ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г.

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

СОДЕРЖАНИЕ

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

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

Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

№ 2
February
2014

Published since September 2010

CONTENTS

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

УДК 004.4

В. В. Липаев, д-р техн. наук, проф., гл. науч. сотр., Институт системного программирования РАН, г. Москва,
e-mail: vlip28@mail.ru

Управление конфигурацией сложных комплексов программ реального времени

На основе анализа положений существующих стандартов и опыта автора, полученного в ходе разработок больших комплексов программ реального времени, кратко излагаются методические вопросы организации управления конфигурацией при коллективном производстве таких комплексов. Особое внимание обращается на обеспечение качества и тестирование программных продуктов, их документирование, архивирование и поставку пользователям.

Ключевые слова: *программные комплексы, программные продукты, реальное время, тестирование, архивирование*

V. V. Lipaev

Advanced Real-Time Software Systems Configuration Management

The methodical issues of the configuration management organization within the collective production of big real-time software systems are briefly outlined. The outline is based on the analysis of standards and author's experience obtained while developing such systems. Special attention is paid to the quality assurance and software testing, as well as documenting, archiving and contribution to an end-user.

Keywords: *software systems, software, real time, testing, archiving*

Введение

Управление конфигурацией — **существенная часть программной инженерии,** ***определяющая качество больших и сложных программных продуктов реального времени. Цель управления конфигурацией (УК)*** программных комплексов (ПК) и систем, состоящих из многих компонентов (единиц конфигурации — ЕК), каждый из которых может иметь разновидности или версии, заключается в том, чтобы обеспечить контролируемое развитие их структуры, состава компонентов и функций, а также сократить число дефектов в течение всего жизненного цикла ПК. В процессе организации такого управления необходимо построить и использовать компактные и наглядные ***схемы однозначной иерархической идентификации и изменения компонентов ПК***, в том числе:

- объектов — модулей и компонентов ПК разного уровня интеграции, подвергающихся модификации;
- корректировок содержания и взаимодействия проводимых изменений, которые должны обеспечивать возможность однозначного контроля, истории развития модификаций компонентов любого уровня во времени и в пространстве элементов версий комплекса программ (типы, содержание и взаимосвязь корректировок);
- специалистов, участвующих в УК, и сокращении числа обнаруживаемых при этом дефектов, их права на доступ к определенным компонентам ПК и документам на конкретных стадиях сопровождения, реализации и утверждения изменений.

Стандартизация процессов УК наиболее активно используется в производстве программных продуктов для критически важных систем [1], в том числе ракетно-космических, а также систем для авиации и др. Примером может служить широко применяемый международный стандарт **DO-178B** для бортовых авиационных систем и систем другого назначения. В стандарте около 15 % текста посвящено подробному описанию основных (двенадцати) процессов и контролю за реализацией в ходе УК. Этот стандарт целесообразно применять и при создании программных продуктов для объектов других критически важных для государства секторов экономики.

Процесс УК в значительной степени обеспечен, если структура ***ПК четко стандартизирована,*** а его компоненты имеют ***унифицированные интерфейсы с позиций управления ими и обмена данными.*** Для этого правила модульно-иерархического построения ПК должны детализироваться до уровня конкретных методик создания и оформления модулей, компонентов и межмодульного взаимодействия. При этом унифицированные межмодульные интерфейсы целесообразно, по возможности, упрощать и ослаблять, а также подготавливать условия для их проверок в реальном режиме функционирования программ. Проектирование ПК сверху вниз и последующее сохранение четкой структуры интерфейсов значительно облегчают УК и продлевают срок жизни версий комплексов программ. Регистрация и учет истории этого процесса обеспечивает возможность его контроля и пошагового восстановления выполненных изменений (отката) при ***выявлении вторичных дефектов***, внесенных в процессе разработки модификаций для очередной базовой версии программного продукта.

Модификация, учет и тиражирование версий могут требовать больших ресурсозатрат. По этой причине при выпуске каждой новой базовой версии разработчики стремятся обеспечить преемственность ее функций и компонентов с версиями предыдущими. Также рассматривается и подготавливается решение для возможного прекращения модификаций некоторой устаревшей версии ПК или ее конкретных компонентов. В результате развития сложного комплекса программ среди всего множества версий для каждого ПК (или компонента) в архиве тиражирования и обеспечения сохранности образуется ***зона сопровождения*** — комплект конфигураций, доступных для изменений базовых версий программного продукта.

В процессе проектирования ПК должна быть формализована и документально зафиксирована ***концепция организации управления конфигурацией,*** содержащая [2, 3]:

- ожидаемую длительность процесса поддержки развития и модификации конкретного проекта ПК;
- масштаб и уровень предполагаемых изменений и модификаций;
- возможное число и периодичность выпуска базовых версий ПК;
- организационные основы процессов сопровождения и конфигурационного управления ПК;
- требования к документированию изменений и базовых версий ПК;
- перечень лиц, которые будут осуществлять управление конфигурацией, например, покупатель, разработчик или специальный персонал поддержки жизненного цикла ПК.

Управление конфигурацией включает действия и средства, позволяющие устанавливать ***категории, статус и личности руководителей***, которые правомочны определять целесообразность и эффективность изменений, а также техническую реализуемость корректируемых версий с учетом ограничений бюджетов и сроков. При анализе и селекции изменений важен точный учет степени влияния каждого изменения на все остальные компоненты и на основные характеристики качества программного продукта. По этой причине решения о кардинальных изменениях ПК и его компонентов должны приниматься на достаточно высоком уровне руководства проектом, на котором можно оценить их влияние на концептуальную целостность и качество всей системы.

Концепция управления конфигурацией в рамках конкретного проекта должна предусматривать возможность ***анализа изменений иерархической структуры (конфигурации)*** программных комплексов и их компонентов как сверху вниз, так и снизу вверх. Первая задача состоит в обеспечении пошаговой детализации сверху вниз возможных причин конкретных дефектов (проявлений ***вторичных ошибок***) или неэффективности функционирования программы в целях обнаружения их первичного источника (***первичной ошибки***). Вторая задача при движении снизу вверх должна обеспечивать проверку корректности взаимодействия связанных корректировок и сохранения концептуальной целостности и качества комплекса программ и/или его компонентов.

