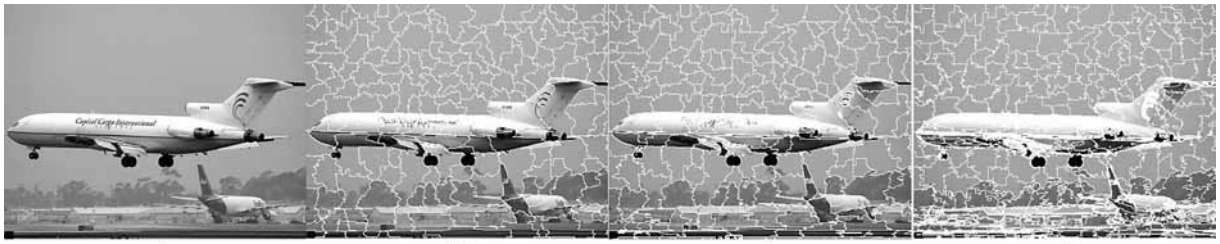
Таблица 1

Зависимость качества сегментации от параметра *ratio*

Метрика качества сегментации	Значение параметра *ratio*				
	0.125	0.25	0,5 (baseline)	1.0	2.0
Ошибка сегментации, %	+205	+87	0	−42	−73
Размер суперпикселей, %	+375	+149	0	−69	−91

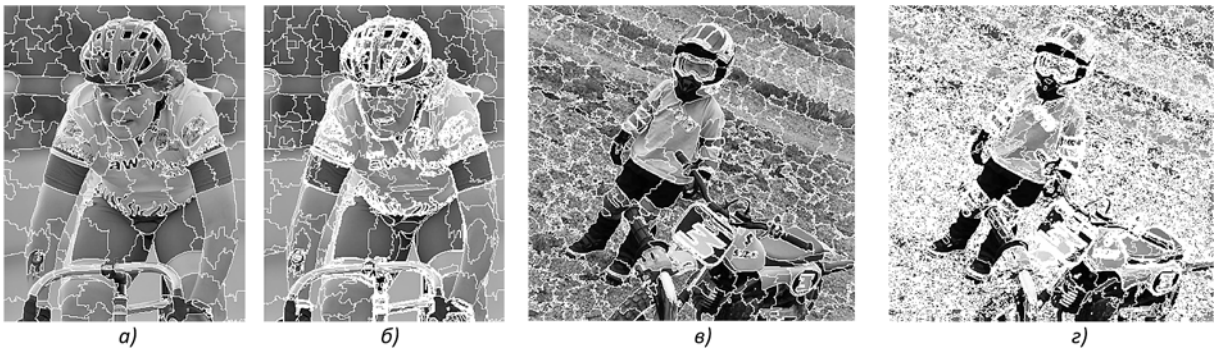
Анализ ошибок сегментации выявил следующие зависимости от параметра *ratio*:

- чем более похожи цвета граничащих объектов, тем значение *ratio* должно быть больше;
- чем больше резко изменчивой текстуры (ветки деревьев, трава, провода, мелкие надписи и т. п.), тем



а) б) в) г)

Рис. 3. Оригинальное изображение (*а*) и примеры сегментации с различными значениями параметра *ratio*: *б* — 0,125; *в* — 0,5; *г* — 2,0. При увеличении параметра в суперпиксели объединяются все более близкие пиксели, что уменьшает размеры суперпикселей



а) б) в) г)

Рис. 4. Примеры сегментации двух изображений при различных значениях параметра *ratio*. Первое изображение содержит мало изменчивой структуры, поэтому при увеличении *ratio* с 0,125 (*а*) до 2,0 (*б*) появляется не так много мелких суперпикселей. На втором изображении такое же изменение этого параметра приводит к тому, что суперпиксели травы и мотоцикла "разваливаются" во множество мелких суперпикселей (*в*, *г*)

больше при увеличении параметра появляется очень мелких суперпикселей; если все текстуры гладкие, то увеличение *ratio* слабо изменит сегментацию (рис. 4); таким образом, отсутствие изменчивой текстуры позволяет задавать меньшее значение параметра;

- чем мельче размеры искомых объектов, тем значение *ratio* должно быть больше.

Объединение под одним порогом *maxdist* цветовой и пространственной разницы между пикселями затрудняет определение оптимального значения параметров *ratio* и *maxdist*. Рассмотрение конкретных ошибок сегментации демонстрирует необходимость иметь два отдельных *maxdist*, а именно — для цветовых и пространственных компонент, чтобы иметь возможность более точно настроить алгоритм.

Цветовые пространства

Был проведен ряд экспериментов с различными цветовыми пространствами [29], значения цветов пикселей в которых было использовано в качестве цветовых признаков. Пример сегментации одного изображения в различных пространствах представлен на рис. 5. Эксперименты позволили получить следующие выводы.

- Пространство **RGB** принципиально хуже других подходит в качестве цветовых компонент признаков при сегментации изображения. Причина в том, что евклидово расстояние в этих пространствах не отражает воспринимаемого различия цветов с точки зрения человека. Данное пространство предназначено в основном для кодирования цвета при цветовоспроизведении.
- Несмотря на то что пространство **XYZ** было спроектировано чтобы смоделировать восприятие цвета в сетчатке человеческого глаза, результаты в этом пространстве не лучше, чем в RGB, так как расстояние в этом пространстве также не соответствует разнице цветов, воспринимаемой человеком.
- Пространства **Luv** и **Lab**, являющиеся попытками сконструировать цветовое пространство, адекватно отражающее восприятие человеком разницы цветов, показали наилучшие результаты при сегментации тестовых изображений.
- Эксперименты с игнорированием **компоненты L** в цветовых пространствах **Luv** и **Lab**, отвечающей за яркость, в общем случае ухудшают результаты сегментации. Вместе с тем в частных случаях игнорирование этой компоненты помогает более точно находить границу суперпикселей.
- Эксперименты в пространстве **HSV**, представляющем цвет в виде тона, насыщенности и яркости, показали результаты, сравнимые с результатами в пространствах RGB и XYZ. Однако евклидова метрика абсолютно не подходит для HSV, так как, во-первых, одна из координат является углом поворота в цветовом круге, во-вторых, вклад от разных компонент должен быть разный (например, вклад тона должен быть больше вклада от яркости).

Для точного определения наиболее подходящего цветового пространства качество сегментации оценивалось с использованием упомянутых метрик и коллекции изображений (табл. 2). Эксперименты показали, что пространство Luv показывает наилучшие результаты, строя при этом наименьшее число суперпикселей.

Таблица 2

Качество сегментации при использовании различных цветовых пространств

Метрика качества сегментации	Цветовое пространство				
	Lab	Lab без L	Luv (baseline)	Luv без L	RGB
Ошибка сегментации, %	+21	+200	0	+174	+321
Размер суперпикселей, %	+8,7	+261	0	+198	+541

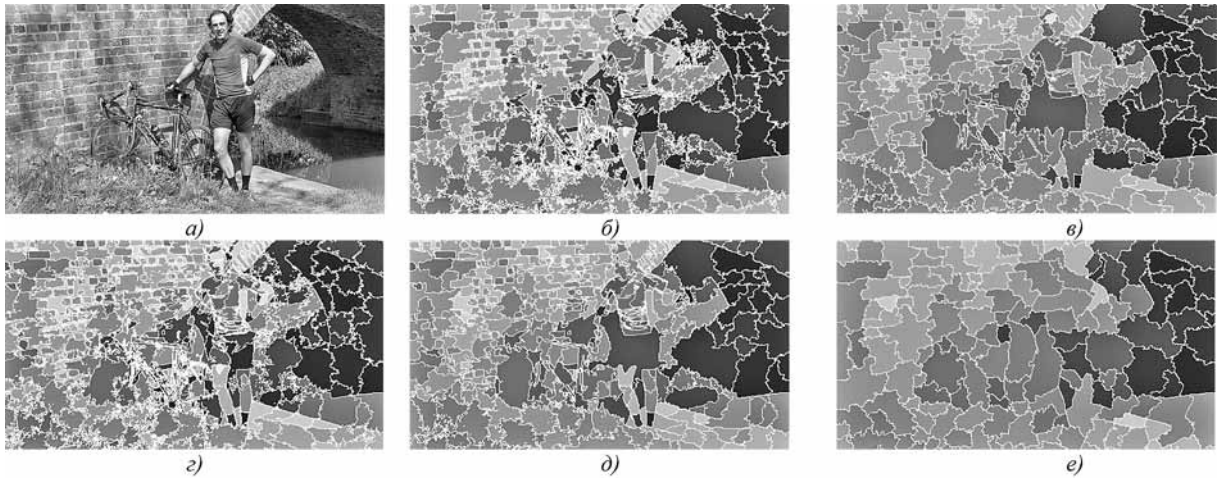


Рис. 5. Оригинальное изображение (*а*) и его сегментации в различных цветовых пространствах: *б* — Lab; *в* — Lab без компоненты L; *г* — Luv; *д* — Luv без компоненты L; *е* — RGB

Таблица 3

Особенности сегментации при использовании различных цветовых пространств

Цветовое пространство	Свойства изображений и особенности сегментации
Lab	Лучше других работает на изображениях, где встречаются блики от вспышек, чрезмерное количество солнечного света или, наоборот, сцена очень затемнена, или же содержатся и затемненные, и засвеченные области. Одним словом, на изображениях с очень плохим освещением, содержание которых плохо различимо на уменьшенной копии. При таких условиях цветовые компоненты пространства Luv становятся очень похожими (особенно в темных областях изображения), а в Lab все еще можно найти различия
Lab без L	Выделяемые объекты и фон имеют принципиально разные цвета, т.е. когда можно положиться лишь на цветовые компоненты (a и b), а яркость вносит лишний шум. Это особенно вредит если яркость изменяется в пределах одной области (потенциального суперпикселя), в этом случае функция плотности искажается, не показывая реальных центров областей. Игнорирование компоненты яркости устраняет эти искажения
Luv	Хорошо работает на изображениях с хорошим освещением сцены, содержание которых можно легко определить даже по уменьшенной копии
Luv без L	Выигрывает у других пространств, когда выделяемые объекты имеют похожие цвета, но разную текстуру. Более неравномерная текстура имеет меньшую плотность, поэтому ее пиксели присоединяются к пикселям соседнего объекта с более равномерной текстурой, а значит, с большей плотностью. Игнорирование компоненты яркости делает карту плотности более равномерной, что позволяет находить свои центры кластеров на каждом из объектов
RGB	В случае изображений с очень темным освещением цветовые компоненты пространств Lab и Luv становились очень похожими, что не позволяло выделить границу. Заметим также, что в этом пространстве функция плотности становится практически плоской и не выполняет функцию индикатора кластерных центров, т.е. нет эффекта «стекания» пикселей к центрам своих кластеров

Далее, несколько уступая, следует пространство Lab. Превосходство Luv над Lab ожидаемо и с теоретической точки зрения, ведь Luv использует более точный механизм адаптации точки абсолютно белого цвета. В качестве дальнейшего улучшения качества за счет цветового пространства можно использовать возможность учитывать модель освещенности (*illuminant*), предусмотренную в алгоритме для Luv.

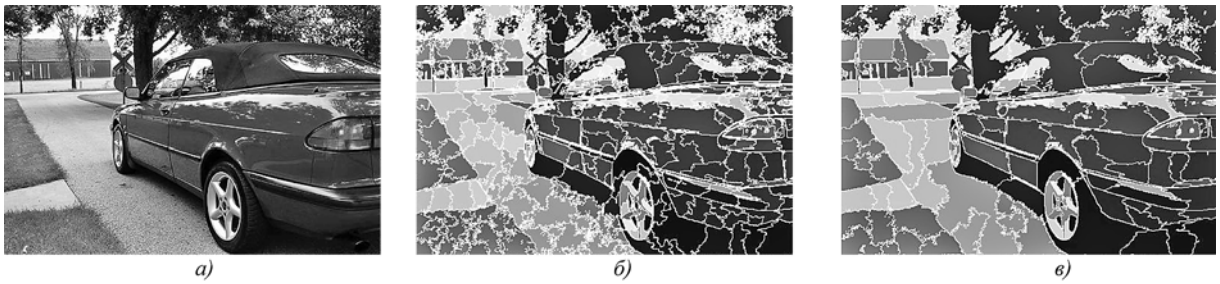
Были также проанализированы общие свойства изображений, хорошо сегментируемых в том или ином пространстве. Наблюдения приведены в табл. 3.

Обобщая приведенные наблюдения можно заключить, что выбор цветового пространства зависит от модели освещения изображения (сбалансированное освещение, засвеченное или затемненное изображение). При неверном выборе модели освещенности (искажение баланса белого) расстояния в цветовом пространстве не соответствуют реальным различиям цвета распознаваемых объектов. В алгоритмах преобразования цвета в пространства Lab и Luv существует возможность выбирать модель освещенности (прямой дневной свет, дневной свет от горизонта, свет лампы накаливания и др.), поэтому эксперименты с подбором модели освещенности можно предложить в качестве продолжения работы по изучению влияния цветовых пространств на качество сегментации.

Эксперименты с игнорированием компоненты яркости L в пространствах Lab и Luv показали, что данный прием дает преимущество лишь в частных случаях (при определенных комбинациях цветов и текстур граничащих объектов).

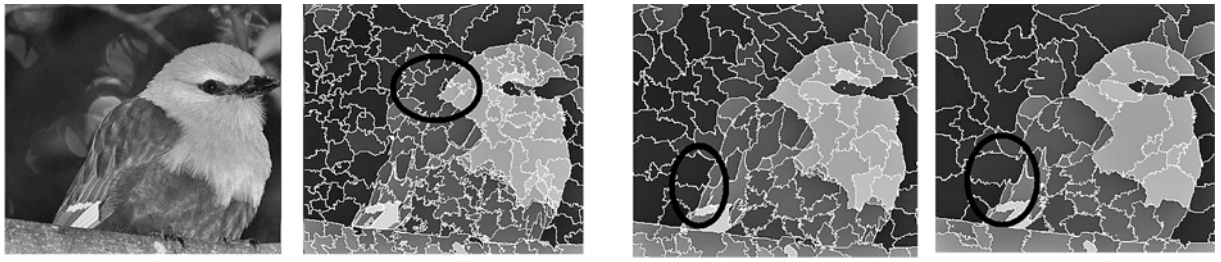
Размер окна Парзена для оценки плотности распределения пикселей

Размер окна для оценки плотности влияет на размер получающихся суперпикселей: чем уже окно, тем больше функция плотности будет иметь экстремумов (большее число суперпикселей), и чем шире окно, тем функция плотности будет более гладкой (более крупные суперпиксели). Другими словами, слишком широкое окно будет игнорировать мелкие детали изображения, а слишком узкое будет выделять слишком много деталей (рис. 6). Рассматриваемый параметр не может быть за-



а) *б)* *в)*

Рис. 6. Оригинальное изображение (*а*) и сегментации при ширине окна 1 (*б*) и 10 (*в*)



а) *б)* *в)* *г)*

Рис. 7. Пример изображения (*а*), когда промежуточное значение ширины окна (*в*) работает лучше более узкого (*б*) и более широкого окон (*г*). Черной линией обведено место наибольшей ошибки. Узкое окно объединяет даже очень разные цвета, более широкое окно присоединяет черное перо на спине к темному фону, а среднее окно выделяет для этого пера отдельный суперпиксель

Таблица 4

Влияние ширины окна на качество сегментации

Метрика качества сегментации	Значение параметра *kernelsize*				
	1.0	3.0	6.0 (baseline)	9.0	12.0
Ошибка сегментации, %	−6,1	−1,7	0	+8,9	+20,5
Размер суперпикселей, %	−40	−21,4	0	+23,0	+47

дан однозначно для всех задач и изображений даже теоретически, так как он зависит от желания пользователя, проводящего сегментацию, — насколько детализированная сегментация ему нужна. Поэтому размер окна Парзена должен подбираться в зависимости от предполагаемого размера интересующих объектов на изображении.

Численные результаты сравнения сегментации при различных размерах окна Парзена приведены в табл. 4. Сужение окна уменьшает размер суперпикселей, однако несмотря на это узкое окно не всегда показывает наименьшую ошибку. Рассматривая сегментацию конкретных изображений, авторы задались вопросами, когда лучше всего работает самое узкое окно (*kernelsize* = 1.0); когда — самое широкое (*kernelsize* = 12.0); когда приходится подбирать некоторое промежуточное значение. Исследование выявило следующие закономерности.

- **Узкое окно** строит такую карту плотности, которая почти не показывает положения центров однородных областей, за исключением очень небольших. Таким образом, карта плотности становится равномерной почти на всем изображении, поэтому сегментация проводится без учета центров областей. В связи с этим узкое окно часто допускает грубые ошибки объединения в один суперпиксель принципиально различных цветов. Часто наблюдалась ошибка, когда очень небольшая, но равномерная область "притягивала" к себе пиксели соседних областей с неравномерной текстурой.
- **Широкое окно** строит более информативную карту плотности, демонстрирующую центры областей. Недостатком очень широкого окна будет, естественно, пропуск небольших деталей, по размеру сравнимых с размером окна. Особенно плохо выделяется граница между объектами, если на границе они имеют похожие цвета. Широкое окно чаще страдает от того, что гладкая текстура имеет более высокую плотность и на объект с гладкой текстурой переходят пиксели областей, где текстура изменчива.
- **Промежуточное значение ширины окна** лучше сегментирует изображение, чем более узкие и более широкие окна, когда узкое окно не дает достаточной информации о центрах областей, а широкое сглаживает функцию плотности на некоторых деталях, на которых следовало бы обнаружить отдельный экстремум плотности и выделить область в отдельный суперпиксель. Пример такого случая представлен на рис. 7.

При практическом применении алгоритма предлагается применять следующие рекомендации для подбора параметра *kernelsize*:

- учитывать размеры искомых объектов: окно шириной *kernelsize* способно выделять объекты размером от 2*kernelsize* и более;
- чем больше отличаются цвета объектов и фона (чем резче граница), тем окно может быть шире;
- чем дальше друг от друга расположены объекты, тем окно может быть шире (например, если между объектами существуют небольшие зазоры, то ширину окна придется уменьшить, чтобы эти зазоры не сливались с объектами).

Функция ядра оценки плотности

Для тестирования были выбраны такие ядра, которые принципиально отличаются формой графика. Каждое из ядер пронормировано так, чтобы площадь под графиком в промежутке от $[-h, h]$ была равна 1, где h — это размер окна. Для экспериментов были использованы следующие ядра.