Важной целью УК является документальное оформление и обеспечение полной наглядности текущей конфигурации программ и данных, а также степени выполнения требований к их функциональным характеристикам. Еще одна задача заключается в том, чтобы все лица, работающие над проектом, на любом этапе его жизненного цикла использовали достоверную и точную информацию обо всех ЕК и их взаимодействии. Процессы УК включают работы по идентификации конфигурации; по контролю изменений; по определению базовой версии разработки и архивированию ПК, включая соответствующие документы жизненного цикла; по аудиту конфигурации; по компоновке и поставке программного продукта в течение всего жизненного цикла системы. Процессы УК, выполняемые совместно с другими процессами жизненного цикла программного средства (ПС), направлены на достижения ***основных целей*** (стандарты **ISO 12207:2008, ISO 10007**), которые позволяют обеспечить:

- возможность оценки соответствия требованиям заказчика результатов функционирования ПК на всех этапах его жизненного цикла;
- определяемую и управляемую конфигурацию ПК на протяжении его жизненного цикла;
- управление входными и выходными данными процесса в течение всего жизненного цикла ПК, что гарантирует непротиворечивость и повторяемость выполнения отдельных работ в ходе реализации этого процесса;

- контроль над тем, чтобы фиксировались дефекты и ошибки, а изменения регистрировались, утверждались и реализовались;
- гарантированно надежное физическое архивирование, восстановление и сопровождение ЕК и документов проекта по созданию и/или сопровождению ПК.

Организация управления конфигурацией комплекса программ

Изменения конфигурации ПК и его компонентов в соответствии с планом проекта должны предусматривать действия, четко определяющие [2—4]:

- ***почему*** и с какой целью проводится корректировка программ или данных;
- ***кто*** выполняет и кто санкционирует проведение изменений комплекса программ или компонентов;
- ***какие*** действия и процедуры должны быть выполнены для реализации изменений единиц конфигурации;
- ***когда*** по срокам и в координации с какими другими процедурами следует реализовать определенную модификацию компонентов и конфигурацию ПК;
- ***как*** и с использованием каких средств и ресурсов должны быть выполнены запланированные изменения ПК и компонентов.

Для того чтобы УК было эффективным, следует определить ***организационную структуру коллектива специалистов*** (рис. 1), которые его реализуют. Эта структура обычно ориентируется на конкретный проект и, при необходимости, адаптируется к потребностям различных этапов жизненного цикла программной продукции. Организационная структура коллектива специалистов, обслуживающих УК, должна способствовать координации действий, а также ***распределению соответствующих полномочий и ответственности*** за все действия по ее реализации. В рамках организации проекта ПК следует идентифицировать инстанцию, уполномоченную утверждать конфигурационные базы и любые изменения к ним. В качестве такого органа выступает ***совет по управлению конфигурацией***. В случае небольших проектов ответственность за УК может быть возложена на отдельных лиц, участвующих в проекте.

Руководитель проекта ***может учредить совет по конфигурации***, который будет иметь полномочия анализировать и утверждать или не утверждать программу и процедуры УК, выбор объектов конфигурации, конфигурационные базы и изменения к этим базам. Членов совета по конфигурации обычно назначает руководитель проекта ПК. В составе совета должны быть представители, имеющие знания и опыт по всем необходимым для выполнения действий по УК дисциплинам. Его должен возглавлять руководитель проекта или его представитель. Совет по конфигурации может быть организован на разных уровнях полномочий. Например, если согласно контрактным требованиям необходимо активное участие заказчика в процессе жизненного цикла ПК, то заказчик тоже может учредить совет по контролю конфигурации.

Рис. 1. Организационная структура коллектива специалистов

Конфигурационная идентификация включает методы и средства, с помощью которых можно однозначно устанавливать и различать версии ПК, входящие в них компоненты (ЕК), их варианты и модификации, а также "родословное дерево" объектов конфигурации, технических условий и идентифицированный номер комплекта документации. Кроме того, каждый вариант

модуля, компонента или ПК и их изменений должен соответствовать правилам нумерации (идентификации).

Цель работ по идентификации конфигурации заключается в однозначной маркировке всех версий каждой ЕК в процессе ее жизненного цикла. Такая маркировка позволяет:

— создать информационную основу для ссылок на ЕК и управления ими;

— идентифицировать каждую ЕК, для каждого отдельно управляемого компонента конфигурации и для комбинаций единиц конфигурации, которые составляют ПК;

— провести идентификацию единиц конфигурации до начала реализации процедуры контроля изменений документов.

Разработчик должен участвовать в выборе ЕК, который выполняется согласно проекту, определяющему архитектуру системы. Он должен однозначно идентифицировать объекты, которые будут задействованы в процессах управления конфигурацией, назначить уникальный для проекта идентификатор каждой ЕК и каждому дополнительному объекту, используемому в таких процессах. Использование более детализированного уровня абстрагирования ЕК позволяет снизить сложности в манипулировании (в действиях) с ними. Большую систему проще собирать из компонентов ЕК, чем из разрозненных версий файлов модулей программ и данных.

В каждом конкретном проекте следует разработать ***правила нумерации ЕК*** и применять их для идентификации объектов конфигурации, документов по конфигурации и изменений. При этом необходимо учитывать существующие процедуры нумерации, принятые у подрядчика, или общепринятые процедуры. Однако идентификационные номера должны быть уникальными. Правила нумерации или другие системы учета конфигурации должны позволять управлять:

— иерархическими отношениями или отношениями подчинения между объектами конфигурации в рамках структуры ПК;

— иерархическими отношениями или отношениями подчинения между компонентами в каждом объекте конфигурации;

— отношениями между объектами и документами;

— отношениями между документами и изменениями.

В ключевых точках разработки проекта все объекты должны быть подвергнуты ***версионному учету***. В этих точках следует фиксировать версии всех объектов и компонентов, составляющих программный продукт или систему. В соответствии с методологией УК в итеративном процессе разработки ПС ***учет версий*** необходим в конце каждой следующей итерации для обеспечения:

— воспроизводимости — возможности вернуться назад во времени и повторить определенный выпуск ранее существовавшего компонента, программной системы или среды разработки;

— контролируемости — объединения требований к системе, проектных планов, результатов тестирования и объектов разработки, для учета версий не только системных компонентов, но и объектов проектирования и планирования;

— отчетов, что позволяет получать сведения о любой версии системы, сравнивать различные версии, находить ошибки и составлять документацию.

Управление запросами на изменения включает регистрацию, отслеживание и анализ запросов от субъектов, взаимодействующих с системой, на модификацию ПК, компонентов и данных. Оно включает процессы принятия решений для планирования необходимых изменений и процессы, которые необходимы для их осуществления. Управление запросами на изменения представляет собой ***центральную часть концепции управления конфигурацией***. Незарегистрированные запросы на изменения могут быть потеряны или останутся без реакции.

Запрос на доработку ПК определяет его новое свойство или указывает на необходимость обеспечить реализацию тех или иных функций. ***Дефект*** — это аномалия или брак в работающем программном продукте. Дефектом может быть любое его отклонение от штатного (планируемого) режима функционирования. Такое отклонение следует взять под контроль и выяснить причину аномалии. Несмотря на то что с запросами на доработку и с устранением дефектов связана достаточно похожая информация, в процессах УК они часто обрабатываются совершенно по-разному. Реализация изменений в ПК требует принятия ряда решений. Запрос, связанный с необходимостью модификации ПК, способен, в свою очередь, породить несколько внутренних, технического характера запросов на доработку.

Следующим шагом после установления типов контролируемых запросов является уточнение объема информации, которая подлежит фиксации в ходе жизненного цикла запроса. Наиболее важным и зачастую сложным шагом является определение процесса, который используется для контроля каждого запроса на изменение. Типы запросов на изменения, модели переходов и состояний варьируются достаточно широко и обычно включают [2]:

- представление и регистрацию запроса на изменение;
- оценку запроса на предмет определения его категории и приоритета;
- решение о порядке выполнения запроса;
- реализацию корректировок, в ходе которых компоненты системы и программная документация создаются или модифицируются в целях реализации запроса;
- проверку на соответствие требованиям или на отсутствие исправленного дефекта.

Представление запроса на изменение предполагает его регистрацию. Дефект и запрос на доработку обыч-

но отличаются своим происхождением и набором фиксируемой информации. Запросы на доработку могут поступать из различных источников. В большинстве случаев они приходят от заказчиков и возникают в процессе работы напрямую или косвенно, через опрос сотрудников.

Оценка запроса заключается в просмотре вновь поступивших запросов и в выводах об их характеристиках — относится запрос к анализу устранения дефекта или он является запросом на доработку; какова значимость этого запроса; каков приоритет в реализации запроса. Необходимо также учесть возможные последствия доработки — изменение доли рынка, увеличение прибыли, влияние на продажи и поддержку пользователей. В общем случае запросы на доработку и модификацию должны оцениваться руководителем проекта.

Изменения кода и/или данных отдельных модулей и небольших компонентов ПК (меньших, чем ЕК) подвергается наименее формализованному и защищенному от случайностей УК комплекса. Их изменения в процессе тестирования обычно не требуют санкции руководителей проекта. Версии функциональных групп программ и комплексов программ в целом могут корректироваться только с разрешения руководителей соответствующего ранга. Тем самым, должны предотвращаться несогласованные и несанкционированные изменения, способные снизить качество и целостность версий ПК. Изменения этих версий допускаются пользователями или поставщиками в пределах, ограниченных эксплуатационными документами по адаптации к характеристикам и особенностям внешней среды и условиями применения конкретной версии ПК.

Решение о корректировке конфигурации ПК состоит в выборе одного из следующих действий: выполнить запрос на изменения; отложить реализацию запроса; отклонить запрос. Дефекты и запросы на доработку почти всегда обрабатываются по-разному. В случае запроса на доработку решение обычно выносит руководитель проекта или аналитик. Запросы на доработку оценивают вместе или по отдельности. Затем относительно каждого из них принимают решение о реализации в данной версии продукта, отсрочке или отклонении. В крупных проектах иногда для контроля за изменениями создается один или несколько специальных советов. Задача таких контрольных советов заключается в нахождении компромисса между качеством продукта и сроками выпуска версии на завершающем этапе разработки программного продукта.

Контроль корректности конфигураций версий компонентов, ПК и данных предназначен для систематической оценки предполагаемых изменений ЕК и координированной их реализации, учитывающей соответствия спецификациям и требованиям заказчика, эффективности каждого из них и затрат на выполнение изменения. Методы такого контроля должны также обеспечивать управление состоянием и процессами развития компонентов ЕК, их вариантов, связей и модификаций, а также адекватность реально изменяющихся объектов их комплектной документации. Целью контроля является анализ дефектов на предмет необходимых модификаций кода для их устранения, утверждение и реализация этих изменений, поиск обратной связи с процессами, на которые такие изменения воздействуют. Последние определяют в ходе планирования проекта создания ПК. Перед сборкой базовой версии ПК рекомендуется ***блокировка интеграционного потока и запрет отправки изменений*** ЕК в соответствующую базу данных. В результате интегратор может проводить сборку и создавать базовую редакцию ПК для стабильного (без отмеченных изменений) множества элементов исходного кода ЕК.

Сборка версии ПС представляет собой первый уровень тестирования, проводимый интегратором с тем, чтобы убедиться, что все модифицированные файлы ЕК могут быть собраны в единый компонент или комплекс программ. Интегратор первым локализует возникшие при сборке неполадки, предотвращая, таким образом, перенос вопросов несовместимости в сферу компетенции специалистов. Если сборка прошла успешно, то следует перейти к квалификационному тестированию или повысить статусы редакций компонентов ЕК.