- Равномерное ядро $K_h(x) = \begin{bmatrix} \frac{1}{2h}, & |x| \leqslant h \\ 0, & |x| > h \end{bmatrix}$.
- Ядро Епанечникова

$$K_h(x) = \begin{cases} \frac{3}{4}\left(1 - \left|\frac{x}{h}\right|^2\right), & |x| \leqslant h \\ 0, & |x| > h. \end{cases}$$

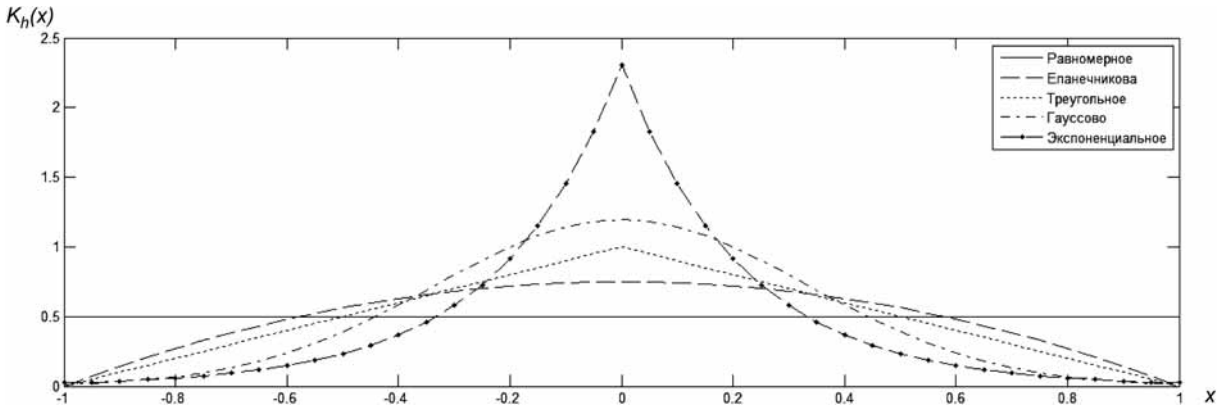


Рис. 8. Графики функции ядер при размере окна $h = 1$. Площадь под каждым графиком на промежутке $[-h; h]$ приблизительно равна 1

Таблица 5

Использование различных функций ядра при сегментации

Метрика качества сегментации	Функция ядра				
	Равномерное	Епанечникова	Треугольное	Гауссово (baseline)	Экспоненциальное
Ошибка сегментации, %	+2,6	+4,6	+3,6	0	+0,1
Размер суперпикселей, %	+13,0	+9,1	+7,0	0	+1,7

- Треугольное ядро $K_h(x) = \begin{cases} 1 - \left|\frac{x}{h}\right|, & |x| \leqslant h \\ 0, & |x| > h \end{cases}$.
- Гауссово ядро $K_h(x) = \frac{3}{h\sqrt{2\pi}}\mathbf{e}^{-\frac{9x^2}{2h^2}}$.
- Экспоненциальное ядро $K_h(x) = \frac{\ln 100}{2}\mathbf{e}^{-|x|\ln 100}$.

Графики функций использованных ядер представлены на рис. 8.

Результаты численной оценки качества сегментации, приведенные в табл. 5, демонстрируют преимущество Гауссова и экспоненциального ядер. Отметим, что равномерное ядро также является хорошим вариантом. Оно простое в вычислении, ненамного уступает им по качеству и даже строит несколько меньшее число суперпикселей. Тем не менее, следует отметить, что оно более чувствительно к выбору ширины окна.

В табл. 6 представлены найденные экспериментально преимущества использования различных ядер. Преимущество Гауссова и экспоненциального ядер при выделении более мелких частей объектов можно объяснить тем, что центральная область в этих ядрах имеет больший вес, чем в других ядрах. Таким образом, можно считать, что эти ядра имеют меньшую "эффективную" ширину окна.

Также было определено, что подобные ядра лучше выявляют центры кластеров. В центре кластера, где точки ближе друг к другу, больший вес центральной части ядра дает возможность зафиксировать, что точки

Таблица 6

Особенности сегментации при использовании различных функций ядра

Функция ядра	Особенности сегментации
Равномерное	Лучше других разграничивает объекты с мелкой, но резко меняющейся текстурой
Епанечникова	Дает выигрыш при разделении объектов, имеющих похожие цвета и относительно равномерную текстуру
Треугольное	Выигрывает при разделении объектов с равномерной текстурой, но работает лучше ядра Епанечникова, если цвета объектов менее похожи
Гауссово	Лучше выделяет более мелкие области, имеющие непохожие цвета с соседними областями
Экспоненциальное	Хорошо выделяет мелкие части объектов

действительно близко расположены друг к другу, в то время как равномерное ядро только считает число точек, попавших в окно. Однако оказывается, что слишком точное обнаружение центров кластеров может ухудшить сегментацию, и следует строить более равномерную карту плотности. Приведем пример. Граничащие объекты имеют похожие цвета, но на одном текстура равномерная, а на другом нет. Тогда равномерная текстура будет определяться как область с большей плотностью и часть пикселей со второго объекта будет отнесена к первому объекту. На некоторых текстурах механизм определения центров кластеров работает неверно, поэтому следует либо учитывать текстурные свойства наряду с чисто цветовыми, либо не слишком полагаться на него.

Метод разделения дерева в лес суперпикселей

Настройка данного параметра является наиболее важной и трудной частью адаптации алгоритма QuickShift. На данный момент разбиение дерева происходит простым удалением всех ребер длиннее заданного параметра *maxdist*. Чем больше этот пара-

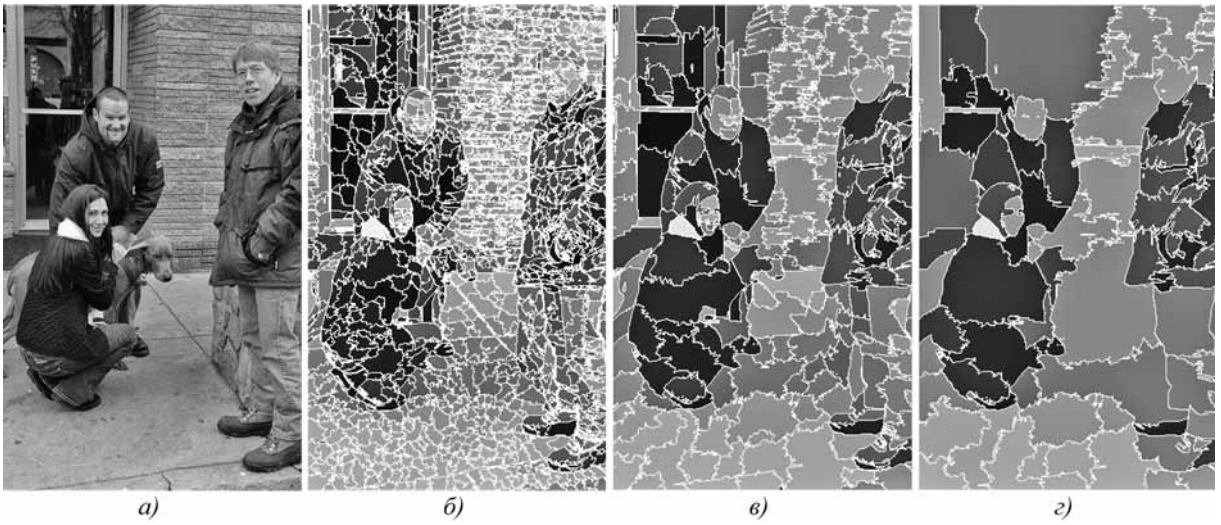


Рис. 9. Оригинальное изображение (*а*) и сегментации при различных значениях параметра *maxdist*: *б* — 5; *в* — 15; *г* — 25

Таблица 7

Зависимость качества сегментации от параметра *maxdist*

Метрика качества сегментации	Значение параметра *maxdist*				
	5	10	15 (baseline)	20	25
Ошибка сегментации, %	−77	−44	0	+70	+149
Размер суперпикселей, %	−84	−48	0	+52	+106

метр, тем крупнее будут суперпиксели, однако тем больше будет вероятность попадания в один суперпиксель нескольких объектов (рис. 9). Необходимо искать оптимальное значение максимальной длины ребра, позволяющее правильно разделить объекты и не получать излишне много суперпикселей.

Численные значения изменения качества сегментации и размеров суперпикселей при увеличении параметра *maxdist* представлены в табл. 7. Как и ожидалось, значение *maxdist* пропорционально размеру суперпикселей и обратно пропорционально качеству сегментации. Напомним также, что с увеличением *maxdist* время работы алгоритма увеличивается пропорционально $maxdist^2$.

Последующие эксперименты показали, что крайне трудно найти значение параметра *maxdist*, подходящее для всех изображений, так как на одних изображениях граничащие объекты могут иметь очень похожие цвета, в то время как на других изображениях эта же разница в цвете не образует границу объектов. Предположение авторов относительно данного параметра заключается в том, что причиной является не неправильно выбранное значение *maxdist*, а способ разделения дерева на основе максимальной длины ребра. Приведем несколько выявленных недостатков данного подхода.

- Ограничение *maxdist* накладывается только на ребра в дереве, что позволяет объединить в один суперпиксель (поддерево) существенно различные цвета, если они находятся в разных частях образуемого поддерева. В частности, при плавном переходе от одного объекта к другому объединяемые соседние пиксели действительно отличаются слабо, тем не менее, граница между этими объектами теряется.
- В один регион могут попасть существенно отличающиеся по цвету, но близко находящиеся пиксели, так как ограничение действует на сумму различий пространственных и цветовых компонент, поэтому разница в цвете компенсируется небольшим значением пространственной компоненты.
- Большую однородную область на изображении, которую следовало бы объединить в один суперпиксель, приходится разбивать, так как *maxdist* ограничивает пространственный размер суперпикселя сверху.

Заключение

В данной работе были проанализированы различные вариации алгоритма сегментации изображений QuickShift. Выявлено, как зависит поведение алгоритма от этих параметров, какие значения параметров дают наилучшие результаты на каких изображениях, сформулированы рекомендации по установке параметров алгоритма. Выявлен также ряд недостатков данной реализации и алгоритма в целом, а также несколько вариантов его улучшения.

Список литературы

1. **Шапиро Л., Стокман Дж.** Компьютерное зрение. М.: Бином. Лаборатория знаний, 2006.

2. **Ren X., Malik J.** Learning a classification model for segmentation // IEEE 9th International Conference on Computer Vision (ICCV). Nice, France. 14—17 October, 2003. Vol. 1. P. 10—17.

3. **Li Z., Wu X.-M., Chang S.-F.** Segmentation Using Superpixels: A Bipartite Graph Partitioning Approach // IEEE International

Conference on Computer Vision and Pattern Recognition (CVPR). Providence, USA. 16—21 June, 2012. P. 789—796.

4. **Fulkerson B., Vedaldi A., Soatto S.** Class Segmentation and Object Localization with Superpixel Neighborhoods // Proc. of IEEE 12th International Conference on Computer Vision (ICCV). Kyoto, Japan. 27 September — 4 October, 2009. P. 670—677.

5. **Landy M. S., Graham N.** Visual perception of texture // The Visual Neurosciences. Cambridge, MA, USA: MIT Press, 2004. P. 1106—1118.

6. **Malik J., Belongie S., Leung T., Shi J.** Contour and Texture Analysis for Image Segmentation // International Journal of Computer Vision. 2001. Vol. 43. N 1. P. 7—27.

7. **Martin D., Fowlkes C., Malik J.** Learning to detect natural image boundaries using local brightness, color, and texture cues // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2004. Vol. 26. N 5. P. 530—549.

8. **Horowitz S. L., Pavlidis T.** Picture Segmentation by a Directed Split and Merge Procedure // Proc. of 2nd International Joint Conference on Pattern Recognition. Copenhagen, Denmark. 13—15 August, 1974. P. 424—433.

9. **Beucher S.** The watershed transformation applied to image segmentation // Proc. of 10th Pfefferkorn Conference on Signal and Image Processing in Microscopy and Microanalysis. Cambridge, UK. 16—19 September, 1991. P. 299—314.

11. **Shi J., Malik J.** Normalized Cuts and Image Segmentation // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2000. Vol. 22. N 8. P. 888—905.

11. **Felzenszwalb P., Huttenlocher D.** Efficient graph-based image segmentation // International Journal of Computer Vision. 2004. Vol. 59. N 2. P. 167—181.

12. **Comaniciu D., Meer P.** Mean shift: A robust approach toward feature space analysis // Proc. of IEEE Transactions on Pattern Analysis and Machine Intelligence. 2002. Vol. 24. N 5. P. 603—619.

13. **Sheikh Y. A., Khan E. A., Kanade T.** Mode-seeking by medoidshifts // Proc. of IEEE 11th International Conference on Computer Vision (ICCV). Rio de Janeiro, Brazil. 14—20 October, 2007. P. 1—8.

14. **Vedaldi A., Soatto S.** Quick Shift and Kernel Methods for Mode Seeking // Proc. of European Conference on Computer Vision (ECCV). Marseille, France. 12—18 October, 2008. P. 705—718.

15. **Kass M., Witkin A., Terzopoulos D.** Snakes: Active contour models // International Journal of Computer Vision. 1988. Vol. 1. N 4. P. 321—331.

16. **Cremers D., Rousson M., Deriche R.** A review of statistical approaches to level set segmentation: Integrating color, texture, motion and shape // International Journal of Computer Vision. 2007. Vol. 72. N 2. P. 195—215.

17. **Zhu S., Yuille A.** Region competition: unifying snake/balloon, region growing, and Bayes/MDL/energy for multi-band image segmentation // IEEE Transactions on Pattern Analysis and Machine Intelligence. 1996. Vol. 18. N 9. P. 884—900.

18. **Levinshtein A., Stere A., Kutulakos K., Fleet D., Dickinson S.** TurboPixels: Fast Superpixels Using Geometric Flows // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2009. Vol. 31. N 12. P. 2290—2297.

19. **Fulkerson B., Soatto S.** Really quick shift: Image segmentation on a GPU // Proc. of Workshop on Computer Vision using GPUs (CVGPU). Crete, Greece. 10—11 September, 2010. P. 350—358.

20. **Vedaldi A., Fulkerson B.** VLFeat: An open and portable library of computer vision algorithms // Proc of 18th Annual ACM International Conference on Multimedia (ACM MM). Firenze, Italy. 25—29 October, 2010. P. 1469—1472.

21. **Herbin M., Bonnet N., Vautrot P.** A Clustering Method Based On the Estimation of the Probability Density Function and on the Skeleton by Influence Zones // Pattern Recognition Letters. 16 September, 1996. Vol. 17. N 11. P. 1141—1150.

22. **Everingham M., van Gool L., Williams C., Winn J., Zisserman A.** The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results // International Journal of Computer Vision. 2010. Vol. 88. N 2. P. 303—338.

23. **Meila M.** Comparing clusterings: An axiomatic view // Proc of 22nd International Conference on Machine Learning (ICML). Bonn, Germany. 7—11 August, 2005. P. 577—584.

24. **Unnikrishnan R., Pantofaru C., Hebert M.** Toward objective evaluation of image segmentation algorithms // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2007. Vol. 29. N 6. P. 929—944.

25. **Martin D.** An Empirical Approach to Grouping and Segmentation. Ph. D. dissertation. University of California, Berkeley, USA. 2002.

26. **Rand W. M.** Objective criteria for the evaluation of clustering methods // Journal of the American Statistical Association (JASA). 1971. Vol. 66. N 336. P. 846—850.

27. **Yang A. Y., Wright J., Ma Y., Sastry S. S.** Unsupervised segmentation of natural images via lossy data compression // Journal of Computer Vision and Image Understanding. 2008. Vol. 110. N 2. P. 212—225.

28. **Arbelaez P., Maire M., Fowlkes C., Malik J.** Contour Detection and Hierarchical Image Segmentation // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2011. Vol. 33. N 5. P. 898—916.

29. **Tkalcic M.** Colour spaces: perceptual, historical and applicational background // Proc. of The IEEE Region 8 EUROCON 2003. Computer as a Tool. Ljubljana, Slovenia. 22—24 September, 2003. Vol. 1. P. 304—308.

ИНФОРМАЦИЯ

8—10 октября 2013 г. в г. Новосибирске состоится
IV Всероссийская конференция с международным участием

"Знания — Онтологии — Теории" (ЗОНТ-13)

Конференция будет посвящена математическим методам представления данных, извлечения знаний и построения теорий предметных областей, анализу формальных понятий, а также методам извлечения информации из текстов естественного языка.

Подробности см. на сайте http://math.nsc.ru/conference/zont/13/

УДК 61.004.62

М. А. Большакова, ассистент,
В. В. Лобанов, канд. техн. наук, доц., e-mail: lobanovvv@gmail.com,
Саратовский государственный технический университет имени Ю. А. Гагарина

Программная реализация интеллектуального электронного словаря для дистанционного обучения

Рассматривается программная реализация интеллектуального электронного словаря по робототехнике на немецком языке в системе электронного образования, а также в виде встраиваемого модуля лингвистического процессора. Предложена аппаратно-техническая архитектура интеллектуального электронного словаря, выбраны модели представления знаний, хранения данных и формирования содержания словаря. Предложенное решение обеспечивает адаптируемость, открытость и виртуальность. Оболочка интеллектуального электронного словаря предоставляет возможность работы с заданной предметной областью.

Ключевые слова: *электронное обучение, интеллектуальный электронный словарь, аппаратно-техническая архитектура, модель представления знаний, модель хранения данных, программная платформа, Internet-сервис, web-интерфейсы*

Введение

Одной из новых образовательных технологий, доказавших эффективность, является электронное обучение (*e-Learning*). Во многих развитых странах электронное обучение охватывает все уровни образования и широко используется не только в университетах, а также в средних школах и в организациях корпоративного (послевузовского) обучения.

Российский рынок средств *e-Learning* также развивается стремительно. Все большее число вузов используют информационно-коммуникативные технологии в образовательном процессе, еще более высокими темпами они применяются при организации корпоративного обучения. Это привело к формированию новой индустрии электронного обучения.

Одним из перспективных направлений электронного обучения является компьютерная поддержка перевода для определенной предметной (проблемной) области, которым занимается так называемая корпусная лингвистика. Основным понятием для нее является корпус текстов, под которым понимается сформированная по определенным правилам выборка текстовых данных из проблемной области. Проблемная область определяется как "область реализации языковой системы, содержащая феномены, подлежащие лингвистическому описанию". Исследования в этой области включают составление моно- и многоязычных корпусов текстов различной тематики. На их основе осуществляется изучение определенных объектов языка, а также совершенствование лингвистического обеспечения разнообразных информационных систем.

Преимуществом использования корпусов является возможность автоматической обработки данных корпуса, в том числе наличие механизмов автоматического поиска необходимой смысловой информации. Обычно для ее сбора необходимо изучать большое число текстов и выписывать примеры "вручную". Этот недостаток затрудняет обработку больших массивов материала. С появлением текстовых корпусов на машинных носителях процесс сбора данных существенно упрощается, а их качество значительно улучшается.