Сформированные базовые версии ЕК должны регистрироваться в контролируемых библиотеках ПК, позволять ссылаться, управлять и прослеживать их изменения. Они должны быть защищены от внесения любых несанкционированных изменений. Конфигурационная база при этом должна состоять из всех утвержденных документов, которые определяют программную продукцию или отдельные ее компоненты в данный момент.

Организация тестирования при конфигурационном управлении комплексами программ

Процессы и структура базы данных управления конфигурацией должны быть ***ориентированы на тестирование*** компонентов и сложных комплексов программ [4]. Для конфигурационного управления тестами, модификациями требований к комплексу программ и результатами испытаний ***в базе данных конфигурации должны собираться сведения***, которые состоят из следующих ***основных частей***:

- ***база данных предлагаемых изменений требований,*** в которой содержаться данные о ситуациях отказа, о дефектах и ошибках, условиях их проявления и характеристиках, обнаруживающих такие ситуации тестов, а также предложения для изменения функций программ, подлежащих анализу, и выделению тех из них, для которых будут разрабатываться корректировки модификаций комплекса программ;

- ***база данных подготовленных и утвержденных корректировок требований и тестов***, включающая разработанные тесты и модификации программ, отобранные группой УК для проведения изменений в очередной версии ПК;
- ***база данных утвержденных требований, тестов и реализаций версий ПК и компонентов***, содержащая откорректированные версии и набор изменений требований и тестов, выполненных в каждой из перечисленных элементов базы.

Перечисленные выше сведения в структурированных подсистемах базы данных УК должны быть защищены от случайных и преднамеренных искажений. Такая защита реализуется путем специально организованного санкционированного доступа к информации, дублирования и контроля модификаций, хранения сведений об истории создания и изменениях в процессах жизненного цикла комплекса программ. Необходимо гарантировать сохранность всех версий изменений с учетом их важности для результативности выполнения всего проекта.

Особенно защищенным от искажений и разрушения следует сохранять ***архив версий программных продуктов***, прошедших успешные испытания, утвержденные заказчиком. Для устранения дефектов, реализации корректировок и ошибок при создании новых версий ПК или отдельных его компонентов целесообразно выделять рабочую копию предшествовавшей версии и архив накопленных изменений. Такой архив обеспечивает возможность ***"отката"*** к предыдущей версии в случае разрушительных некорректных изменений в процессе разработки новой версии программного продукта. Такая ***система обеспечения информацией процессов УК в ходе тестирования ПК*** может быть структурирована на подсистемы в соответствии с адаптированной версией жизненного цикла конкретного комплекса программ. В соответствии с задачами специалистов, участвующих в проекте, на рис. 2 представлены ***основные подсистемы базы данных*** информационного обеспечения модификаций и тестирования, ориентированные на определенные процессы и компоненты комплексов программ. Для каждой подсистемы управления конфигурацией (левая колонка) целесообразно выделять достаточно автономную базу данных компонентов (правая колонка) с ограниченным доступом только для определенных категорий специалистов.

Эти подсистемы базы данных могут быть построены на основе стандартизированной системы управления базой данных (СУБД) проекта. Они призваны взаимодействовать с аналогичными по структуре предшествующей и последующей базами данных жизненного цикла комплекса программ. Функции таких подсистем — накапливать и содержать основные компоненты и документы проекта на соответствующем уровне производства и сопровождения комплекса программ. Интерфейсы этого взаимодействия подсистем базы данных должны быть стандартизированы и ограничены по объему и возможности доступа к текущей и отчетной информации для отдельных категорий специалистов в соответствии с регламентом принятой политики безопасности. Для каждого сложного проекта создания комплекса программ целесообразно оформлять и утверждать ***руководство и схему подсистем базы данных, обеспечивающие управление конфигурацией комплекса программ, его требованиями и тестами***. Следует также определять категории лиц, ответственных за поэтапную реализацию, контроль и сохранность информации обо всем проекте.

Контроль корректности состояния и изменений требований и тестов должен обеспечивать регистрацию, оценку, рассмотрение и утверждение их состояния и изменений на всех этапах жизненного цикла комплекса программ. Такой контроль призван:

- обеспечивать целостность компонентов конфигурации и базовых версий комплекса, а также защиту их от некорректных изменений требований и тестов;
- гарантировать, что каждое изменение компонента конфигурации учтено в изменении идентификации конфигурации требований и тестов;
- автоматизировать регистрацию, утверждение и контроль изменений в базовых версиях и компонентах конфигурации требований и тестов;
- контролировать изменения требований к программному продукту, которые должны прослеживаться вплоть до места их источника, а выполнение процессов в жизненном цикле ПК должно быть отслеживаемо с того момента, когда изменения начинают сказываться на выходных данных;
- автоматизировать проведение работ по внесению изменений требований, по модификации и обновлению документов, сопровождающих жизненный цикл комплекса программ, на которые эти изменения могут влиять.

Документирование состояния управления требованиями и тестами УК конкретного сложного ПК и его компонентов должно отражать:

- общую структуру комплекта документов на конфигурацию требований и тестов к программному продукту;
- номенклатуру и содержание каждого документа, отражающего требования и/или их изменения;
- требования к качеству, оформлению и обозначению каждого документа;
- регламент комплектования, корректировки и хранения документов;
- правила обращения, процессов корректировки и сопровождения документов, отражающих требования и тесты;
- графики подготовки, проверки, редактирования, согласования, утверждения и распространения документов.

Работы по созданию и применению документов конфигурационного управления должны гарантиро-

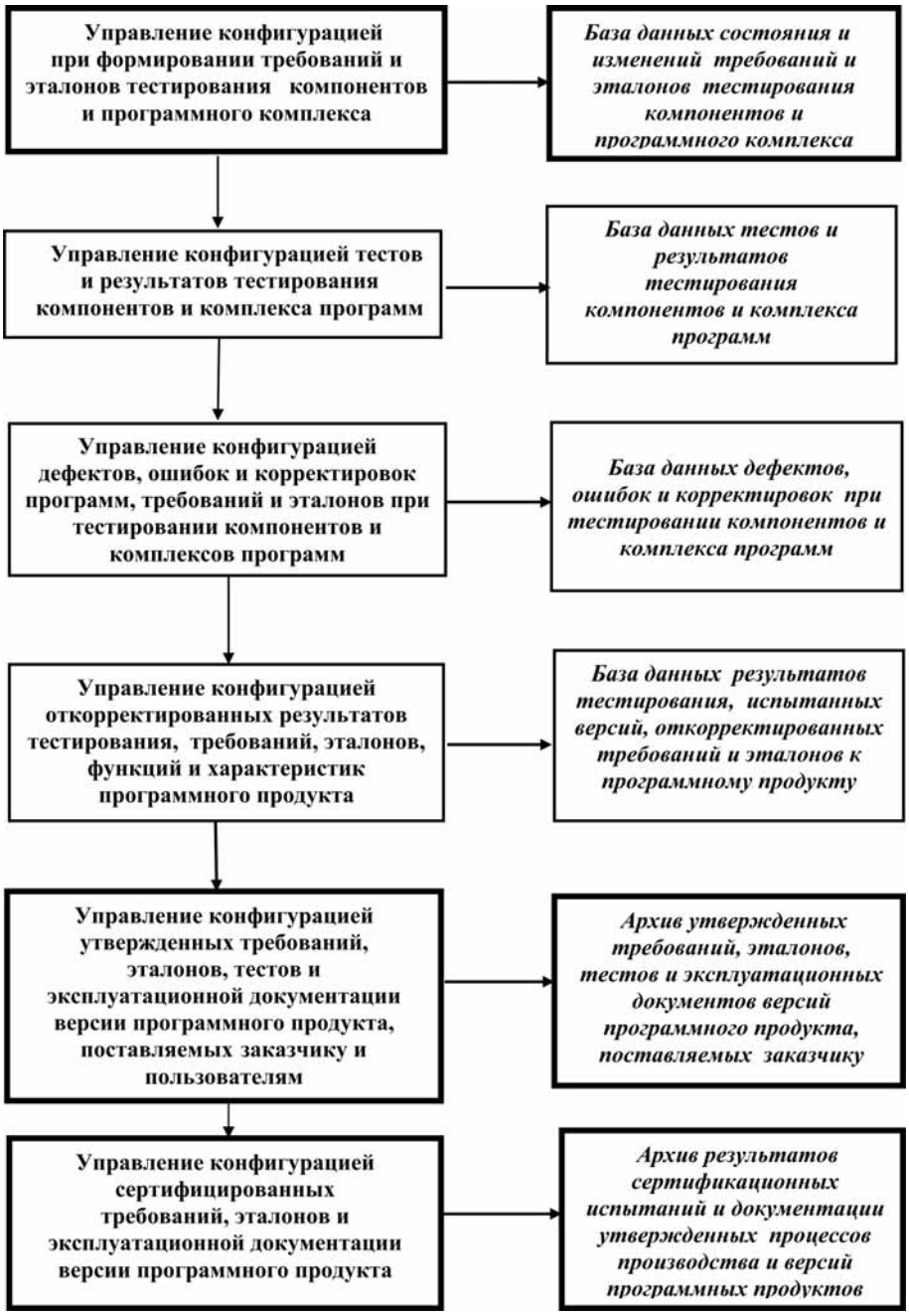


Рис. 2. Основные подсистемы базы данных информационного обеспечения модификаций и тестирования комплексов программ

вать пользователям возможность применения ***требований и тестов, только адекватных санкционированным версиям ПК и компонентов***. Официальные документы при этом должны быть получены только из архива соответствующей базы данных (каталога). Каждая ЕК должна быть идентифицирована, документирована и выпущена ее официальная версия до того, как осуществляется производство программных продуктов для пользователей. Цель работ по архивированию и получению документов — обеспечить получение документов, которые необходимы для копирования, повторной генерации, повторного тестирования и модификации ПК (рис. 2).

Методика оформления отчетов о выявленных дефектах, ошибках и предложениях по корректировке требований должна содержать рекомендации заказчикам, специалистам, участвующим в испытаниях, пользователям, которым она необходима для выявления, регистрации и формализации условий проявления и содержания отказов и дефектов версий программного продукта, которые испытывают и/или эксплуатируют. Эта методика должна быть включена в состав эксплуатационной документации и передана ***каждому пользователю версии***. В методике следует стимулировать специалистов-пользователей анализировать, подготавливать и представлять рекомендации заказчику и разработчикам по совершенствованию и развитию требований и тестов к основным функциям и характеристикам качества поставляемой версии программного продукта.

Результаты анализа и предложения необходимо передавать для управления конфигурацией в унифицированной форме ***отчетов специалистов о выявленных дефектах и предложениях по корректировке требований*** к программному продукту. Эти отчеты должны содержать:

- подробное описание сценария функционирования и тестирования программного продукта, а также исходных данных, при которых выявлен отказ или дефект;
- описание проявления отказовой ситуации или дефекта и документы результатов его регистрации;
- предположение о причине, вызвавшей проявление отказа или дефекта;
- предложение по корректировке требований к ПК и его компонентам для устранения дефекта и/или совершенствования функциональной пригодности применения программного продукта.

Изменения, отобранные советом проекта по управлению конфигурацией ПК, переводятся из базы данных предварительных изменений требований в следующую подсистему базы данных. Кроме того, для каждой подготовленной корректировки следует регистрировать результаты ее рассмотрения советом по управлению конфигурацией комплекса программ и утверждения на введение в новую версию программного продукта или для частных извещений пользователям. Отвергнутые корректировки возвращают в базу данных предлагаемых изменений требований.

Отчеты о дефектах и следующих за ними корректирующих действиях, в том числе изменениях требований и тестов, которые связаны с процессами жизненного цикла комплексов программ, целесообразно фиксировать в отдельной ***подсистеме*** управления конфигурацией. В ней должны содержаться отчеты о дефектах и отказовых ситуациях, которые предусматривают возможность идентификации затрагиваемых при этом компонентов и требований; состояние сообщений о дефектах; утверждение и закрытие сообщений о дефектах после соответствующих корректировок.