Результаты исследования корпусов текстов могут использоваться для решения многих лингвистических задач. К их числу относятся: составление разнообразных словарей (слов, словосочетаний, частотных словарей

и т. д.), описание грамматического строя языка, смысловая классификация типов текстов. Результаты таких исследований создают основу для моделирования разнообразных систем автоматической обработки текста [1].

В качестве корпуса текстов могут служить корпусы текстов общей тематики, предназначенные для решения широкого круга задач автоматической обработки текста (такие как Penn TreeBank, British National Corpus, BulTreeBank, Croatian National Corpus). В таких корпусах текст снабжается грамматической информацией, обычно имеющей общепринятый вид согласно существующей в рамках рассматриваемого языка грамматики. Корпусы текстов также собираются для решения конкретной научно-исследовательской или практической задачи. Они обычно относятся к одной довольно узкой предметной области и размечаются с использованием различных грамматических моделей в зависимости от задачи, для которой они предназначены [2, 3].

Последний вид корпусов довольно часто используется исследователями. Такие корпуса предоставляют возможность самостоятельно определить лингвистический материал для работы и способ разметки в терминах принятой исследователем модели представления знаний и обработки языковых явлений. В связи с этим обстоятельством используют специальный корпус текстов. Специальный корпус текстов — это сбалансированный корпус, как правило, небольшой по размеру (несколько тысяч словоупотреблений), относящийся только к определенной предметной области и предназначенный для использования в целях, соответствующих замыслу составителя [4—6].

В нашем случае выбран корпус текстов в предметной области "мехатроника и робототехника", что обусловлено активным развитием этого научно-технического направления в разных странах ЕС. В частности, Германия является своего рода "законодателем моды" по робототехнике в ЕС — в этой стране создан образовательный стандарт по мехатронике и робототехнике, который широко используется в различных странах Европы.

Постановка задачи

В связи с изложенным выше актуальной является задача повышения эффективности перевода на основе создания интеллектуального электронного словаря (ИЭС) и его апробация для области робототехники на немецком языке для систем *e-Learning*. Такой словарь может также использоваться в качестве встраиваемого модуля лингвистического процессора в интеллектуальных системах автоматического перевода. Под ИЭС будем понимать часть интеллектуальных автоматизированных систем, которые содержат сформированные и определенным образом упорядоченные наименования понятий предметной области [5]. Под лингвистическим процессором будем понимать реализованную на ЭВМ формальную лингвистическую модель, способную "понимать тексты" на иностранном (в данном случае — на немецком) языке. Она обычно включает три основных набора правил — морфологические, синтаксические и семантические, а также и обслуживающие их толковые словари. Компоненты обеспечивают автоматическое взаимно-однозначное преобразование текста в морфологические, синтаксические и семантические структуры.

Отличие предлагаемого ИЭС от существующих состоит в том, что он должен содержать базу знаний (БЗ), в которой хранится информация о терминах, их словосочетаниях, грамматических категориях. Предлагаемая база основана на представлении знаний методами искусственного интеллекта. Собственно ИЭС как интеллектуальная система должна состоять из двух частей — вариативной и инвариантной. Вариативная часть отвечает за содержание и зависит от предметной области. Инвариантная часть образует оболочку системы, которая содержит, в частности, знания для контроля контента по формальным признакам при анализе базы данных (БД) словаря на непротиворечивость для организации интеллектуального ввода и других целей.

Для создания ИЭС необходимо решить следующие задачи.

1. Разработать архитектуру системы.
2. Выбрать модель представления знаний и модель хранения данных.
3. Осуществить выбор необходимой смысловой информации из различных источников для формирования корпуса текстов в сфере "робототехника и мехатроника" с учетом существующей классификации в предметной области.
4. Сформировать файл данных (формата Excel) со структурой, отражающей особенности корпуса текстов для ИЭС.
5. Реализовать ИЭС на выбранной программной платформе.
6. Сформулировать правила проверки правильности содержания БД данных на основе морфологии и семантики немецкого языка и формализовать их на основе интеллектуальной модели представления знаний.
7. Осуществить заполнение БЗ инвариантной части БД формализованными правилами морфологической и семантической проверки.
8. Выполнить ввод корпуса текстов по робототехнике и мехатронике из файла формата Excel в вариативную часть базы данных электронного словаря.
9. Провести апробацию функциональных возможностей автоматизированного интеллектуального немецко-русского отраслевого словаря для задач дистанционного обучения.

Далее представлено краткое описание результатов, полученных при решении перечисленных задач.

Описание функциональной структуры ИЭС

Управляющий модуль координирует действия подсистем (модулей) ИЭС в соответствии с определенными сценариями (рис. 1, см. третью сторону обложки). Общение с внешней средой осуществляется на основе интерфейса ввода-вывода: из внешней среды поступают запросы к системе по установленным формам,

предопределенным сценариями, в соответствии с которыми функционирует управляющий модуль. Определенные запросы пользователей инициируют соответствующие сценарии работы, такие как поиск определенного термина, ввод информации в БД в пакетном варианте; для эксперта — изменение в БД (модернизация); для пользователя — получение информации о склонении термина в заданных падежах. Результаты обработки представляются пользователям через интерфейс ввода-вывода.

Одним из центральных звеньев является система управления БД, построенной на основе семантической сети. Эта база содержит термины на немецком языке в единственном и множественном числах, перевод и артикль, а также (при наличии) информацию об устойчивых словосочетаниях. Система управления базой данных (СУБД) управляет функционированием БД (сохранение, восстановление, резервное копирование и другие операции).

Интеллектуальный модуль управления (МУ) содержит модуль управления и БЗ продукционного типа. При этом МУ БЗ функционирует аналогично механизму логического вывода (машине логического вывода) в экспертной системе (ЭС).

Модуль управления рабочей памяти (РП) и собственно РП также аналогичны подсистеме в ЭС (называемой "рабочая доска"), в которой сохраняются временные данные, реализуется их обработка. На его основе, например, осуществляется ввод из файла формата Excel (пакетный режим), обработка информации по определенным правилам (проверка правильности ввода и другие операции) с дальнейшим вводом в БД, а также реализация обратной связи при запросе из БД в модуль РП по сценарию управляющего модуля.

Важный компонент ИЭС — это лексико-семантический модуль, который выполняет анализ поступающих данных в модуль управления РП и проверку их корректности. Кроме этого лексико-семантический модуль осуществляет синтез различных словоформ на основе информации об устойчивых словосочетаниях по правилам, которые содержатся в БЗ.

Разработка архитектуры и выбор программной платформы

При программной реализации электронной системы необходимо определить область и среду ее функционирования. В рассматриваемом случае необходимо разработать кросс-платформенную, масштабируемую, общедоступную и двуязычную систему с упрощенными стандартизированными механизмами доступа к ее функциональной части. В силу учебно-методического характера проекта и особенностей большей части исходной информации в сети Internet для заполнения информационной БД особую роль приобретает выбор инструментальных средств реализации ИЭС. Для обеспечения предъявляемых к словарю программной системы требований целесообразна его разработка в виде Internet-сервиса с обеспечением доступа к нему на основе web-интерфейса. Для реализации целевой системы необходимо использовать некоммерческое программное обеспечение, в отношении которого у пользователя права на неограниченную (по времени) установку, запуск, а также свободное использование, изучение, распространение и изменение будут защищены юридически авторскими правами на основе свободных лицензий.

Некоммерческое программное обеспечение ИЭС должно быть общесистемным и прикладным. При этом общесистемное программное обеспечение содержит программное обеспечение сервера БД, в том числе операционную систему Linux CentOS 5.x или Window Server СУБД MySQL v5, а также web-сервер Apache v2 с поддержкой PHP 5.1.

Серверную часть образует web-сервер, возвращающий страницы приложения по запросам пользователя. Эти страницы создаются динамически на основе информации, обрабатываемой приложением. Именно на создание страниц "на лету" направлены различные расширения web-серверов, например, -CGI или ASP.

Программные средства мониторинга, резервного копирования и восстановления данных включают следующие инструменты: cron; bash; mysql-tools; munin.

Программное обеспечение рабочих станций включает операционную систему, совместимую с Microsoft Windows 2000/XP, Mac OS X или выше, Linux, Android (для мобильных устройств). Клиентское приложение Internet Browser (браузер) обеспечивает поддержку javascript, последовательно запрашивает страницы с сервера, используя Dynamic HTML для управления интерфейсом и частичной обработкой информации на компьютере пользователя.

Пользовательский интерфейс выделен отдельно, так как именно формированием клиентского интерфейса и работой с ним web-приложения отличаются от привычных клиент-серверных приложений. В последнем случае клиентское приложение обменивается с сервером только данными, используя для формирования интерфейса ресурсы приложения. В web-приложении интерфейс полностью формируется на сервере, оставляя для исполнения клиентом только управление созданной страницей. Существующие стандарты на браузеры определяют дополнительную специфику процедуры использования приложений. В частности, два свойства, которые необходимо принимать во внимание при реализации приложения, это наличие истории просмотра страниц и произвольный доступ к любой странице приложения по известному адресу. Последнее свойство обязательно должно учитываться в приложениях, использующих авторизацию пользователя.

Обобщая изложенное выше можно выделить основные особенности аппаратно-технической и функционально-инструментальной архитектуры ИЭС:

- отсутствие необходимости использовать дополнительное программное обеспечение на стороне кли-

ента, что позволяет автоматически реализовать клиентскую часть как кросс-платформенную;

- открытость, т. е. возможность подключения практически неограниченного числа клиентов;
- благодаря единственному месту хранения данных и наличию СУБД обеспечение выполнения минимальных требований для поддержки целостности данных;
- общедоступность при работоспособности сервера и каналов связи;
- отсутствие существенных ограничений относительно объема данных web-систем.

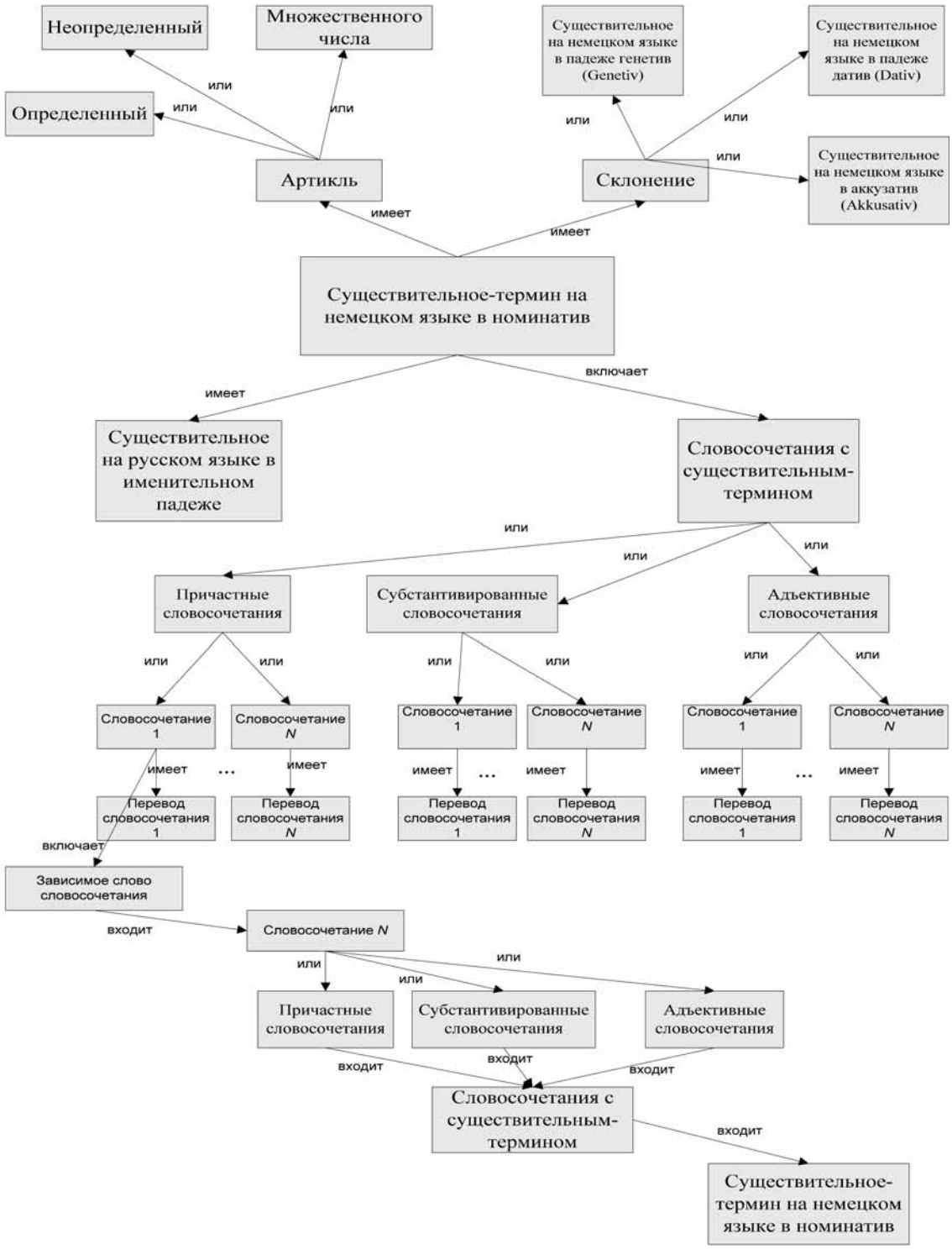


Рис. 3. Семантическая структура вариативной части БД словаря

Упрощенная схема аппаратно-технической архитектуры ИЭС представлена на рис. 2, см. третью сторону обложки.

Далее более подробно описываются система управления БД и ее структура.

Характеристика СУБД ИЭС

Сформированная БД содержит информацию о термине, словосочетаниях и их грамматических категориях. СУБД обеспечивает выполнение следующих функций над ее вариативной частью:

- ввод информации в интерактивном режиме;
- ввод информации в режиме off-line из Excel-файла соответствующего формата;
- вывод информации в Excel-файл заданного формата;
- хранение данных;
- интеллектуальная проверка правильности ввода;
- поиск существительных предметной области "робототехника и мехатроника" на немецком языке;
- вывод заданной информации (экранная и печатная формы):
  - найденный термин на немецком языке (инфинитив, единственное число);
  - перевод (аналог) на русском языке;
  - дополнительная информация (артикль; окончания существительных; окончания прилагательных; устойчивые словосочетания).

Исходная информация для вариативной части БД, выбранная из различных источников, содержит записи более 4000 слов существительных из области робототехники.

Описание структуры БД ИЭС

В качестве формализма для представления информации в вариативной части ИЭС выбрана семантическая сеть как наиболее подходящая модель представления знаний для хранения и обработки корпусов текстов на немецком языке для многих технических предметных областей [6—9].

База знаний словаря построена на основе расширенной семантической сети. Используемая расширенная семантическая сеть представлена в виде дерева, т. е. состоит из обыкновенных графов, содержащих N вершины и $N - 1$ дуг (рис. 3).

Вершины связаны дугами, которые отражают различные грамматические и морфологические отношения. Корнем дерева семантической сети является существительное на немецком языке в именительном падеже (номинатив). От него исходят дуги к вершинам "артикль", "склонение", "существительное на русском языке в именительном падеже", которые выражают отношения между вершинами-концептами и представляют функциональные связи, т. е. связаны глаголом "иметь", а также по типу отношений являются N-арными. Вершина "артикль" связана с вершинами, отражающими виды артиклей "определенный", "неопределенный" и "множественный". Вершина "склонения существительного" связана с вершинами "существительное на немецком языке в падеже генетив", "существительное на немецком языке в падеже датив" и "существительное на немецком языке в падеже аккузатив" с помощью дуг. Вершина "словосочетания с существительным-термином" связана с вершиной существительное на немецком языке в номинатив" дугой, которая выражает функциональную связь с глаголом "включать". Вершина "словосочетания с существительным-термином" включает вершины "причастные словосочетания", "субстантивированные словосочетания" и "адъективные словосочетания". Каждая из вышеперечисленных вершин имеет N-арные отношения с вершинами "словосочетание 1...N". Вершина "словосочетание" связана с вершиной "перевод". От вершины "словосочетание" исходит дуга, отображающая отношение принадлежности к вершине "зависимое слово словосочетания". Эти вершины объединены на основе дуг с вершиной "словосочетания с существительным-термином", которая связана дугой с вершиной "существительное термин на немецком языке в именительном падеже".

Процедура ввода информации в БД

Доступ к БД ИЭС реализуется на основе программы, вход в которую осуществляется двумя категориями пользователей: собственно, пользователем, функции которого ограничиваются поиском и просмотром информации; экспертом, который может вносить изменения в БД словаря по установленным правилам.

Авторизированный вход в систему экспертом осуществляется на основе пароля. В программе используются две формы, с которыми работает эксперт (рис. 4, 5).

Информация о термине, введенная экспертом, записывается в БД ИЭС. Осуществление правильности переноса в БД релевантной информации о термине реализуется на основе продукционных правил. Для формы 1, отвечающей за изменение данных о термине, в программе используются соответствующие правила проверки всех полей.

При проверке форм 1, 2 по изменению данных о термине и словосочетаниях применяется более 200 продукционных правил, которые сформированы авторами.