Система производства комплекса должна быть закрытым циклом, гарантирующим, что все обнаруженные дефекты и отказовые ситуации ***вводят в систему регистрации***, необходимые модификации требований инициируются, состояния корректирующих действий отслеживают, а сообщения об изменениях требований сопровождают в течение срока действия контракта. Каждый дефект или модификация требований должны быть классифицированы по категориям и приоритетам. Тесты и корректирующие действия должны быть оценены с тем, чтобы определить, были ли дефекты устранены, а изменения компонентов и комплекса программ были правильно выполнены без внесения дополнительных дефектов.

Конфигурационная база реализованных изменений требований и тестов должна быть установлена соглашением разработчиков с заказчиком и использоваться для официального контроля за конфигурацией требований и их реализацией. После первоначального выпуска документов на конфигурацию версии программного продукта все изменения следует контролировать. Влияние изменения на продукт, требования заказчика и подвергнутая такому влиянию конфигурационная база должны определять степень строгости выполнения всех положений при работе с изменениями. Они создают основу для системы классификации, которая используется для распределения изменений требований по категориям и приоритетам. Разрушение сведений о выполненных изменениях программного продукта может приводить к большим затратам на их восстановление. По этой причине база, содержащая данные об изменении требований, о тестах и их последствиях, должна дублироваться и поддерживаться методами и средствами сопровождения, аналогичными тем, которые применяют для основной документации, тестов и текстов комплекса программ.

Отчеты о состоянии конфигурации требований, компонентов и комплексе программ [5] должны содержать информацию обо всех идентификаторах конфигурации, обо всех отклонениях от установленных требований к конфигурации подсистем баз данных. Проверки конфигурации требований следует проводить до принятия конфигурационной базы программного продукта заказчиком. Такой подход гарантирует, что продукция соответствует установленным контрактом требованиям и они точно отражены в документах на конфигурацию продукта. Такие проверки могут требоваться для официальной приемки конфигурации версии программного продукта.

Особое значение при сопровождении требований и тестов имеет ***документация на реализованные корректировки и тесты,*** с помощью которых проверялась корректность версий компонентов и комплекса в целом. Эта документация должна позволять ***восстанавливать историю разработки*** и обеспечивать возможность проверки каждого изменения требований для любого компонента. Разработка и тестирование изменений могут опережать их документальное оформление. В течение этого времени возможны отдельные уточнения изменений требований в версии. В результате документация должна непрерывно ***"догонять"*** реальное состояние программного комплекса в базе данных конфигурации. Для упорядочения этого процесса стандартами установлена возможность оперативного выпуска ***предварительных извещений на частные изменения требований и их реализацию***. Эти извещения регистрируют как временные и "погашают" (устраняют) при полном оформлении документации на версию программного продукта и все реализованные изменения.

Документирование при управлении конфигурацией комплекса программ

Первичные сообщения от испытателей или пользователей об отказах, дефектах и ошибках следует регистрировать в специализированных таблицах, отражающих их формализованные параметры. Сообщения при этом должны на содержательном уровне представляться вместе ***с тестовыми данными и описаниями ситуаций***, при которых зарегистрирован дефект. На содержательном уровне или в виде спецификации требований необходимо также регистрировать в базе данных предложения по совершенствованию качества комплекса программ. По результатам анализа и тестирования от группы управления конфигурацией эти данные получают признаки подлежащих дальнейшей проработке или отвергнутых изменений требований.

Для эффективной реализации процессы управления требованиями и тестами сложных комплексов программ должны быть поддержаны ***службой тиражирования, архивирования и гарантированного хране-***

ния всей необходимой информации о документах, об их корректировках, о версиях, об авторах и их правах на изменения. Эта ***служба архива*** должна обеспечивать поставку пользователям только утвержденных копий версий программного продукта и/или их компонентов с полным, адекватным комплектом эксплуатационных документов. Для гарантированного сохранения состояния, результатов модификаций требований к программам и обеспечения возможности их анализа на любой стадии проекта, а также в целях возможности реализации ***отката по истории*** выполненных корректировок требований и компонентов, следует организовать четкую ***систему хранения и копирования подлинников, дубликатов и копий требований и тестов*** программного продукта.

Для реализации на практике процедур и планов УК требований к программным продуктам и тестам необходимы организационные мероприятия, гарантирующие участникам проектов определенную ***культуру, дисциплину разработки и выполнения модификаций***. Такая организационная система должна обеспечивать специалистам разных квалификаций и ролей в проекте возможность взаимодействия при решении поставленных перед ними комплексных задач для накопления, хранения и обмена упорядоченной информацией о состоянии и изменениях требований, тестов и компонентов комплекса программ.

Сложной задачей установки реальных ***ориентиров затрат*** для заказчика является оценка размера программного продукта и его изменений. Нереальные ожидания, основанные на неточных оценках требуемых ресурсов, представляют собой одну из ***частых причин провала проектов.*** Это возникает, как правило, вследствие превышения ограничений на выделяемые ресурсы. Так как достоверность определения и размер возможных модификаций комплекса программ составляют один из ключевых факторов для последующего анализа их влияния на необходимые ресурсы, то целесообразно применять несколько доступных на настоящее время методов для оценивания этого размера. На базе всего комплекса использованных тестов для каждой версии программного продукта должна быть создана и задокументирована ***эталонная тестовая (контрольная) задача и контрольные результаты ее решения***. Эти документы следует оформлять в соответствии со стандартами, тиражировать и передавать пользователям вместе с версией программного продукта, а также с остальными эксплуатационными документами.