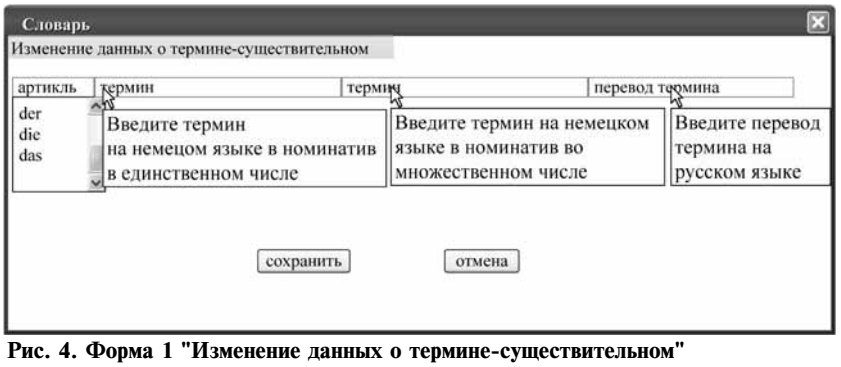


Рис. 4. Форма 1 "Изменение данных о термине-существительном"

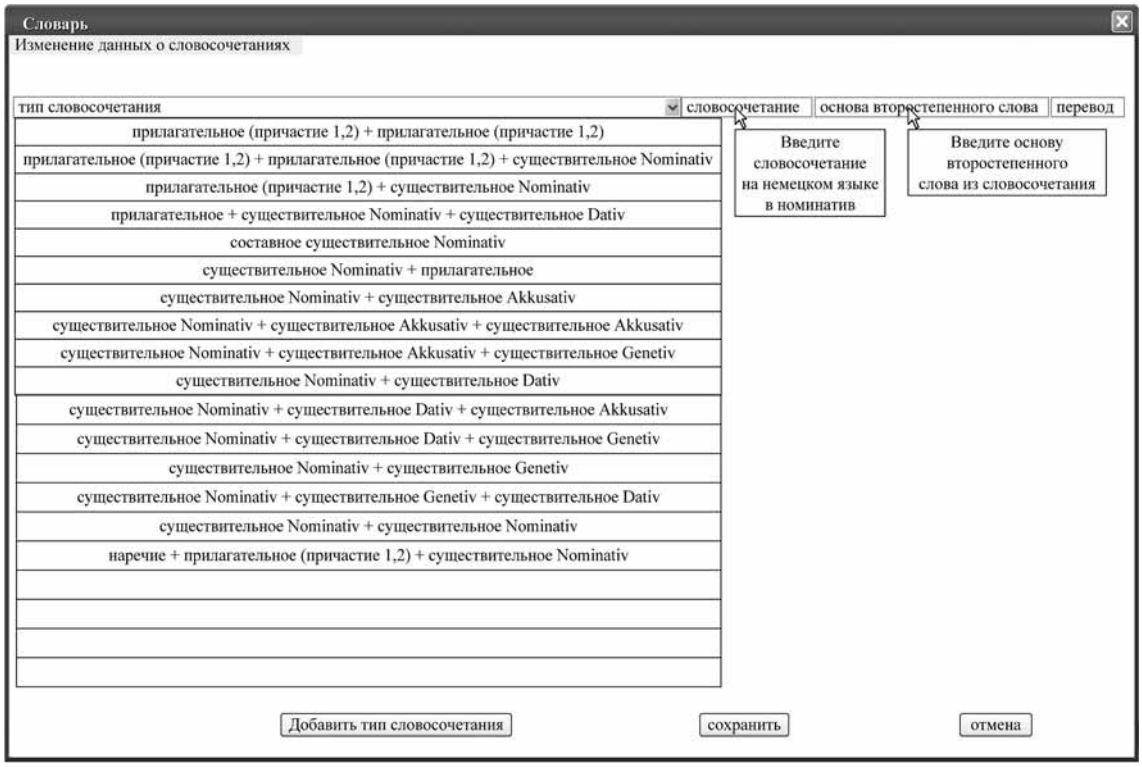


Рис. 5. Форма 2 "Изменение данных о словосочетаниях"

Заключение

Разработаны методические основы для создания ИЭС, выбраны модели представления знаний, хранения данных и формирования его содержания, программно реализована система ИЭС. Принятая архитектура ИЭС обеспечивает адаптируемость, открытость, "дружелюбный" пользовательский интерфейс и "виртуальность", что является ее несомненным преимуществом. Оболочка ИЭС предоставляет возможность работы с заданной предметной областью. Это достоинство можно использовать для совершенствования автоматизированного рабочего места преподавателя и переводчика, работающего в различных предметных областях, не только в научно-технической, но и, например, в сфере экономики, права, экологии и др. При этом специалисты-эксперты обеспечивают заполнение содержания, при необходимости модернизируют его, наполняют новыми терминами и словосочетаниями. Работа эксперта облегчается благодаря стандартным формам для ввода информации и интеллектуальной процедуре заполнения БЗ на основе правил проверки корректности введенных данных по формальным признакам.

Пользователь получает готовый продукт с понятным интерфейсом для работы с терминологией интересующей его предметной области. Таким образом, предлагаемый ИЭС представляет интеллектуальную систему, которая обеспечивает хранение и использование знаний о грамматических категориях и словосочетаниях с искомым термином. Пользователь системы имеет "полную картину" термина, а эксперт — возможность не только задавать для этого информацию, но и расширить (актуализировать) ее. Предлагаемый электронный словарь будет полезен изучающим иностранный язык, профессионалам в сфере преподавания и переводоведения. Он может быть использован также в качестве встраиваемого модуля для системы обработки и распознавания текстов на немецком языке.

Список литературы

1. **Biber D., Conrad S., Reppen R.** Corpus Linguistics: Investigating Language Structure and Use. Cambridge: Cambridge University Press. 1998. 312 p.

2. **Barbu C., Evans R., Mitkov R.** A corpus based investigation of morphological disagreement in anaphoric relations // In Proceedings of the Language Resources and Evaluation Conference. Wolvehapton. 2002. P. 1995—1999.

3. **Ng V., Cardie C.** Improving machine learning approaches to coreference resolution // In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. 2002. P. 104—111.

4. **Камшилова О. Н., Кожина М. В., Николаева Е. А.** Разработка корпуса текстов петербургских школьников: задачи и перспективы // Материалы IV Международной научно-практической конференции. Прикладная лингвистика в науке и образовании. 27—28 марта, 2008. Санкт-Петербург, РГПУ. 2008. С. 92—98.

5. **Потапова Р. К.** Основные тенденции развития многоязычной корпусной лингвистики // Речевые технологии. 2009. № 3. С. 93—112.

6. **Потапова Р. К.** Многоязыковая корпусная лингвистика в XXI веке // Материалы Международной научной конференции "Вызовы XXI в. — поликультурный и многоязычный мир". Москва. 2009. С. 110—121.

7. **Савина А., Типикина Т.** Что внутри электронного словаря? // Наука и жизнь. 2008. № 10. С. 56—58.

8. **Мешалкин В. П.** Экспертные системы в химической технологии. М.: Химия, 1995. 365 с.

9. **Кафаров В. В., Мешалкин В. П., Зархина И. И.** Продукционно-фреймовые модели представления знаний для автоматизированного синтеза ресурсосберегающих химико-технологических систем // Доклады АН СССР. 1989. Т. 307. № 3. С. 660—664.

УДК 004.056.5

А. В. Галатенко, канд. физ.-мат. наук, стар. науч. сотр., МГУ имени М. В. Ломоносова, e-mail: agalat@msu.ru,

В. А. Галатенко, д-р физ.-мат. наук, зав. сектором, Федеральное государственное бюджетное учреждение науки Научно-исследовательский институт системных исследований Российской академии наук (НИИСИ РАН), г. Москва, e-mail: galat@niisi.msk.ru

К постановке задачи разграничения доступа в распределенной объектной среде

Рассматривается задача разграничения доступа применительно к объектным системам. По мнению авторов, с позиций объектного подхода и с опорой на логический аппарат удается добиться естественной и технологичной постановки этой задачи. Подобный подход целесообразно реализовать и применить в рамках концепции контролируемого выполнения.

Ключевые слова: *разграничение доступа, распределенная объектная среда, предикатное задание*

Введение

Информационная безопасность (ИБ) является относительно замкнутой дисциплиной, развитие которой не всегда синхронизировано с изменениями в других областях информационных технологий. В частности, в ИБ пока не нашли отражения основные положения объектно-ориентированного подхода, ставшего основой при построении современных информационных систем. Не учитываются в ИБ и достижения в технологии программирования, основанные на накоплении и многократном использовании программистских знаний [1].

В данной работе сделана попытка рассмотреть задачу разграничения доступа применительно к объектным системам. В определенной степени это просто переформулировка традиционной постановки задачи (например, [2]). Представляется, однако, что с позиций объектного подхода и с опорой на логический аппарат удается добиться более естественной и технологичной постановки.

1. Недостатки существующих моделей разграничения доступа с точки зрения объектного подхода

Исходя из основных положений объектно-ориентированного подхода следует в первую очередь отказаться от традиционного деления на активные и пассивные сущности (субъекты и объекты в привычной для ИБ терминологии). Пользователям объектов доступны только методы (активные сущности), реализация которых (и, в частности, доступ к пассивным сущностям, таким как переменные и их значения) является скрытой, инкапсулированной.

Инкапсуляция вступает в конфликт и с таким важным положением ИБ как безопасность повторного использования объектов. Объект нельзя очистить внешним образом (заполнить нулями или случайной последовательностью бит), если только он сам не предоставляет соответствующий метод. При наличии такого метода надежность очистки зависит от корректности реализации и вызова метода.

По-видимому, следует признать устаревшим положение о том, что разграничение доступа направлено на защиту от злоумышленных пользователей [3]. Современные информационные системы характеризуются чрезвычайной сложностью и их внутренние ошибки представляют не меньшую опасность.

Динамичность современной программной среды в сочетании со сложностью отдельных компонентов существенно сужают область применимости самой употребительной — дискреционной модели управления доступом. При определении допустимости доступа важно не только (и не столько) то, кто обратился к объекту, но и то, какова семантика действия. Без привлечения семантики нельзя определить троянские программы, противостоять которым дискреционное управление доступом, как известно, не в состоянии.

В последнее время появляются новые модели управления доступом, например, модель "песочницы" (sandbox [4]). К сожалению, и эта модель не учитывает семантику программ, что, по мнению авторов, является основной причиной выявляемых слабостей в системе безопасности.

Представляется, что в настоящее время проблема разграничения доступа существует в трех почти не связанных между собой проявлениях:

- традиционные модели (дискреционная и мандатная);
- модель песочницы (используемая в Java-среде и близкой ей по духу системе Safe-Tcl [5]);
- модель фильтрации (используемая в межсетевых экранах, см., например, [2]).

Отметим, что ролевой доступ не является самостоятельной моделью. Это надстройка над существующими моделями, предназначенная для упрощения администрирования.

Описание и контроль интерфейсов средствами программирования по контракту (см., например, [6]) в первую очередь служат целям тестирования, отладки и верификации, однако эти средства можно рассматривать и применительно к задаче разграничения доступа.

В модели фильтрации и в программировании по контракту правила разграничения доступа задаются в наиболее общем виде, для их интерпретации созданы наиболее продвинутые системы.

По мнению авторов, необходимо построить унифицированную модель разграничения доступа на основе развития и обобщения существующих подходов.

2. Основные предположения

Рассматривается множество объектов (в смысле объектно-ориентированного программирования). Часть объектов может являться контейнерами, группирующими объекты-компоненты, задающими для них общий контекст, выполняющими общие функции и реализующими итераторы. Контейнеры либо вложены друг в друга, либо не имеют общих компонентов.

С каждым объектом ассоциирован набор уникальных идентификаторов (UID). К объекту можно обратиться только по UID. Разные UID могут предоставлять разные методы и быть доступными для разных объектов.

Каждый контейнер позволяет опросить набор UID объектов-компонентов, удовлетворяющих некоторому условию. Возвращаемый результат в общем случае зависит от вызывающего объекта.

Объекты изолированы друг от друга. Единственным видом межобъектного взаимодействия является вызов метода.

Предполагается, что используются надежные средства аутентификации и защиты коммуникаций. В плане разграничения доступа локальные и удаленные вызовы не различаются.

Предполагается, что разрешение или запрещение доступа не зависит от возможного параллельного выполнения методов (синхронизация выделяется в отдельную проблему, которая здесь не рассматривается).

3. Формальная постановка задачи

Разграничивается доступ к уникальным идентификаторам объектов, а также к методам объектов (с учетом значений фактических параметров вызова). Правила разграничения доступа (ПРД) задаются в виде предикатов над объектами.

Рассматривается задача разграничения доступа для выделенного контейнера CC, компонентами которого должны являться вызывающий и/или вызываемый объекты. UID этого контейнера предполагается общеизвестным. Считается также, что между внешними по отношению к выделенному контейнеру объектами возможны любые вызовы.

Выполнение ПРД контролируется монитором обращений.

4. Первый уровень конкретизации

При вызове метода будем разделять действия, производимые вызывающим объектом (инициация вызова) и вызываемым методом (прием вызова и завершение вызова).

При инициации вызова может проводиться преобразование UID фактических параметров к виду, доступному вызываемому методу (трансляция UID). Трансляция может иметь место, если вызываемый объект не входит в тот же контейнер, что и вызывающий.

Параметры методов могут быть входными и/или выходными. При приеме вызова возникает информационный поток из входных параметров в вызываемый объект. В момент завершения вызова возникает информационный поток из вызываемого объекта в выходные параметры. Эти потоки могут фигурировать в правилах разграничения доступа.

Структурируем множество всех ПРД, выделив четыре группы правил.

4.1. Политика безопасности контейнера

Обозначим через $SP(C)$ правила, общие для всех объектов, входящих в контейнер C. Назовем $SP(C)$ политикой безопасности контейнера C.

Будем считать, что все внешние объекты (не входящие в выделенный контейнер CC, см. разд. 3), равно как и сам CC принадлежат контейнеру UC с пустой (т. е. тождественно истинной) политикой безопасности.

4.2. Ограничения на вызываемый метод

Пусть метод $M1$ объекта $O1$ в точке $P1$ своего выполнения должен вызвать метод M объекта O. Обозначим через $RE(P1)$ правила, которым должен удовлетворять метод M. Их можно подразделить на четыре подгруппы:

- правила, описывающие требования к формальным параметрам вызова (обозначим их $PRE(P1)$);
- правила, описывающие требования к семантике M ($SRE(P1)$);
- реализационные правила, накладывающие ограничения на возможные реализации M ($IRE(P1)$);
- разграничительные правила ($ARE(P1)$), накладывающие ограничения на вызываемый объект O.

4.3. Ограничения на вызывающий метод

Метод M объекта O, потенциально доступный для вызова, может предъявлять к вызывающему объекту следующие группы требований (весь набор требований обозначим через $RR(M)$):

- правила, описывающие требования к фактическим параметрам вызова $PRR(M)$;
- разграничительные правила $ARR(M)$, накладывающие ограничения на вызывающий объект; вычисление предикатов $ARE(P1)$ и $ARR(M)$ может потребовать взаимной аутентификации объектов или иных действий аналогичного назначения.

Определение. Метод M совместим по параметрам в точке $P1$, если при означивании формальных параметров фактическими обеспечивается истинность предиката $PRE(P1)$ & $PRR(M)$.

4.4. Добровольно налагаемые ограничения

Можно выделить три разновидности предикатов, соответствующих семантике и/или особенностям реализации методов:

- утверждения о фактических параметрах вызова метода M в точке $P1$ (см. подразд. 4.2), обозначим их через $PFR(P1)$;
- предикат, описывающий семантику метода M ($SFE(M)$);
- предикат, описывающий особенности реализации метода M ($IFE(M)$).

Перечисленные ограничения можно назвать добровольными, поскольку они соответствуют реальному поведению объектов и не связаны с какими-либо внешними требованиями.

4.5. Условие допустимости вызова

Определение. Метод M, совместимый по параметрам, семантически совместим в точке $P1$, если при означивании формальных параметров фактическими обеспечивается истинность предиката $(PFR(P1)\ \&\ SFE(M)) => \mathrm{SRE(P1)}$.

Определение (условие допустимости вызова). Пусть метод $M1$ объекта $O1$ в точке $P1$ своего выполнения должен вызвать семантически совместимый метод M объекта O. Пусть далее $O1$ принадлежит набору вложенных контейнеров $C1 \ldots OC$, а O принадлежит набору $C \ldots OC$, где OC — минимальный контейнер, содержащий $O1$ и O. Вызов $M1 \rightarrow M$ разрешен, если $O1$ обладает $\mathrm{UID}(O)$, по этому UID доступен метод M и следующий предикат при означивании формальных параметров вызова фактическими имеет значение "истина" при инициации, приеме и завершении вызова:

$$SP(C1)\ \&\ \ldots\ \&\ SP(OC)\ \&\ \ldots\ \&\ SP(C)\ \&\ \&\ IFE(M1)\ \&\ \&\ IRE(P1)\ \&\ IFE(M)\ \&\ ARE(P1)\ \&\ ARR(M). \quad (1)$$

В момент начала выполнения метода M в качестве предиката, описывающего особенности реализации M, принимаем выражение $IRE(P1)\ \&\ IFE(M)$, заменяющее в выражении (1) $IFE(M)$ при рассмотрении вызовов, выполняемых M. Это очевидно влияет на допустимость вызовов, выполняемых M и методами, вызываемыми им прямо или косвенно.

4.6. Внутренние и внешние вызовы

Определение. Рассмотренный в подразд. 4.5 вызов называется внутренним, если O входит в контролируемый контейнер CC.

При внутреннем вызове может проверяться допустимость всех действий, выполняемых M.

Определение. Если O не принадлежит CC, вызов называется внешним. При внешнем вызове контролироваться (по общему правилу (1)) могут только вызовы компонентов CC, имеющие место в процессе выполнения M.

Будем считать, что каждое межобъектное взаимодействие происходит в рамках известных протоколов, так что монитор обращений в состоянии связать вызов компонента CC внешним объектом и предшествующий вызов $M1 \rightarrow M$, чтобы, в частности, проверить истинность предиката $IFE(M)$.

5. Оптимизация вычисления ПРД

Оптимизация вычисления ПРД выполняется с использованием традиционных методов оптимизации программ. Она может осуществляться по двум основным направлениям:

- однократное вычисление предикатов (экономия общих выражений);
- уменьшение числа членов ПРД (свертка выражений).

5.1. Однократное вычисление предикатов

Вообще говоря, при определении допустимости вызова должны каждый раз вычисляться все компоненты выражения (1), поскольку в них может входить текущее время (или иные меняющиеся значения) или их вычисление может сопровождаться побочными эффектами.

Если возможно однократное вычисление предикатов, то его наиболее естественно проводить в следующие моменты времени:

- при добавлении объекта к контейнеру;
- при подборе вызываемого объекта и установлении связи с ним (назовем это редактированием связей).

5.2. Уменьшение числа членов ПРД

Идея уменьшения числа членов ПРД состоит в использовании выражения $(A => B) => ((A\ \&\ B) = A)$.

Определение. Пусть контейнер $C1$ с политикой $SP(C1)$ вложен в контейнер $C2$ с политикой $SP(C2)$. Назовем это вложение информационно безопасным, если истинно соотношение $SP(C1) => SP(C2)$.

Иными словами, безопасно вложенный контейнер может только ужесточать политику объемлющего контейнера. Очевидно, отношение безопасной вложенности является транзитивным. Для безопасно вложенных контейнеров достаточно проверять выполнение политики безопасности самого внутреннего из них.

Определение. Пусть объект O вложен в контейнер C с политикой $SP(C)$. Это вложение называется информационно безопасным, если для каждого метода M объекта O истинно соотношение $(SFE(M)\ \&\ IFE(M)) => SP(C)$.

Для безопасно вложенных объектов можно не проверять выполнение требований политики безопасности контейнера.

Определение. Метод M, совместимый семантически, совместим по реализации в точке $P1$, если при означивании формальных параметров фактическими обеспечивается истинность предиката $IFE(M) => IRE(P1)$.

Для совместимых по реализации методов можно не проверять выполнение требований вызывающего метода (за исключением разграничительных требований).

6. Обработка ПРД

Для работы с ПРД служат методы "прочитать/записать", доступ к которым разграничивается на общих основаниях (в соответствии с выражением (1)). Целостность объектов вообще и ассоциированных с ними ПРД, в частности, может обеспечиваться традиционными средствами электронной подписи.

Естественно считать, что все ПРД, кроме разграничительных, задает автор объекта.

Из технологических соображений целесообразно позволить администратору безопасности ужесточать требо-

вания к реализации в точке вызова ($IRE(P1)$) или описание особенностей реализации метода ($IFE(M)$). Подобное ужесточение может использоваться при выявлении недостатков реализации, чреватых нарушением ИБ. Таким образом, в общем случае требования к реализации представляются в виде

$$IRE(P1) = AIRE(P1) \ \& \sim NIRE(P1),$$

где $AIRE$ — авторские требования, а $NIRE$ — так называемые негативные спецификации, добавляемые администратором безопасности. Аналогичное соотношение имеет место для $IFE(M)$.

7. Второй уровень конкретизации — реализация традиционных моделей разграничения доступа

Описанный выше общий формализм, очевидно, может быть применен для реализации привычных моделей управления доступом.

7.1. Реализация дискреционной модели

Дискреционную модель можно реализовать, если в ПРД будет входить уникальный идентификатор пользователя, ассоциированный с выполняемым методом. С формальной точки зрения пользователь может рассматриваться как объект, внешний по отношению к контролируемому контейнеру. Идентификацию и аутентификацию пользователей могут проводить специальные объекты — роли, задающие первоначальные ограничения на семантику и реализацию последующих вызовов.

Для реализации дискреционной модели должны в той или иной форме поддерживаться списки управления доступом к методам объектов. Почти все предикаты, входящие в выражение (1), будут пустыми, лишь $ARE(P1)$ примет вид

$$CU(M1) \in ACL(M),$$

где $CU(M1)$ — идентификатор пользователя, ассоциированный с вызывающим методом, а $ACL(M)$ — множество пользователей, имеющих право вызывать метод M.

Если необходимо разграничивать объекты по степени надежности создавшего или распространяющего их источника, можно включить в ПРД уникальные идентификаторы автора и/или распространителя. По сути это будет лишь минимальным усложнением описанной реализации дискреционной модели.

7.2. Реализация мандатной модели

Для реализации мандатной модели управления доступом целесообразно рассмотреть информационные потоки, возникающие при вызовах методов (см. разд. 4). С каждым объектом ассоциируются две метки безопасности — текущая (обозначим ее $LC(O)$) и максимальная ($LM(O)$). В любой момент времени должно выполняться соотношение

$$LC(O) \leqslant LM(O).$$

Первоначально метки $LC(O)$ могут устанавливаться в минимально возможные значения.

В момент приема вызова текущая метка объекта O должна стать равной

$$\max(LC(O),\ LC(IAP1),\ ...,\ LC(IAPn))$$

где $IAPi$ — входные параметры вызываемого метода объекта O, а операция взятия максимума понимается в естественном для решеток смысле. Вызов возможен, если новое значение $LC(O)$ не превосходит $LM(O)$.

В процессе выполнения метода текущая метка объекта может изменяться (увеличиваться). При завершении вызова для всех выходных параметров $OAP1, ..., OAPm$ должно выполняться соотношение

$$LC(O) \leqslant LM(OAPj),\ j = 1, ..., m.$$

Если это условие выполнено, завершение вызова возможно, а новыми значениями текущих меток выходных параметров становятся выражения

$$\max(LC(O),\ LC(OAPj),\ j = 1, ..., m.$$

Описанные выше условия естественно включить в политику безопасности $SP(CC)$.

7.3. Реализация модели песочницы

Для реализации модели песочницы достаточно наложить ограничение на множество объектов, вызываемых компонентами контейнера. Это ограничение естественно оформить как политику безопасности контейнера. Часть доступных объектов задается статически, другие по определенным правилам ассоциированы с компонентами контейнера (например, web-сервер, с которого поступил данный аплет).

Отметим, что если к подобной политике безопасности добавить добровольные ограничения на семантику ($SFE(M)$) и реализацию ($IFE(M)$), получим так называемую "естественную песочницу", границы которой определяются не только политикой безопасности, но и естественными рамками поведения методов объектов.

Заключение

В данной работе сделана попытка увязать постановку задачи разграничения доступа с современным состоянием информационных технологий и, в частности, с объектно-ориентированным подходом.

Предложенную постановку можно считать развитием и обобщением существующих подходов к задаче разграничения доступа, в первую очередь модели фильтрации и программирования по контракту. Подобный подход целесообразно реализовать и применить в рамках концепции контролируемого выполнения [7].

Авторы признательны Валерию Александровичу Васенину за внимание к работе и ценные замечания.

Список литературы

1. **Бетелин В. Б., Галатенко В. А.** ЭСКОРТ — инструментальная среда программирования. Юбилейный сборник трудов институтов Отделения информатики РАН. Москва, 1993, Том. II. 21 с.

2. **Галатенко В. А.** Информационная безопасность — практический подход / Под ред. В. Б. Бетелина. М.: Наука, 1998, 301 с.

3. **Гостехкомиссия России.** Руководящий документ. Концепция защиты СВТ и АС от НСД к информации. Москва, 1992.

4. **Gong L.** Java 2 Platform Security Architecture — Oracle on-line documentation. URL: http://docs.oracle.com/javase/1.4.2/docs/guide/security/spec/security-spec.doc.html.

5. **Tennenhouse D. L., Wetherall D. J.** Towards an Active Network Architecture // ACM SIGCOMM Computer Communication Review. 2007. Vol. 37. Is. 5. P. 81—94.

6. **Dahlgren T. L.** Performance-Driven Interface Contract Enforcement for Scientific Components. Dissertation submitted in partial satisfaction of the requirements for the degree of Doctor of Philosophy in Computer Science. Lawrence Livermore National Laboratory, UCRL-TH-235341. 2008. 142 p.

7. **Бетелин В. Б., Галатенко В. А., Костюхин К. А.** Основные понятия контролируемого выполнения сложных систем // Приложение к журналу "Информационные технологии". 2013. № 3. 32 с.

УДК 519.6

А. Н. Коварцев, д-р техн. наук, проф., зав. каф.,
Д. А. Попова-Коварцева, аспирант,
Самарский государственный аэрокосмический университет им. акад. С. П. Королёва (Национальный исследовательский университет),
e-mail: kovr_ssau@mail.ru

Структурная оптимизация управляющего графа на основе алгоритма топологической сортировки

В данной работе алгоритм топологической сортировки направленного графа используется в качестве основной эвристики для построения приближенного алгоритма структурной оптимизации управляющего графа программы в целях снижения количества тестируемых маршрутов программы.

Особенность предлагаемого обобщенного алгоритма топологической сортировки заключается в том, что он работает не только с бесконтурными направленными графами, для которых известно большое число эффективных алгоритмов, но и с графами, содержащими контуры.

Ключевые слова: *структурная оптимизация, граф управления алгоритма, алгоритм топологической сортировки, глобальная оптимизация*

Введение

Решения задач оптимизации, верификации, тестирования, распараллеливания и параллельного программирования часто связаны с использованием алгоритмов потокового анализа программ. Потоковый анализ является средством получения достоверной информации о поведении программы без ее реального исполнения [1].

Часто потоковый анализ связан с анализом потока управления, когда исходная программа описывается упрощенной моделью — схемой программы, обычно представленной в виде графа управления (**уграфа**). Многие задачи создания и обработки программы существенно упрощаются, если программа имеет регулярную структуру, допускающую представление в виде иерархии вложенных фрагментов специального вида. В частности, структурная оптимизация программы всегда сводится к некоторым эквивалентным преобразованиям программы, приводящим ее к виду, сокращающему время выполнения программы, размер исходного кода, объем используемой памяти, структурную сложность программы и т. д.

В данной работе рассматривается пример использования обобщенного алгоритма топологической сортировки контурного уграфа для решения задачи снижения структурной сложности оптимизируемой программы.

Основные положения

Оптимизация структурной сложности программы в значительной степени зависит от выбранных метрик и мер оценки их качества. Рассмотрим метрики оценки структурной сложности программ, связанные с решением задачи их тестирования.

Наиболее известной в этой области является цикломатическая мера Мак-Кейба [1], основанная на идее оценки сложности программы по числу базисных путей в уграфе программы, комбинирование которых позволяет построить всевозможные пути из входа графа в выход. Данная мера сложности программы соответствует интуитивному пониманию этого понятия

и легко вычисляется по формуле $\lambda(G) = m(G) - n(G) + \rho(G)$, где $m(G)$ — число дуг орграфа G; $n(G)$ — число вершин графа; $\rho(G)$ — число компонент связности орграфа G. Но она нечувствительна к размеру программы и к изменению ее структуры.

На практике чаще используют структурные критерии [2], обеспечивающие покрытие каждой вершины графа (критерий **C0**), каждой дуги графа (**C1**) или наибольшего числа путей (маршрутов), соединяющих входную и выходную вершины графа (**C2**). Более результативным с точки зрения полноты тестирования программы является критерий **C2**. Однако для графов, содержащих контуры (циклы), число возможных маршрутов бесконечно. Этого недостатка лишена метрика, применяемая в технологии графосимволического программирования [3] и использующая обобщающее понятие *схемы маршрута*. В работе [3] доказано, что для конечного графа число схем маршрутов из входной вершины в выходную всегда конечно, независимо от наличия или отсутствия контуров в графе. Понятие схемы маршрута можно интерпретировать как маршрут в обычном понимании, из описания которого удалены все повторные вхождения пройденных вершин. Для бесконтурных графов понятия схемы маршрута и маршрута полностью совпадают. В общем случае схема маршрута определяет подграф исходного графа, состоящего только из вершин, входящих в схему маршрута.

Снижения структурной сложности графа можно добиться путем его иерархической декомпозиции на несколько вложенных графов с меньшей суммарной структурной сложностью. На рис. 1 представлен пример структурной декомпозиции графа G (число схем маршрутов — 10) на два иерархически связанных графа (рис. 1, *б*), эквивалентных исходному, полученных в результате разрезания графа G. Число схем маршрутов для подграфов: $G1$ — 4, $G2$ — 3.

Для задач структурной оптимизации характерна проблема, связанная с выбором способа определения места разреза, дающего наилучший результат с точки зрения уменьшения суммарной структурной сложности графа.

Решению задачи снижения структурной сложности графа помогает введение частичного порядка на множестве вершин графа, т. е. введение правильной нумерации его вершин, когда большинство дуг ведут от вершины с меньшим номером к вершинам с большим номером. В этом случае вершины графа можно расположить на одной линии (рис. 1, *в*) в порядке правильной нумерации и ввести числовую ось для переменной x, определяющей место "разреза" графа.

Для каждого из целочисленных значений переменной x можно вычислить значение критерия оценки структурной сложности разбиения графа на N частей:

$$K(x_1, ..., x_N) = \sum_{j=1}^{N} C_j,$$

где C_j — структурная сложность j-го подграфа.

Пусть $x \in [0; b]$. Введем дополнительные переменные $y_1, ..., y_{N-1} \in [0; 1]$, тогда линии разрезов исходного графа G на N частей вдоль оси x можно вычислить через переменные y_i следующим образом:

$$\begin{cases} x_1 = by_1, \\ x_2 = x_1 y_2, \\ \ldots \\ x_{N-1} = x_{N-2} y_{N-1}. \end{cases}$$

Причем для переменных x_i, реализующих разрезание графа на N частей, автоматически выполняются условия $x_{N-1} < ... < x_2 < x_1$.

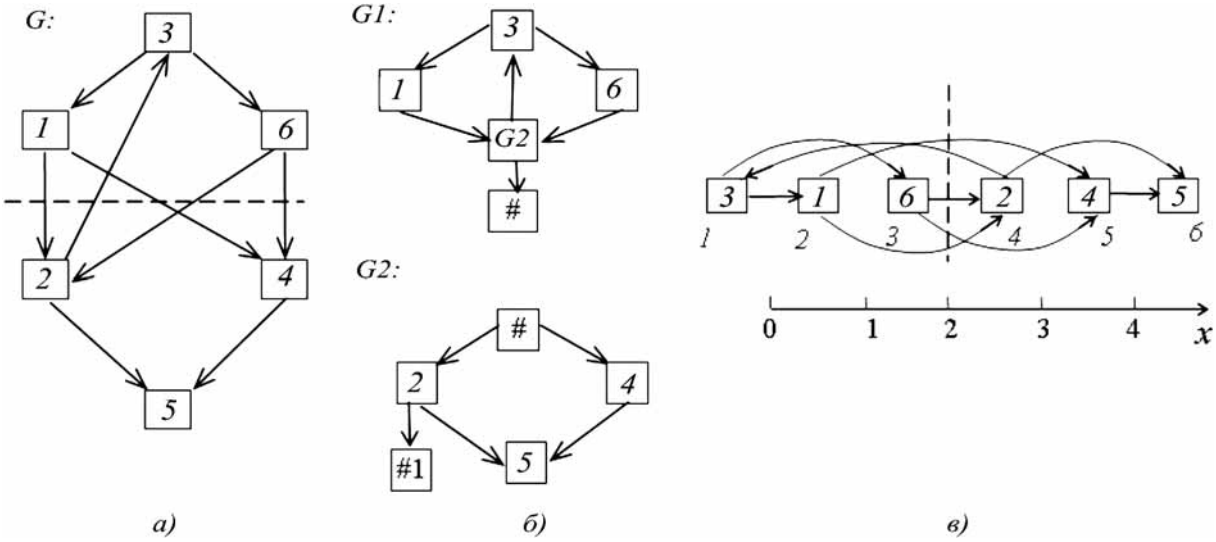


Рис. 1

Тогда задачу структурной оптимизации графа можно поставить как задачу глобальной оптимизации:

$$\min_{y_1, \ldots, y_{N-1}} K(y_1, \ldots, y_{N-1}).$$

Однако для реализации этой идеи требуется построить эффективный алгоритм правильной нумерации (топологической сортировки) произвольного графа, содержащего в том числе и контуры. Следует отметить, что для задачи структурной оптимизации графов топологическая сортировка графа является удобной эвристикой. Иначе пришлось бы рассматривать *NP*-полную задачу перебора всех вариантов разбиения графа на части.

Алгоритм топологической сортировки

Далее будем использовать алгебраическое определение ориентированного графа, заимствованное из работы [4], где под орграфом понимается пара $\vec{G} = (V, \alpha)$. Здесь V — множество вершин орграфа; $\alpha \subseteq V \times V$ — отношение на множестве V (пара $(u, v) \in \alpha$ называется дугой графа).

В графе, являющимся моделью некоторой программы, выделим две вершины: входную, с которой начинается вычислительный процесс, и выходную, в которой заканчиваются все вычисления. Для определения факта принадлежности графа к классу направленных бесконтурных графов удобно использовать теорему 3.33 работы [4]. Теорема утверждает, что в орграфе $\vec{G}$ существует правильная нумерация вершин тогда и только тогда, когда $\vec{G}$ — бесконтурный граф.

В данном случае под *правильной нумерацией* будем понимать такую нумерацию вершин графа $v_1, v_2, \ldots, v_n$, когда из $(v_i, v_j) \in \alpha$ следует неравенство $i \leqslant j$. Таким образом, проверка свойств орграфа на принадлежность классу классических Р-графов сводится к построению алгоритма правильной нумерации графа.

Однако в практических приложениях при построении моделей вычислительных алгоритмов используют не только классические бесконтурные схемы графа. В принципе, ургграф может содержать циклы. В этом случае предварительно необходимо провести расконтуривание графа за счет удаления из каждого контура графа по одной дуге. Теорема 3.36 из работы [4] позволяет проводить подобные операции без нарушения связности графа.

Введем понятие *ранга* вершины графа. Под рангом $r(v)$ *вершины* v графа $\vec{G} = (V, \alpha)$ будем понимать наибольшую из длин простых путей, начинающихся из входной вершины графа и заканчивающихся в вершине v.

Вершиной поглощения назовем вершину, в которую входят одна или несколько непомеченных дуг. *Вершиной ветвления* назовем вершину, в которую входят только помеченные дуги, и которая имеет по крайней мере две исходящие дуги. Статус "поглощения" будем считать более приоритетным по сравнению со статусом "ветвления". Здесь в качестве пометок рассматриваются ранги вершин, которыми в том числе размечаются и дуги, исходящие из них. Пометка означает, что данная вершина или дуга уже обработана алгоритмом.

Отметим важное свойство введенного понятия ранга вершины графа.

Лемма 1. *Вершины равного ранга двухполюсного бесконтурного направленного графа не связаны.*

Доказательство. Пусть вершины v и u двухполюсного направленного бесконтурного графа $\vec{G} = (V, \alpha)$ имеют одинаковые ранги:

$$r(v) = r(u) = r. \qquad (1)$$

Пусть вершины v и u соединены дугой $(v, u) \in \alpha$. По определению ранг вершины $r(v) = r$ означает, что из корневой вершины v_0 графа $\vec{G}$ до вершины v существует путь максимальной длины, равный r. Но тогда длина пути до вершины u по крайней мере не меньше, чем $r + 1$, поскольку v и u связаны дугой $(v, u) \in \alpha$. Следовательно, ранг вершины u равен $r + 1$, что противоречит условию (1).

Предлагаемый алгоритм нумерации вершин графа состоит из двух этапов. На первом этапе для всех вершин графа вычисляются их ранги. Для контурных графов проводится его расконтуривание. На втором этапе реализуется правильная нумерация его вершин.

Введем четыре списка: S_{out} — список вершин ветвления; S_{in} — список вершин поглощения; S_r — список вершин, которым присвоен ранг; S_L — список неиспользуемых (свободных) вершин ($S_L = V \backslash (S_{in} \cup S_{out} \cup S_r)$). Определим операции добавления и удаления из списков его элементов: $add(\cdot, \cdot)$ и $del(\cdot, \cdot)$ соответственно.

Алгоритм Р-нумерации.

Шаг 0. Фиктивной дуге (∞, v_0), входящей в начальную вершину графа, присваивается ранг, равный $r(\infty, v_0) = -1$. Если входная вершина имеет статус "поглощающей", то все входящие в корневую вершину дуги удаляются, а в качестве текущей вершины рассматривается вершина v_0. Переходим к шагу 1.

Шаг 1. Текущей вершине v_{i_k}, рассматриваемой на k-й итерации алгоритма, присваивается ранг

$$r\left(v_{i_k}\right) = \begin{cases} r\left(\left(v_{i_{k-1}}, v_{i_k}\right)\right) + 1, & \text{если } v_{i_k} \text{ — вершина ветвления,} \\ \max\limits_{\beta} r\left(\left(v_\beta, v_{i_k}\right)\right) + 1, & \text{если иначе.} \end{cases}$$

Если вершина v_{i_k} является вершиной ветвления, то выполняется операция пополнения списка S_{out}: $add(S_{out}, v_{i_k})$.