Архивирование и тиражирование базовых версий программных продуктов и документов

Постановка и выпуск — важные процессы при УК базовыми версиями программного продукта. ***Постановка*** — процесс помещения объектов ЕК (исполняемых модулей, откомпилированных файлов, документов) в базу данных под жесткий версионный контроль. Основное назначение постановки состоит в помещении копий исполняемых модулей и других производных файлов ЕК в хранилище базовых версий для надежного и безопасного доступа к ним. ***Выпуск*** — придание собранным ЕК окончательной формы и предоставление доступа к ним конечным пользователям. Цель выпуска заключается в подготовке базовой версии программного продукта и документов для заказчика.

Поставка должны гарантировать использование исключительно санкционированных (в рамках проектных соглашений) версий ПК и компонентов с тем, чтобы официальные версии могли быть получены только из архива. Каждая ЕК должна быть идентифицирована и документирована. Ее официальная версия должна быть выпущена до того, как осуществляется производство ПК для пользователей. Цель работ по архивированию и применению документов — обеспечить получение документов жизненного цикла ПК для копирования, повторной генерации, повторного тестирования и модификации программного продукта.

Для принятых к внедрению изменений должен разрабатываться ***план доработок программ***, а также должен быть определен конкретный специалист, ответственный за каждую корректировку программы. Изменения, выделенные и утвержденные уполномоченным лицом для реализации, как правило, отбирают по приоритетам на следующие группы:

- срочные изменения, которые должны не только быть внесены в очередную базовую версию ПК, но и сообщены пользователям для оперативной корректировки программ до внедрения очередной официальной версии;
- изменения, которые требуют дополнительного анализа целесообразности и эффективности их реализации в последующих базовых версиях и могут пока не внедряться в очередную версию ПК;
- изменения, которые не оправдывают затрат на разработку и выполнение корректировок или практически не влияют на качество и эффективность ПК, а поэтому пока не подлежат реализации.

Эти группы изменений регистрируют в ***описании и базе данных подготовленных корректировок*** для последующего использования с указанием их подготовивших специалистов и лиц, принявших решение о реализации. Подготовленные и утвержденные предложения на изменения ПК поступают к специалистам, осуществляющим планирование их внесения в реальные программы, а также разработку и корректировку конкретных компонентов и документов. После выполнения доработок разработчиками компонентов и корректировки документации следует окончательная проверка корректности изменений, которая осуществляется ***специалистами по квалификационному***

тестированию и испытаниям очередной версии ПК (см. рис. 2).

Тестирование необходимо сосредоточивать на компонентах, впервые вводимых или значительно модифицируемых в данной версии. Если изменения в программе или в данных невелики, то тестирование обычно можно ограничить компонентами, непосредственно связанными с выполненной корректировкой. При этом следует учитывать, что корректировки программ в 10...30 % случаев сами содержат ошибки и требуют тщательного тестирования не только тех частей, где внесены изменения.

Объединение и сборка функциональных компонентов — групп откорректированных программ позволяет создать ***эталон базовой версии ПК***, подлежащий квалификационному тестированию по полной программе испытаний. При большом числе выполненных изменений сложность испытаний может приближаться к испытаниям первой базовой версии. Объем тестирования при испытаниях базовой версии должен быть согласован между разработчиком, заказчиком и/или приниматься с учетом мнения пользователей. Результаты испытаний регистрируют в типовых протоколах и в проекте акта о завершении испытаний версии. Все проверенные и подтвержденные при испытаниях корректировки программ регистрирует и утверждает уполномоченный руководитель совета по управлению конфигурацией ПК в акте, определяющем завершение подготовки новой базовой версии.

Конфигурационная база программного продукта должна быть зарегистрирована официально в определенный момент времени и использоваться в качестве отправной точки для контроля за состоянием конфигурации и утвержденными изменениями версии. Пользователям может поставляться базовая копия версии для самостоятельной ***адаптации*** к характеристикам внешней среды и особенностям конкретного применения. В этом случае адаптация должна проводиться строго по инструкциям разработчика этой версии и должны быть запрещены любые дополнительные изменения за пределами, предписанными документацией. Подобные изменения автоматически снимают гарантии разработчика, а ответственность за корректность адаптации возлагается при этом на пользователя.

Для независимого удостоверения (определения) качества проведенных модификаций в испытанной и утвержденной разработчиком версии ПК по заявке разработчика, заказчика или пользователей она может подвергаться обязательной или добровольной ***сертификации*** [6]. Комплект поставки, состоящий из копии физических носителей и эксплуатационной документации, а также применявшиеся разработчиками квалификационные тесты и методики испытаний передают в сертификационную лабораторию. При сертификационных испытаниях допускается расширение набора и параметров тестов, однако только в пределах, ограниченных технической документацией на конкретную версию ПС. Успешно проведенные испытания и полученный сертификат качества подтверждают соответствие комплекса программ документации, надежность и безопасность применения версии до тех пор, пока в нее не будут внесены какие-либо изменения. Сертификат может быть аннулирован при любых, даже внешне незначительных корректировках версии программного продукта. Целесообразно выделять ***специальный архив с резко ограниченным доступом***. В таком архиве следует накапливать подлинники базовых версий и дополнительные данные, которые гарантируют их сохранность и возможности восстановления.

Любое изменение программного продукта может ***потребовать пересмотра соглашений*** с заказчиком о его функциональных возможностях, о качестве, сроках и бюджете. Группы управления изменениями часто становятся частью такого процесса и организуют форум для открытых дискуссий и поиска компромиссов с заказчиком по поводу ранее принятых решений. Необходимо быть внимательным к организационным, функциональным, технологическим и проектным изменениям, принимать в расчет любые дополнительные знания о проекте, которые оказывают ***воздействие на риски программных продуктов***.

Заключение

Перечисленные выше методы и средства управления конфигурацией больших и сложноорганизованных программных комплексов, которые разрабатывают и сопровождают коллективы разноплановых специалистов исполнителя с участием представителей заказчика этих работ, должны планироваться и реализоваться. В противном случае неизбежны дефекты программного продукта на разных этапах выполнения проекта и на разных архитектурных уровнях продукта, которые, как правило, приводят не только к ухудшению его качества, но и к потере управляемости проекта в целом.