Шаг 2. Для текущей вершины v_{i_k} выбирается непомеченная дуга перехода в следующую вершину. Выбранная дуга помечается рангом вершины v_{i_k}: $r((v_{i_k}, v_{i_{k+1}})) = r(v_{i_k})$. Если все дуги, исходящие из этой вершины, помечены, то автоматически выполняется операция $del(S_{out}, v_{i_k})$.

Шаг 3. Анализируется статус вершины $v_{i_{k+1}}$, исходящей из v_{i_k} в направлении помеченной дуги: $(v_{i_k}, v_{i_{k+1}}) \in \alpha$.

Шаг 4. Если вершина $v_{i_{k+1}}$ не является вершиной поглощения, то в качестве текущей вершины рассматривается вершина $v_{i_{k+1}}$. Переходим к шагу 1.

Шаг 5. Иначе, при условии $v_{i_{k+1}} \notin S_{in}$, выполняется операция пополнения списка S_{in}: $add(S_{in}, v_{i_{k+1}})$. Если не все дуги, входящие в вершину $v_{i_{k+1}}$ помечены, то из списка S_{out} выбирается последний элемент $v_j \in S_{out}$, который рассматривается как текущий. Переходим к шагу 2.

Шаг 6. Если все дуги, входящие в вершину $v_{i_{k+1}}$ помечены, то выполняется операция $del(S_{in}, v_{i_{k+1}})$, и в качестве текущей вершины выбирается $v_{i_{k+1}}$. Переходим к шагу 1.

Шаг 7. Если список S_{out} пуст, а в списке S_{in} имеются элементы $S_{in} = [v_{i_1}, v_{i_2}, ..., v_{i_{k+1}}]$, то это означает, что в графе $\vec{G}$ имеются контуры, причем в циклических маршрутах участвует одна или несколько вершин списка S_{in}. Для идентификации вершин, принадлежащих контурам, для подграфа, составленного из вершин $V \setminus S_r$, строится двоичная булева матрица смежности $\mathbf{M}$ и матрицы ее степеней $\mathbf{M}^2$, $\mathbf{M}^3$, ..., $\mathbf{M}^k$, $k \leqslant n$ до тех пор, пока не возникнет ситуация $\exists j (m_{jj}^{(k)} = 1)$. Последнее означает, что в вершину v_j существует циклический маршрут, состоящий из k шагов. Составляется список $\tilde{S}_{in} = [v_{j_1}, v_{j_2}, ..., v_{j_m}] \subseteq S_{in}$, $m_{j_i j_i}^{(k)} = 1$, $i = 1, ..., m$. Из списка $\tilde{S}_{in}$ выбирается вершина v_β, имеющая наименьшее число не пройденных предшественников, которые не принадлежат списку S_r. При этом удаляемая дуга не должна нарушать принцип достижимости выходной вершины уграфа из любой его вершины. В графе удаляются все непомеченные дуги, входящие в вершину v_β, и выполняется операция $del(S_{in}, v_\beta)$, т. е. проводится расконтуривание графа. В качестве текущей вершины выбирается вершина v_β. Переходим к шагу 1.

Шаг 8. Если списки S_{out} и S_{in} пусты и достигнута концевая вершина, то переходим к шагу 9.

Шаг 9. Вершины орграфа нумеруются послойно от корневой вершины до концевой. Здесь под слоем графа будем понимать вершины равного ранга.

Утверждение 1. *Для двухполюсного орграфа алгоритм P-нумерации реализует правильную нумерацию его вершин.*

Доказательство. Первоначально рассмотрим случай бесконтурного двухполюсного графа. Описанный выше алгоритм реализует известную стратегию поиска в глубину с возвратом [1]. Движение "в глубину" реализуют шаги алгоритма 1—4. За механизм "возврата" отвечают шаги 5 и 6. Алгоритм останавливается в двух случаях: на шаге 8, но это штатное завершение обхода вершин графа, и на шаге 7, возникающем, когда в графе имеются контуры. Если в графе отсутствуют контуры, то все дуги графа будут помечены рангами, а за счет шага 1 всем вершинам будет присвоен ранг. Очевидно, что каждому двухполюсному бесконтурному графу соответствует единственная разметка вершин графа их рангами.

После разметки всех вершин графа на последнем шаге (шаг 9) алгоритма проводится послойная нумерация вершин графа. При этом всегда формируется правильная нумерация его вершин. Последнее верно, поскольку по лемме 1 между слоями графа (вершинами одинакового ранга) отсутствуют связи, а любой маршрут из корневой в концевую вершину двухполюсного направленного графа проходит поочередно через разные слои графа в направлении увеличения их рангов. В этом случае номера вершин будут всегда увеличиваться.

Предположим теперь, что граф $\vec{G} = (V, \alpha)$ содержит контур $v_{\beta_k} \to v_{\beta_1} \to v_{\beta_2} \to ... \to v_{\beta_k}$ и других кон-

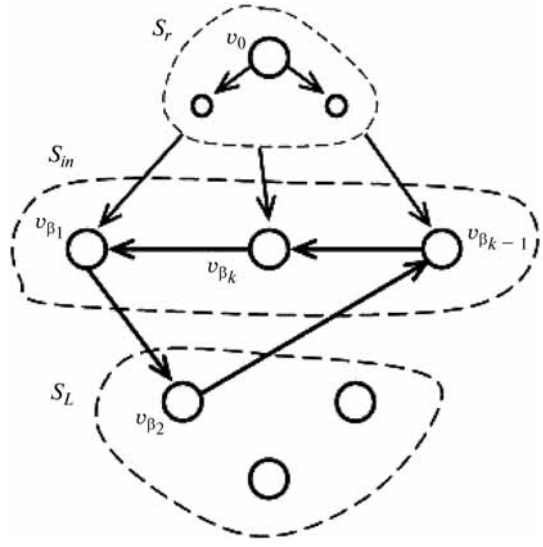


Рис. 2

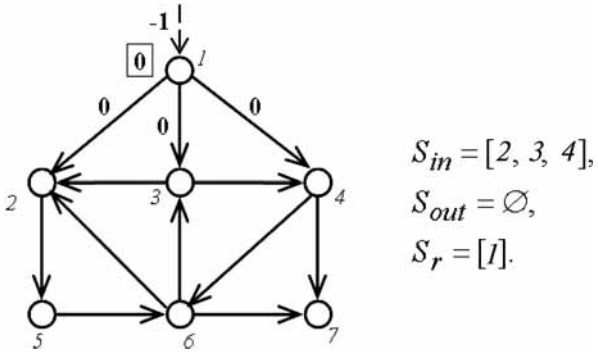


Рис. 3

туров в нем нет (рис. 2). Алгоритм на некотором шаге придет к ситуации, когда часть вершин v_{β_1}, v_{β_2}, ..., v_{β_k} войдет в список S_{in}, поскольку они имеют входящие, не размеченные дуги, т. е. являются вершинами поглощения, циклически ссылающимися друг на друга. Очевидно, что рано или поздно все размеченные вершины "упрутся" в вершины поглощения, без альтернативы продвинуться вглубь структуры графа, поскольку последние циклически ссылаются друг на друга. В результате возникнет ситуация, описанная в шаге 7. Если граф имеет несколько контуров, то список S_{in} может содержать вершины от разных контуров (рис. 3).

Для разрешения конфликтной ситуации на шаге 7 проводится разрыв контура по дуге, входящей в вершину поглощения. Далее алгоритм отработает в штатном режиме до момента обнаружения следующего контура или завершения этапа разметки вершин графа.

Пример работы алгоритма

Рассмотрим работу алгоритма на примере контурного графа, представленного на рис. 3. Разметка, приведенная на рис. 3, соответствует случаю, когда алгоритм обнаружил первый контур графа. Дуги *1—2*, *1—3*, *1—4* помечены рангом вершины *1*, из которой они исходят. На рисунке видно, что ранг равный нулю приобрела вершина *1* и размечены все исходящие дуги *1—2*, *1—3*, *1—4*.

$\mathbf{M}$ =

	2	3	4	5	6	7
2	0	0	0	1	0	0
3	1	0	1	0	0	0
4	0	0	0	0	1	1
5	0	0	0	0	1	0
6	1	1	0	0	0	1
7	0	0	0	0	0	0

$\mathbf{M}^2$ =

	2	3	4	5	6	7
2	0	0	0	0	1	0
3	0	0	0	1	1	1
4	1	1	0	0	0	1
5	1	1	0	0	0	1
6	1	0	1	1	0	0
7	0	0	0	0	0	0

$\mathbf{M}^3$ =

	2	3	4	5	6	7
2	**1**	1	0	0	0	1
3	1	**1**	0	0	1	1
4	1	0	**1**	1	0	0
5	1	0	1	**1**	0	0
6	0	0	0	1	**1**	1
7	0	0	0	0	0	0

Рис. 4

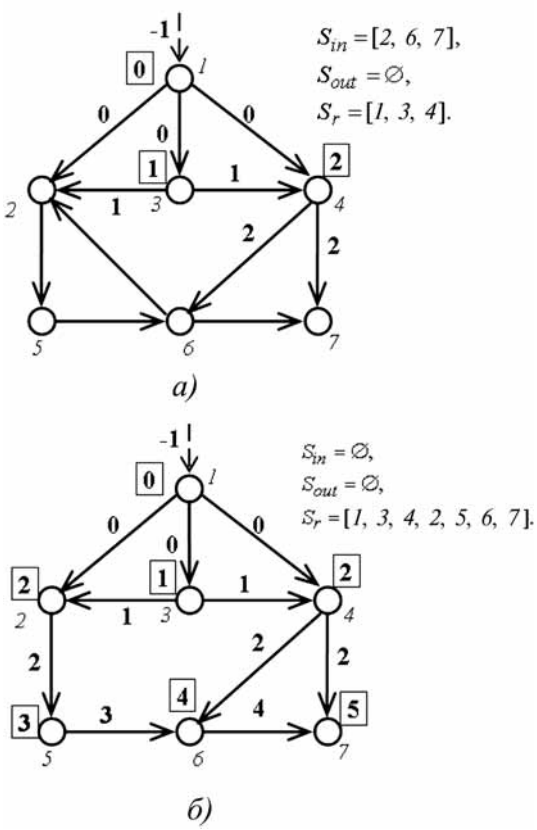


Рис. 5

$\mathbf{M}^3$ =

	2	5	6	7
2	**1**	0	0	1
5	0	**1**	0	0
6	0	0	**1**	0
7	0	0	0	0

Рис. 6

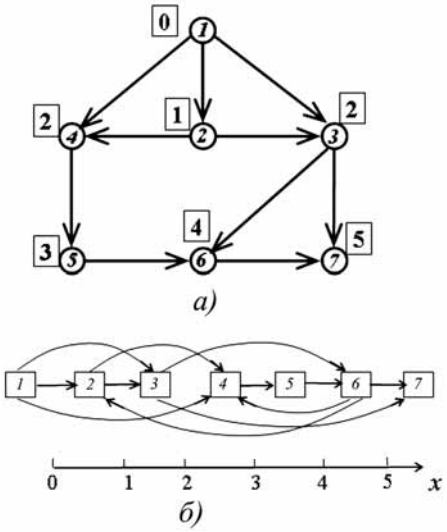


Рис. 7

Для определения контурных вершин на множестве $V \setminus S_r = \{2, 3, 4, 5, 6, 7\}$ построим матрицу смежности и матрицы ее степеней (рис. 4).

Из матрицы $\mathbf{M}^3$ видно, что в циклических маршрутах участвуют вершины *2*, *3*, *4*, *5*, *6*. Причем вершины *2*, *3*, *4* входят в список S_{in}, каждый элемент которого связан с циклами. Например, вершина *2* входит в цикл *2562*; вершина *3* — *3463* и *32563*; вершина *4* — *4634*. В качестве разрешающего элемента выберем вершину *3*, поскольку в нее входит только одна дуга, а в цикле участвуют две вершины (*3* и *4*). Из графа удалим дугу *6—3*. В результате получим разметку, представленную на рис. 5, *а*.

Алгоритм вновь "наталкивается" на контур *2562*, который определяется из матрицы, представленной на рис. 6.

Удалив дугу *6—2*, получим бесконтурный граф, представленный на рис. 5, *б* с окончательной разметкой вершин графа. Правильная нумерация графа представлена на рис. 7, *а*.

Линейная "развертка" уграфа показана на рис. 7, *б*. На рис. 8 построен график критерия оценки структурной сложности разрезания исходного графа $\bar{G}$ на две части, вычисленная с использованием цикломатической метрики Мак-Кейба. На рисунке видно, что оптимальное разделение графа на две части реализуется по линии разреза при $x = 1$, при этом число тестируемых маршрутов уменьшается в 6 раз.

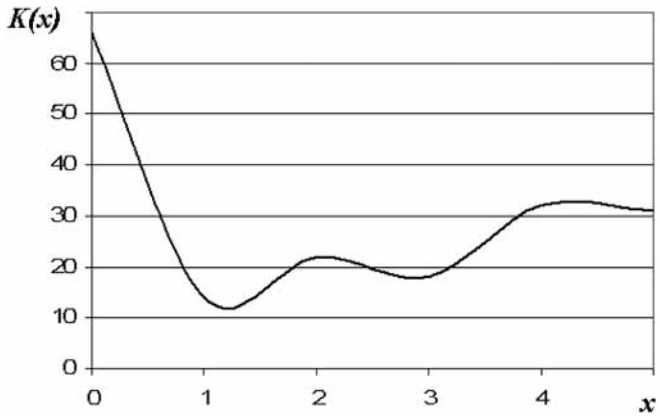


Рис. 8

Сложность алгоритма

Грубая оценка сложности предложенного алгоритма правильной нумерации двухполюсного направленного бесконтурного графа оценивается величиной $O(m)$, где m — число дуг графа. Однако основной особенностью данного алгоритма является то, что алгоритм позволяет проводить одновременно и расконтуривание графа. В этом случае трудоемкость алгоритма несколько увеличивается, поскольку использование стратегии построения транзитивного замыкания отношения достижимости вершин графа требует приблизительно $\tilde{n}^4$ бинарных операций. Здесь $\tilde{n}$ — число непомеченных вершин графа.

Заключение

В работе рассмотрен приближенный метод структурной оптимизации управляющего графа алгоритма, позволяющий существенно снизить трудоемкость тестирования разрабатываемой программы. Постановка задачи структурной оптимизации в значительной степени опирается на использование эффективного универсального алгоритма топологической сортировки управляющего графа, что существенно снижает сложность ее решения.

Универсальность алгоритма топологической сортировки заключается в том, что он позволяет выявлять контуры в уграфе, реализовывать их расконтуривание и осуществлять правильную нумерацию вершин графа.

Список литературы

1. **Касьянов В. Н., Евстигнеев А. В.** Графы в программировании: обработка, визуализация и применение. СПб.: БХВ—Петербург, 2003. 1104 с.

2. **Липаев В. В.** Отладка сложных программ. Методы, средства, технология. М.: Энергоатомиздат, 1993. 384 с.

3. **Коварцев А. Н.** Автоматизация разработки и тестирования программных средств. Самара: Самарский государственный аэрокосмический университет, 1999. 150 с.

4. **Богомолов А. М., Салий В. Н.** Алгебраические основы теории дискретных систем. М.: Наука, 1997. 368 с.

УДК 005

О. В. Бутырин, канд., техн. наук, доц.,
И. Е. Помогаев, аспирант кафедры,
Иркутский государственный университет путей сообщения,
e-mail: alz5@mail.ru

Применение метода решения задачи линейного программирования при распределении дефектоскопов

Рассмотрены вопросы методики планирования контроля состояния рельсов ручными средствами дефектоскопии относительно заданного интервала времени при помощи решения задач линейного программирования, учитывая выполнение ряда условий.

Ключевые слова: *дефектоскопия, планирование контроля, задача линейного программирования*

Общие сведения о дефектоскопии рельсов

Важнейшим элементом системы обеспечения безопасности движения поездов является центр диагностики пути при соответствующей службе на железной дороге, осуществляющий современный контроль состояния рельсов средствами дефектоскопии. Работа средств рельсовой дефектоскопии дистанции пути на основании приказа 2Ц3 [1] выполняется согласно графику, который ежемесячно разрабатывается руководителем участка дефектоскопии и утверждается начальником дистанции пути.

В соответствии с условными знаками в графической форме представляется планируемая ежедневная работа дефектоскопной автомотрисы, вагона дефектоскопа, съемных дефектоскопов сплошного контроля и стрелочных переводов. При этом должны учитываются следующие данные:

- число телег-дефектоскопов;
- интервалы времени между проверками;
- месячные нормы проверки рельсов и элементов стрелочных переводов средствами рельсовой дефектоскопии;
- выходные и праздничные дни;
- число станций и остановочных пунктов;
- последовательность проверки;
- проведение технической учебы;
- протяженность дистанции пути.

Необходимость автоматизации процесса составления графика заключается в сложности определения маршрутов передвижения дефектоскопов, их перевалочных пунктов и различных направлений следования при условии наложения всех ограничений и заданных интервалов в соответствии с нормативной документацией и более ранними графиками распределения. Тем более, компьютерная реализация позволяет выдавать несколько возможных вариантов распределения при различных исходных параметрах и любом заданном интервале времени.

В ходе исследования подходов к организации проведения проверок по дистанции Иркутск—Пассажирский, в качестве определяющих при постановке задачи рассматривались следующие данные.

Число телег-дефектоскопов зависит от интервала времени, в течение которого проводится проверка участка пути. К ним относятся следующие, используемые в разное время года дефектоскопные тележки: летом — Авикон-01 № 218, Авикон-01 № 309, РДМ-2 № 733, РДМ-2 № 741, РДМ-2 № 747, РДМ-2 № 877, зимой — эти же модели и РДМ-2 № 1274. Используются также дефектоскопы для проверки сварных стыков, а именно РДМ-3 № 258, РДМ-3 № 580. Резервными тележками при этом являются РДМ-1 М1 № 244, Поиск-10Э № 1557, РДМ-3 № 850. Резерв предусматривается на случаи выходов дефектоскопов из строя.

При планировании проверок необходимо учитывать то обстоятельство, что дефектоскопы не могут оставаться на перегонах и должны доходить до станций, на которых предусмотрено место хранения дефектоскопных тележек. После проверки участка дефектоскопной тележкой типа Авикон-01, следующая проверка в обязательном порядке проводится дефектоскопной тележкой типа РДМ. Период времени между проверками одного и того же участка пути в под-

лежащий анализу исследуемый интервал времени, равный одному месяцу или 30 дням, должен составлять: при двухразовой проверке 14...16 дней; при трехразовой проверке 10...11 дней; при четырехразовой проверке 7...8 дней. Норма, определяющая прохождение расстояния дефектоскопной тележкой, не должна превышать расстояние, равное 7,7 км. Кроме того, на графике должны указываться дни технической учебы и профилактические осмотры дефектоскопов, а также их ремонт в Дорожной лаборатории дефектоскопии.

Как отмечалось выше, на определенных остановочных пунктах или станциях организуются пункты хранения тележек, так как расположение участков, проверяемых ручными средствами дефектоскопии, удалено относительно расположения цеха дефектоскопии. По этой причине целесообразнее планировать работу по проверке рельсового хозяйства ручными средствами дефектоскопии от пунктов их хранения на текущую дату. Одними из критериев эффективности подхода в этом случае являются время прибытия к месту проведения проверки очередного участка пути специалиста цеха дефектоскопии, а также время доставки технического средства проведения контроля состояния рельсового хозяйства. Таким образом, данную задачу можно рассматривать как многокритериальную с детерминированными данными.

Описание задачи распределения дефектоскопных тележек и ее решение

Задача распределения дефектоскопных тележек в виде графика с учетом всех параметров и условий приводит к необходимости анализа большого числа потенциально возможных решений. Проанализировать их методом простого перебора быстро и относительно дешево не удается. Для этого необходима покупка или аренда нескольких мэйнфреймов и распределенного программного обеспечения, а сама обработка данных может занять несколько недель. Как следствие, логично применение одного из методов математического программирования для решения задач с детерминированными данными: линейное программирование, нелинейное или целочисленное программирование [2].

Наиболее исследованным и широко применяемым на практике методом решения задач поиска оптимальных решений в статической постановке является линейное программирование (ЛП) [3]. В этом случае целевая функция линейна, а множество, на котором ищется экстремум целевой функции, задается системой линейных равенств и (или) неравенств. Дополнительными преимуществами перед выбором метода решения детерминированных задач в пользу метода ЛП являются следующие: данный метод помогает в достижении оптимального использования производственных факторов; этот метод помогает в предоставлении более совершенных инструментов для адаптации к меняющимся условиям; с использованием этого метода подчеркиваются "узкие места" производственных процессов, повышается качество принимаемых решений [4].

Если принять, что m — это число заданных участков пути, а n — число дней в заданном периоде, то учитывая отмеченные выше условия решаемой задачи целевая функция будет иметь следующий вид:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} r_{ij}y_{ij} \to \min. \qquad (1)$$

Здесь y_{ij} — подлежащие определению неизвестные переменные, равные 1, если дефектоскоп планируется для проверки в i-й день на j-м участке дистанции пути в течение заданного периода времени, и равные 0 в противном случае; r_{ij} — время следования ручного дефектоскопа в i-й день на j-й участок пути, подлежащий проверке.

При решении поставленной задачи будут использоваться две матрицы значений. Одна из них матрица с известными значениями коэффициентов (МК), вторая матрица — с неизвестными значениями переменных (НП). Соответствующие этим матрицам значения r_{ij} и y_{ij} используются в формуле (1). Для удобства представления полученных результатов матрицы МК и НП будут приводиться в табличной форме. В таблице число строк и столбцов равны, соответственно, n и m, строки соответствуют числу учтенных рабочих дней, а столбцы — числу участков пути.

Исследование вариантов решения задачи распределения дефектоскопных тележек при наличии различных исходных данных

Рассмотрим решение последовательности тестовых задач на выделенном интервале времени, равном 12 дням, и выделенной совокупности участков пути, равной 12 участкам, другие исходные данные изменяются относительно ограничений, налагаемых на среду окружения и на исследуемые данные.

Первая тестовая задача состоит в распределении дефектоскопов по соответствующим участкам дистанции пути на основе известных значений числа дефектоскопов и числа проверок по каждому участку. Условия постановки задачи приведены далее.

Исходные данные первой тестовой задачи

Число дефектоскопов d	4
Число участков m	12
Исследуемый интервал времени n, число дней	12
Число проверок по соответствующему участку дистанции пути на планируемый период g_j	{2, 3, 2, 2, 3, 2, 2, 2, 2, 2, 3, 2}

Введем следующее ограничение:

$$\sum_{j=1}^{m} y_{ij} \geqslant w_i, \quad i = \overline{1, n}, \qquad (2)$$

где w_i — минимальное количество ручных средств дефектоскопии, необходимое для проверки всех участков дистанции пути в i-й планируемый день. В рассматриваемом примере $w_i \leqslant d$. Учитывая, что участки

дистанции пути, подлежащие проверке, как правило, содержат два основных направления движения подвижного состава — четное и нечетное, то для рассматриваемой тестовой задачи выберем одно из направлений. В данной тестовой задаче параметр $w_i = 2$ для каждого из направлений следования.

Следующее ограничение позволяет учесть суммарное число проверок на всех обслуживаемых участках:

$$\sum_{i=1}^{n} y_{ij} = g_j, \quad j = \overline{1, m}. \qquad (3)$$

Здесь g_j — число проверок, которое необходимо провести на заданном участке пути в границах дистанции пути железной дороги в течение исследуемого интервала времени.

Если неравенство в ограничении (2) указывает на минимально необходимое число дефектоскопов, требующееся для проверки в i-й день плана графика, то ограничение (3) предполагает жесткое равенство. Последнее обусловлено тем обстоятельством, что минимальное число проверок регламентируется нормативными документами и его изменение в планируемый период привнесет еще большую неопределенность в рассматриваемую задачу.

Таким образом, общая постановка первой тестовой задачи состоит в следующем: при заданных параметрах задачи определить оптимальный план распределения дефектоскопов по участкам пути на основе решения задачи ЛП (1)—(3), при этом элементы матрицы МК установить равными 1 для выполнения граничных условий параметра y_{ij}.

Результаты расчетов первой версии задачи приведены в табл. 1.

Согласно постановке задачи выполнены все условия оптимальности и ограничений задачи ЛП (1)—(3). Однако продолжительность интервала времени между проверками на каждом участке не является равнопериодичной. Это не соответствует требованиям действующей нормативно-технической документации, регламентирующей порядок организации и проведения проверок состояния железнодорожного пути. Причина в том, что при заданном значении числа проверок на любом участке пути, уменьшение продолжительности временного интервала между первой и второй проверками неизбежно приведет к увеличению продолжительности интервала между второй и третьей проверками и наоборот.

Таблица 1

Результаты расчетов

День	Участок											
	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	1	0	0	0	1	0	0	0	0
2	0	0	0	0	0	0	0	0	1	0	1	0
3	0	0	0	1	1	0	0	0	0	0	0	0
4	0	1	0	0	0	0	0	1	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0	1	0
6	0	0	0	0	0	0	0	0	0	1	0	1
7	0	0	1	0	1	0	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	0	0	0	1
9	0	0	0	0	0	1	0	0	0	1	0	0
10	1	0	0	0	0	1	0	0	0	0	0	0
11	0	1	0	0	0	0	0	0	0	0	1	0
12	0	1	1	0	0	0	0	0	0	0	0	0

Для исключения отмеченного выше несоответствия изменим условия постановки задачи ЛП. Введем параметр q_j, определяющий периодичность проверки по каждому участку пути, измеряющийся в днях. Введем также параметр g_j, определяющий число проверок по соответствующему участку дистанции пути на исследуемый интервал времени. В соответствии с определением g_j, для рассматриваемого участка пути каждая из проверок проводится через временные интервалы одинаковой продолжительности, равные отношению продолжительности планируемого интервала n, к числу проверок g_j. Таким образом, вторая тестовая задача является модификацией предыдущей с учетом нового введенного параметра. Исходные данные второй тестовой задачи ЛП приведены ниже.

Исходные данные второй тестовой задачи

Число дефектоскопов d 4
Число участков m 12
Исследуемый интервал времени проверки n, число дней 12
Число проверок по соответствующему участку дистанции пути на исследуемый интервал времени g_j . . . . . {2, 3, 2, 2, 3, 2, 2, 2, 2, 3, 2}
Периодичность проверки по каждому участку пути в соответствии с числом проверок в течение исследуемого интервала времени q_j, число дней . . . . . {6, 4, 6, 6, 4, 6, 6, 6, 6, 4, 6}

Математически решение этой задачи повторяет решение первой тестовой задачи. Исключением является дополнительное ограничение на периодичность проверки, позволяющее обеспечить равномерное распределение каждой следующей проверки в соответствии с указанной продолжительностью интервала:

$$\sum_{i=l-q_j+1}^{l} y_{ij} = 1, \quad l = \overline{q_j, n}, \quad j = \overline{1, m}. \qquad (4)$$

Продолжительность интервала q_j зависит от количества проверок для соответствующего участка пути. Так, при двухразовой проверке участка железнодорожного пути в течение периода времени равного одному месяцу или 30 дням, продолжительность интервала между проверками составит 14...15 дней. Таким образом, в соответствии с установленными для каж-

дого участка нормативными требованиями, продолжительность интервала равна:

$$q_j = \begin{cases} 14, \text{ если } g_j = 2,\ j = \overline{1, m}; \\ 9, \text{ если } g_j = 3,\ j = \overline{1, m}; \\ 7, \text{ если } g_j = 4,\ j = \overline{1, m}. \end{cases} \quad (5)$$

В рассматриваемом примере планируемый период проверки n равен 12 дням, тогда:

$$q_j = \begin{cases} 6, \text{ если } g_j = 2,\ j = \overline{1, m}; \\ 4, \text{ если } g_j = 3,\ j = \overline{1, m}; \\ 3, \text{ если } g_j = 4,\ j = \overline{1, m}. \end{cases} \quad (6)$$

Тогда общая постановка второй тестовой задачи следующая: при заданных значениях параметров задачи определить оптимальный план распределения ручных дефектоскопов по участкам пути на основе решения задачи ЛП (1)—(4), при этом элементы матрицы МК установить равными 1 для выполнения граничных условий параметра y_{ij}. Результаты расчетов приведены в табл. 2.

Согласно постановке задачи выполнены все условия оптимальности и совместимости задачи ЛП (1)—(4).

Таблица 2

Результаты расчетов

День	Участок											
	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	1	0	0	0	0	1	0	0
2	0	1	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	1	0	0	1	0
4	0	0	0	1	0	0	0	0	0	0	0	1
5	0	0	0	0	1	1	0	0	1	0	0	0
6	1	1	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	1	1	0
8	0	0	1	0	0	0	0	0	0	0	0	1
9	0	0	0	0	1	0	0	1	0	0	0	0
10	0	1	0	1	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	1	0	1	0
12	1	0	0	0	0	1	0	0	0	0	0	0

Продолжительность интервала между проверками на каждом участке носит равномерный характер.

Однако последовательность прохождения дефектоскопных тележек должна плавно переходить от соответствующего участка на соседний участок, без разрывов по дням проверки, вплоть до места хранения дефектоскопов.

Для создания управляемого распределения плавного нисходящего прохождения дефектоскопных тележек по участкам дистанции пути необходимо внести изменения в матрицу МК. Задавая значения коэффициентов целевой функции, имеющих вид диагонального увеличения, например: $y_{1j} \ll y_{nm}$, можно добиться управляемого распределения элементов матрицы МК в соответствии с направлением вектора целевой функции.

В третьей тестовой задаче будем рассматривать вертикальное задание коэффициентов матрицы МК с диагональным смещением. Исходные данные третьей тестовой задачи идентичны исходным данным второй тестовой задачи.

Так как у каждой матрицы есть две диагонали, то необходимо рассматривать две разные МК с одинаковым интервалом, равным, например, 80, указанные матрицы приведены на рис. 1, 2 соответственно.

Таким образом решение третьей тестовой задачи включает две задачи ЛП с разными значениями матрицы МК. Введем следующие параметры: $d_{\text{ч}}$ — число дефектоскопов при расчете матрицы НП четного направления; $d_{\text{нч}}$ — число дефектоскопов при расчете матрицы НП нечетного направления. В нашем примере общее число дефектоскопов из условий постановки задачи равно четырем, а каждое из $d_{\text{ч}}$ и $d_{\text{нч}}$ равно двум.

Общая постановка третьей тестовой задачи следующая: при заданных параметрах задачи найти оптимальный план распределения дефектоскопов по участкам пути в четном направлении движения на основе решения задачи ЛП (1)—(4); на основе значений параметров задачи найти оптимальный план распределения дефектоскопов по участкам пути в нечетном направлении движения на основе решения задачи ЛП (1)—(4); на основе полученных оптимальных планов составить совмещенный план; элементы матриц МК приведены на рис. 1, 2.

Результаты решения для четного и нечетного направлений движения приведены в табл. 3 и 4 соответственно.

Совмещенный план проверки приведен в табл. 5, где значение "1" соответствует элементу матрицы НП четного направления, а значение "−1" — элементу матрицы НП нечетного направления.

Анализ табл. 5 показывает, что недостаток третьей версии задачи исключен не полностью. Следовательно, требуется разработка специализированных алгоритмов, позволяющих задавать предсказуемый порядок распределения в интервале соседних участков пути, находящихся между пунктами хранения дефектоскопов.

Введем дополнительное условие в постановку задачи для рассматриваемого примера, которое учитывает выходные и учебные дни, когда не ведется работа

по проверке рельсов на дефекты. Исходные данные четвертой версии задачи указаны ниже.

Исходные данные четвертой тестовой задачи

Число дефектоскопов d 4

Число участков m 12

Планируемый период проверки n, число дней. 12

Число проверок по соответствующему участку дистанции пути на планируемый период g_j. . . . . {2, 3, 2, 2, 3, 2, 2, 2, 2, 3, 2}

Периодичность проверки по каждому участку пути в соответствии с числом проверок за заданный интервал времени q_j, число дней. . . . . {6, 4, 6, 6, 4, 6, 6, 6, 6, 6, 4, 6}

Дни заданного интервала времени, когда не ведутся работы на участках v_i, номер дня . . . . . {6, 7}

Данная задача является модификацией третьей версии с ограничением того, что в дни, указанные в массиве V (который состоит из элементов v_i), не ведутся работы по проверке участков пути, или формально:

$$\sum_{j=1}^{m} y_{ij} = 0, \quad i \in V,\ i = \overline{1, n}. \qquad (7)$$

0	80	160	240	320	400	480	560	640	720	800	880
80	160	240	320	400	480	560	640	720	800	880	960
160	240	320	400	480	560	640	720	800	880	960	1040
240	320	400	480	560	640	720	800	880	960	1040	1120
320	400	480	560	640	720	800	880	960	1040	1120	1200
400	480	560	640	720	800	880	960	1040	1120	1200	1280
480	560	640	720	800	880	960	1040	1120	1200	1280	1360
560	640	720	800	880	960	1040	1120	1200	1280	1360	1440
640	720	800	880	960	1040	1120	1200	1280	1360	1440	1520
720	800	880	960	1040	1120	1200	1280	1360	1440	1520	1600
800	880	960	1040	1120	1200	1280	1360	1440	1520	1600	1680
880	960	1040	1120	1200	1280	1360	1440	1520	1600	1680	1760

Рис. 1. Матрица МК с диагональным смещением для проверки участков железнодорожного пути в четном направлении

880	800	720	640	560	480	400	320	240	160	80	0
960	880	800	720	640	560	480	400	320	240	160	80
1040	960	880	800	720	640	560	480	400	320	240	160
1120	1040	960	880	800	720	640	560	480	400	320	240
1200	1120	1040	960	880	800	720	640	560	480	400	320
1280	1200	1120	1040	960	880	800	720	640	560	480	400
1360	1280	1200	1120	1040	960	880	800	720	640	560	480
1440	1360	1280	1200	1120	1040	960	880	800	720	640	560
1520	1440	1360	1280	1200	1120	1040	960	880	800	720	640
1600	1520	1440	1360	1280	1200	1120	1040	960	880	800	720
1680	1600	1520	1440	1360	1280	1200	1120	1040	960	880	800
1760	1680	1600	1520	1440	1360	1280	1200	1120	1040	960	880

Рис. 2. Заданная матрица МК с диагональным смещением для проверки участков железнодорожного пути в нечетном направлении

Общая постановка четвертой тестовой задачи следующая: при заданных параметрах задачи определить оптимальный план распределения дефектоскопов по участкам пути в четном направлении движения на ос-

Таблица 3

Матрица НП четного направления

День	Участок											
	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	0	0	0	0	0	1	0	0	0	0
2	0	0	0	0	1	0	0	0	0	1	0	0
3	0	0	0	1	0	0	1	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0	0	0	0
5	0	1	0	0	0	0	0	0	0	0	1	0
6	1	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0	1	0	0	1	0	0	0	0
8	0	0	0	0	0	0	0	0	1	1	0	0
9	0	0	0	1	0	0	1	0	0	0	0	0
10	0	1	1	0	0	0	0	0	0	0	0	0
11	0	0	0	0	1	0	0	0	0	0	0	0
12	1	0	0	0	0	1	0	0	0	0	0	0

Таблица 4

Матрица НП нечетного направления

День	Участок											
	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	1	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	1	0	0	0	0
3	0	1	0	0	0	0	1	0	0	0	0	0
4	0	0	0	1	0	0	0	0	1	0	0	0
5	0	0	1	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	1	0	0	0	1	0	0
7	1	0	0	0	0	0	1	0	0	0	0	0
8	0	1	0	0	0	0	0	1	0	0	0	0
9	0	0	1	0	1	0	0	0	0	0	0	0
10	0	0	0	1	0	0	0	0	1	0	0	0
11	1	0	0	0	0	0	1	0	0	0	0	0
12	0	1	0	0	0	1	0	0	0	0	0	0

Таблица 5

Результаты расчетов

День	Участок											
	1	2	3	4	5	6	7	8	9	10	11	12
1	0,0	1,0	−1,0	0,0	−1,0	0,0	0,0	1,0	0,0	0,0	0,0	0,0
2	−1,0	0,0	0,0	0,0	1,0	0,0	0,0	−1,0	0,0	1,0	0,0	0,0
3	0,0	−1,0	0,0	1,0	0,0	0,0	1, −1	0,0	0,0	0,0	0,0	0,0
4	0,0	0,0	1,0	−1,0	0,0	0,0	0,0	0,0	−1,0	0,0	0,0	0,0
5	0,0	1,0	−1,0	0,0	−1,0	0,0	0,0	0,0	0,0	0,0	1,0	0,0
6	1,0	0,0	0,0	0,0	0,0	1, −1	0,0	0,0	0,0	−1,0	0,0	0,0
7	−1,0	0,0	0,0	0,0	1,0	0,0	−1,0	1,0	0,0	0,0	0,0	0,0
8	0,0	−1,0	0,0	0,0	0,0	0,0	0,0	−1,0	1,0	1,0	0,0	0,0
9	0,0	0,0	−1,0	1,0	−1,0	0,0	1,0	0,0	0,0	0,0	0,0	0,0
10	0,0	1,0	1,0	−1,0	0,0	0,0	0,0	0,0	−1,0	0,0	0,0	0,0
11	−1,0	0,0	0,0	0,0	1,0	0,0	−1,0	0,0	0,0	0,0	0,0	0,0
12	1,0	−1,0	0,0	0,0	0,0	1, −1	0,0	0,0	0,0	0,0	0,0	0,0

нове задачи ЛП (1)—(7); при заданных параметрах определить оптимальный план распределения дефектоскопов по участкам пути в нечетном направлении движения на основе задачи (1)—(7); на основе полученных оптимальных планов составить совмещенный план; элементы матриц МК остаются из предыдущей задачи (рис. 1, 2).

Результаты расчетов задач линейного программирования для четного и нечетного направлений движения приведены в табл. 6 и 7 соответственно.

Совмещенный план проверки приведен в табл. 8, где значение "1" соответствует элементу матрицы НП четного направления, а значение "−1" — элементу матрицы НП нечетного направления.

Таблица 6

Матрица НП четного направления

День	Участок											
	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	1	0	0	1	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	1	1
3	0	0	1	0	0	1	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	1	0	0
5	0	1	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	1
9	0	0	1	0	1	0	0	0	0	0	0	0
10	0	0	0	1	0	1	0	0	0	0	0	0
11	0	1	0	0	0	0	0	0	0	0	1	0
12	1	0	0	0	0	0	0	0	0	1	0	0

Таблица 7

Матрица НП нечетного направления

День	Участок											
	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	1	1	0	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0	0
3	0	0	0	0	0	0	0	1	0	0	1	0
4	1	0	1	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	1	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	1	1	0	0	0	0	0	0
10	0	0	1	1	0	0	0	0	0	0	0	0
11	1	0	0	0	0	0	0	1	0	0	0	0
12	0	0	0	0	0	0	0	0	0	1	1	0

Таблица 8

Полная матрица НП

День	Участок											
	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	−1	1, −1	0	0	1	0	0	0	0
2	0	−1	0	0	0	−1	0	0	0	0	1	1
3	0	0	1	0	0	1	0	−1	0	0	−1	0
4	−1	0	0	1, −1	0	0	0	0	0	1	0	0
5	0	1	0	0	1, −1	0	0	0	0	−1	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0	−1	0	0	0	0	0	1	0	0	−1	1
9	0	0	1	0	1, −1	−1	0	0	0	0	0	0
10	0	0	−1	1, −1	0	1	0	0	0	0	0	0
11	−1	1	0	0	0	0	0	−1	0	0	1	0
12	1	0	0	0	0	0	0	0	0	1, −1	−1	0

Выводы

Построенные модели позволят в режиме реального времени реагировать на нештатные ситуации посредством принятия управленческих решений в следующих задачах:

- прогнозирование различных вариантов составления плана графика проведения проверок средствами ручной дефектоскопии;
- на основе данных прогноза осуществлять определение требуемого числа дефектоскопных тележек или других средств ручной дефектоскопии;
- минимизировать общее время составления плана графика проведения проверок средствами ручной дефектоскопии;
- автоматизировать сквозной учет проведения проверок средствами дефектоскопии;
- повысить эффективность функционирования подразделений дефектоскопии за счет снижения временных, трудовых и материальных издержек.

Учитывая, что размер матриц МК и НП в случае решения рассмотренной задачи для реального объекта исследования будет составлять сотни тысяч элементов и более, то необходима разработка соответствующего специализированного программного обеспечения по обработке многомерных данных. В качестве инструментальных средств целесообразно использование систем управления базами данных, взаимодействующих с соответствующим программным комплексом, обеспечивающим поддержку многомерной обработки и анализа данных, являющихся одной из обеспечивающих ее подсистем.

Для последующей программной реализации рассмотренных задач будет применяться оптимальная связка программных средств: приложение win32, разработанное в объектно-ориентированной программной среде Delphi, и система управлением баз данных Mysql 5.0.

Список литературы

1. **Приказ** МПС РФ № 2-ЦЗ от 25 февраля 1997 г. "О совершенствовании системы контроля состояния рельсов средствами дефектоскопии". 28 с.

2. **Карманов В. Г.** Математическое программирование. М.: Наука, 1986. 288 с.

3. **Кормен Т. Х.** Алгоритмы: построение и анализ. 2-е изд. М.: Вильямс, 2006. 1296 с.

4. **Электронный** сборник научных статей. URL: http://loremate.com

УДК 519.683

К. Е. Селезнёв, канд. техн. наук, доц., Воронежский государственный университет,
e-mail: skostik@relex.ru

Полнотекстовый поиск в системе Бератор

Рассматривается подсистема полнотекстового поиска в информационно-справочной системе Бератор. Особенностью данной реализации является высокая функциональность при минимальных требованиях к аппаратным ресурсам.

Ключевые слова: *полнотекстовый поиск, морфологический анализ, мобильные устройства*

Программный комплекс Бератор представляет собой информационно-справочную систему для работы с информацией о правилах и особенностях ведения бухгалтерского учета. Такая информация хранится на локальном жестком диске и обновляется по сети Интернет. Большая часть информации представлена в виде набора текстовых документов. Одной из функций системы является полнотекстовый поиск. Данная статья содержит описание технической реализации подсистемы полнотекстового поиска, особенностями которой являются относительная простота программной реализации и механизмов ее использования при минимальных требованиях к ресурсам, которые для этого задействуются.

Постановка задачи

База данных системы Бератор представляет собой иерархически организованную коллекцию текстовых документов, представленных в формате HTML. Документы могут содержать различные виды форматирования, рисунки, таблицы и перекрестные ссылки. От подсистемы полнотекстового поиска требуется:

- выполнять поиск в коллекции документов, которые содержат заданные слова или их морфологические формы;
- выводить список поисковых подсказок, содержащих заданные слова или их морфологические формы.

Обе указанные задачи хорошо изучены [1] и реализованы на практике. Предлагаемые методы апробированы и используются во многих современных системах полнотекстового поиска. Однако система Бератор имеет свои важные особенности, к числу которых относятся следующие.

- Система должна работать на различных платформах, в том числе и мобильных устройствах, поэтому реализация подсистемы полнотекстового поиска должна предъявлять минимальные требования к вычислительным ресурсам. В частности, индексирование и любая другая ресурсоемкая работа по возможности должны проводиться на сервере.
- Система должна скачивать обновления и затем работать локально, не требуя наличия постоянного канала связи с центральным сервером. Это означает, что сам полнотекстовый поиск тоже должен проводиться локально, а вся необходимая для него информация должна скачиваться вместе с обновлениями.
- В целях минимизации сетевого трафика размер файлов обновлений должен быть минимальным. Информация, необходимая для полнотекстового поиска, не должна занимать большой объем памяти, поскольку это может быть критично для некоторых мобильных устройств. По той же причине установка обновлений не должна требовать больших вычислительных ресурсов.
- Словарь системы постоянно обновляется и содержит большое число различных специализированных терминов. Как следствие, должны быть предельно простые в использовании способы корректировки и пополнения словаря.
- Список поисковых подсказок должен быть настроен администраторами системы.

Перечисленные требования не позволяют в полном объеме применить без изменений традиционно используемые подходы к полнотекстовому поиску [1] и морфологическому анализу [2]. Далее описаны принятые в рассматриваемой подсистеме технические решения и полученные характеристики.

Морфологический анализ

Исходя из функциональных требований ко всей подсистеме полнотекстового поиска, можно сформулировать ключевые требования к модулям морфологического анализа. Во-первых, анализатор должен по

заданному слову определять текст основной словоформы (или возможных словоформ). При этом должны быть востребованы сами морфологические характеристики (граммемы). Во-вторых, такие требования должны быть ориентированы на корректную обработку всех общеупотребимых слов русского языка. В-третьих, должны быть предусмотрены простые в реализации возможности для внесения новых слов и терминов. В-четвертых, могут потребоваться выполнение обратного морфологического анализа и получение по основной словоформе всех остальных форм слова.

Было принято решение реализовать модули морфологического анализа на основе точной морфологии с использованием сразу двух словарей [2]. Первый из них основывается на словаре А. А. Зализняка [3] и содержит сведения об общеупотребимых словах русского языка. Он генерируется один раз и затем модифицируется только в том случае, если были обнаружены какие-либо ошибки или неточности. Второй словарь содержит описание всех обладающих определенной спецификой терминов, которые модифицируются администраторами системы по мере необходимости.

Указанную идею трудно реализовать в чистом виде, поскольку словарь общеупотребимых слов содержит более восьми миллионов значений. Это обстоятельство приводит к тому, что файл словаря становится слишком большим и работа с ним может быть затруднена на некоторых типах мобильных устройств. В связи с этим было принято решение разделить словарь общеупотребимых слов на 33 части, каждая из которых содержит все слова на одну и ту же букву: первая — на букву "а", вторая — на букву "б" и т. д. Считается, что буквы "ь", "ъ" и "ы" не могут быть первыми в слове, и потому получилось 30, а не 33 файла словаря.

Далее было использовано одно очень важное свойство русского языка: в большинстве случаев все словоформы одного слова имеют общий префикс-основу и различаются только окончаниями. Благодаря этому свойству все формы одного и того же слова в большинстве случаев окажутся внутри одного и того же из 30 файлов. Слова-исключения, у которых формы не имеют общего префикса, были вынесены в отдельный файл, и, таким образом, число файлов стало равным 31. Наконец, было решено, что термины потенциально тоже могут изменяться и иметь различные морфологические формы без общего префикса-основы. Таким образом, общее число словарей системы стало равным 32.

Все словари хранятся во внутреннем формате, обеспечивающем высокую скорость поиска слова и минимальные требования к объему памяти. Построение словарей осуществляет специальный модуль, который на входе принимает текстовый файл словаря, а на выходе выдает файл словаря во внутреннем формате. Каждая строка исходного файла содержит написание всех форм (через пробел) одного и того же слова, при этом первая форма считается основной. Такое представление предельно просто и понятно для пользователя, осуществляющего корректировку словаря.

Внутренний формат файла основан на представлении словаря в виде множества пар "ключ—значение", где ключом является текст основы (неизменяемого префикса) слова, а значением — множество всех форм этого слова. Первая из указанных форм считается основной. Для каждой формы слова, в том числе и для первой, хранится только текст, добавляемый к основе. В словаре безосновных слов для каждой формы слова хранится число букв, которые берутся из основы, и добавляемое окончание. Хранение пар вида "ключ—значение" основано на хэш-таблицах, а для разрешения коллизий используются бинарные деревья поиска [4].

Морфологический анализ слова происходит в несколько итераций. На первой итерации в качестве основы берется все исходное слово, на второй итерации — исходное слово без последнего символа, на третьей — исходное слово без двух последних букв и т. д. Для каждого варианта основы вычисляется значение хэш-функции, а затем просматривается бинарное дерево поиска и ищутся все слова с данной основой. Для каждой найденной основы просматриваются все ее словоформы, сравниваются с исходным словом, и в случае совпадения в накапливаемый результат анализа добавляется основная словоформа. По той же схеме происходит синтез всех словоформ указанного слова — разница лишь в том, что в результат добавляется не только основная, а абсолютно все словоформы.

Полнотекстовый индекс

Индекс полнотекстового поиска строится по принципу инвертированного индекса [1], что обеспечивает высокую степень компрессии данных при скачивании обновлений по сети Интернет. Однако для достижения высокой скорости работы и снижения нагрузки на процессор, что особенно важно для мобильных приложений, после скачивания обновлений индексные файлы хранятся на локальном диске в распакованном виде.

Внутреннее устройство индекса полнотекстового поиска основано на хранении множества пар вида "ключ—значение", где ключом является текст слова, а значением — номера документов, в которых встречается данное слово. Хранение указанного множества пар осуществляется с помощью бинарных деревьев поиска. В индекс заносятся только основные словоформы. Если в документе встречается некоторая неосновная форма слова, то с помощью морфологического анализа она сначала приводится к основной и только потом добавляется в индекс.

По аналогии со словарями морфологического анализа индекс полнотекстового поиска разбит на 31 файл, из которых первые 30 соответствуют буквам русского алфавита — содержат слова, начинающиеся на эти буквы. Последний файл содержит слова, которые начинаются не на букву русского алфавита, а на какой-либо другой символ.

Процедура построения полнотекстового индекса на входе принимает следующую информацию: кол-

лекция индексируемых документов; морфологические словари и список слов исключений, которые не нужно добавлять в индекс. В ходе построения индекса система просматривает все текстовые документы, извлекает из них все встречающиеся слова, приводит их к основной форме, и если она не указана в списке слов-исключений, то полученная основная форма добавляется в индекс. Если сканируемое слово отсутствует в словаре, то оно считается своей основной словоформой и без изменений добавляется в индекс, а также в специальный файл новых слов. Этот файл просматривает администратор системы и найденные при сканировании слова добавляет либо в словарь морфологического анализа, либо в список слов-исключений.

Полнотекстовый поиск в системе Бератор осуществляется на основе рассмотренных выше индексных файлов, которые для каждого слова (точнее, его основной словоформы) содержат номера документов, в которых хотя бы один раз встречается данное слово или его словоформа. Система никак не учитывает ни взаимное расположение слов, ни частоту их встречаемости.

Если в поисковой строке вводится несколько слов, то для каждого из них проводится отдельный полнотекстовый поиск и формируется множество документов. Пересечение полученных множеств дает результат полнотекстового поиска всей фразы.

Таким образом, алгоритм полнотекстового поиска состоит их следующих шагов.

1. Формируются варианты основной формы каждого слова из поисковой строки.

2. Для каждой основной словоформы каждого слова выполняется полнотекстовый поиск и определяется множество документов.

3. Для каждого слова исходной фразы происходит объединение результатов поиска, полученных для всех словоформ этого слова.

4. Строится пересечение всех множеств документов, полученных на предыдущем шаге.

Здесь важно сделать одно замечание. Индексы полнотекстового поиска не содержат сведений о расположении искомых слов в документе. Тем не менее число документов, которые показываются пользователю, позволяет прочитать каждый из них и найти в нем фрагменты, где встречаются исходные слова. При открытии документа можно еще раз прочитать его содержимое и уже при показе выделить все словоформы всех слов исходной поисковой фразы.

Поисковые подсказки

Список поисковых подсказок, который задает администратор системы, входит в состав файла обновлений. Для каждой поисковой подсказки явно указывается список документов, которые необходимо отобразить. Иными словами, если пользователь выбирает поисковую подсказку, то полнотекстовый поиск как таковой не проводится, а происходит отображение заранее составленного списка документов.

Алгоритм формирования списка предлагаемых пользователю поисковых подсказок также достаточно прост и похож на рассмотренный выше алгоритм полнотекстового поиска. Если пользователь ввел несколько слов, то ему предлагаются поисковые подсказки, в которых встречаются все введенные слова или их словоформы.

Технические характеристики

В табл. 1 показаны размеры морфологического словаря системы. Данные приведены для каждого файла словаря отдельно.

В сумме все указанные файлы занимают чуть меньше 13 Мбайт. Это небольшой объем даже по меркам мобильных устройств, что позволяет все словари морфологического анализа загрузить в память сразу же после запуска системы. На момент написания статьи в системе было 3486 документов, в сумме занимающих 75 Мбайт памяти. В табл. 2 приведены размеры каждого файла индекса полнотекстового поиска.

Суммарный размер полнотекстового индекса без учета сжатия составляет чуть больше 1 Мбайта, что в сумме с морфологическим словарем дает меньше, чем 15 Мбайт, и легко умещается в памяти даже самых простых мобильных устройств.

Таблица 1

Файл для буквы	Размер, байт	Файл для буквы	Размер, байт
а	241813	р	806832
б	346578	с	1050213
в	1004945	т	335603
г	254746	у	483321
д	520474	ф	150444
е	54094	х	127585
ё	32784	ц	80678
ж	87787	ч	120334
з	810581	ш	165019
и	411417	щ	56220
й	34529	ы	32795
к	539392	э	120470
л	189200	ю	40646
м	355248	я	55502
н	784981	Безосновные слова	32880
о	1274815	Словарь терминов	34223
п	2676977		

Таблица 2

Файл для буквы	Размер, байт	Файл для буквы	Размер, байт
а	22136	п	189631
б	31433	р	64186
в	70783	с	110624
г	20402	т	35674
д	65236	у	43210
е	5155	ф	19393
ё	16	х	5554
ж	4158	ц	5711
з	38415	ч	12451
и	36440	ш	3809
й	160	щ	137
к	43437	ы	16
л	13646	э	15364
м	34829	ю	880
н	79405	я	4899
о	101432	Остальные слова	5398

Архитектура и реализация

Рассматриваемая подсистема полнотекстового поиска имеет распределенную архитектуру и состоит из серверной и клиентской частей.

Серверная часть включает модули, которые участвуют в сборке файлов обновлений и проводят формирование файлов полнотекстового индекса. Эта операция включает в себя сканирование всех документов, лексический анализ, морфологический анализ и собственно построение полнотекстового индекса.

Клиентская часть подсистемы включает модули, позволяющие читать файлы морфологического словаря и файлы полнотекстового индекса, а также модули, выполняющие полнотекстовый поиск согласно изложенным выше алгоритмам. Клиентская часть системы реализована для различных платформ.

Серверная и клиентская части системы не взаимодействуют непосредственно друг с другом. Это обстоятельство является принципиальным отличием от классической архитектуры "клиент—сервер", позволяющим использовать систему в режиме off-line.

* * *

В статье рассмотрена реализация морфологического анализа и полнотекстового поиска в информационно-справочной системе Бератор. Предлагаемое решение можно справедливо считать примитивным по сравнению с поисковыми машинами в сети Интернет или поисковыми модулями иных систем, ориентированных на обработку большого объема текстовых документов. Однако простота предложенного в статье решения имеет важные достоинства, поскольку предъявляет минимальные требования к используемым вычислительным ресурсам. Полученные модули полноценно функционируют на мобильных устройствах и полностью удовлетворяют потребностям конечных пользователей.

Список литературы

1. **Маннинг К. Д., Рагхаван П., Шютце Х.** Введение в информационный поиск: Пер. с англ. М.: Вильямс, 2011. 528 с.
2. **Ножов И. М.** Реализация автоматической синтаксической сегментации русского предложения. дис. ... канд. техн. наук. М.: РГГУ, 2003.
3. **Зализняк А. А.** Грамматический словарь русского языка. Словоизменение. 3-е изд. М.: Русский язык, 1987.
4. **Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К.** Алгоритмы: построение и анализ = Introduction to Algorithms / Под ред. И. В. Красикова. 2-е изд. М.: Вильямс, 2005. 1296 с.

ИНФОРМАЦИЯ

CONTENTS

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4

Дизайнер *Т. Н. Погорелова*. Технический редактор *Е. М. Патрушева*. Корректор *Е. В. Комиссарова*

Сдано в набор 04.03.2013 г. Подписано в печать 22.04.2013 г. Формат 60×88 1/8. Заказ PI513
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
105120, г. Москва, ул. Нижняя Сыромятническая, д. 5/7, стр. 2, офис 2.