Список литературы

1. **Критически** важные объекты и кибертерроризм. Часть 1. Системный подход к организации противодействия. / Под ред. В. А. Васенина. М.: МЦНМ, 2008. 398 с.

2. **Уайт Б. А.** Управление конфигурацией программных средств. Практическое руководство по Rational ClearCase. Пер. с англ. М.: ДМК Пресс, 2002.

3. **Фатрелл Р. Т., Шафер Д. Ф., Шафер Л. И.** Управление программными проектами: достижение оптимального качества при минимальных затратах. Пер. с англ. М.: Вильямс, 2003.

4. **Липаев В. В.** Тестирование компонентов и комплексов программ. Учебник. М.: СИНТЕГ, 2010.

5. **Вигерс К. И.** Разработка требований к программному обеспечению. Пер. с англ. М.: Русская редакция, 2004.

6. **Липаев В. В.** Сертификация программных средств. Учебник. М.: СИНТЕГ, 2009.

7. **Блэк Р.** Ключевые процессы тестирования. Пер. с англ. М.: ЛОРИ, 2006.

УДК 004.31

П. Н. Бибило, д-р техн. наук, проф., зав. лаб., **С. Н. Кардаш**, канд. техн. наук, ст. науч. сотр., **Н. А. Кириенко**, канд. техн. наук, доц., ст. науч. сотр., e-mail: kir@newman.bas-net.by, **Ю. В. Поттосин**, канд. физ.-мат. наук, доц., вед. науч. сотр., **В. И. Романов,** канд. техн. наук, доц., вед. науч. сотр., Объединенный институт проблем информатики Национальной академии наук Беларуси, г. Минск

Автоматизация синтеза конвейерных КМОП-схем

Описываются структура и функциональные возможности программной системы ArCon конвейеризации логических схем. Система предназначена для проектирования схем с повышенным быстродействием. Представлены результаты экспериментов по исследованию быстродействия и энергопотребления многоуровневых логических схем из библиотечных элементов заказных сверхбольших интегральных схем (СБИС), выполненных по КМОП-технологии.

Ключевые слова: *автоматизация проектирования, заказные КМОП СБИС, конвейеризация, синтез схем с повышенным быстродействием*

P. N. Bibilo, S. N. Kardash, N. A. Kirienko, Yu. V. Pottosin, V. I. Romanov

Automation of Synthesis of Pipelined CMOS Circuits

The structure and the functionality of the software system ArCon for synthesis of pipelined CMOS circuits of high speed are described. The results of investigating the speed and power consumption of multilevel pipelined circuits of custom VLSI library elements made in CMOS technology are presented.

Keywords: *design automation, custom CMOS VLSI, pipelining, synthesis of high speed circuits*

Введение

Быстродействие нерегулярных логических схем из библиотечных элементов можно повысить, используя конвейерную структуру. В основу принципа конвейеризации положена возможность приема очередных сигналов на входы устройства, перерабатывающего входные дискретные сигналы в выходные, до того как закончилась обработка набора предыдущих сигналов [1, 2]. Принцип конвейеризации эффективно используется, когда характер обработки информации представляется как последовательность операций, каждая из которых состоит из последовательности этапов [3, 4]. Для того чтобы начать выполнение последующей операции, не надо ждать окончания всего процесса выполнения предыдущей операции. Достаточно, чтобы у предыдущей операции был закончен только первый этап.

Для того чтобы представить комбинационную схему в виде конвейерной структуры, ее необходимо разбить на блоки и вставить между блоками регистры (наборы триггеров), имеющие общий синхросигнал. Таким образом, за один такт входной набор сигналов преобразуется в первом (входном) блоке в набор выходных сигналов и сохраняется в первом регистре триггеров. На следующем такте входной набор обрабатывается во втором блоке, а на вход первого блока поступает следующий входной набор. В течение последующих тактов сигналы с выходов предыдущих блоков поступают на входы последующих и сохраняются в соответствующих регистрах в течение текущего такта. Одновременно схема обрабатывает столько входных наборов, сколько блоков (и, соответственно, регистров) в схеме.

Современные промышленные синтезаторы логических схем по алгоритмическим описаниям их поведения на языках высокого уровня (Verilog, VHDL) позволяют получать логические схемы в целевой библиотеке пользователя (проектировщика). При этом проектировщик может задавать различные технологические ограничения, требования к быстродействию, критерии оптимизации и др. Для некоторых специальных применений требуются логические схемы, обеспечивающие максимальное быстродействие. Такое быстродействие определяется используемой библиотекой логических элементов, однако необходимые для этого опции синтеза в синтезаторах, как правило, отсутствуют. В данной работе достижения максимальных показателей быстродействия комбинационных (без элементов памяти) нерегулярных логических схем предлагается добиваться с помощью конвейеризации.

Представляемая далее программная система ArCon:

— позволяет по структурному описанию проектируемого устройства на языке высокого уровня VHDL (Very high speed integrated circuits Hardware Description

Language) получить структурное VHDL-описание конвейеризованной логической схемы;

— имеет интерактивные средства управления числом блоков конвейеризованной схемы, типом используемых триггеров;

— позволяет определять задержку схемы и составляющих ее блоков, согласно заданным характеристикам элементов библиотеки проектирования.

Метод построения блоков конвейера

В комбинационной логической схеме из библиотечных элементов, выполненных по КМОП-технологии (КМОП-элементов), задержка складывается из задержек элементов самой длинной цепочки [5]. Пусть на вход схемы поступает последовательность p наборов двоичных сигналов. Если T — время задержки схемы, то период смены входных сигналов не может быть меньше T. Время реакции устройства на данную последовательность в этом случае будет не меньше pT.

Разобьем схему на k блоков (C_1, C_2, ..., C_k) и, если τ_C — время задержки самого медленнодействующего блока, то $T \leqslant k\tau_C$. На выходы каждого блока поставим элементы задержки (D-триггеры), пропускающие сигналы с выходов блока по сигналу синхронизации. Этот же сигнал синхронизации определяет период смены сигналов на входе устройства. Этот период должен быть не меньше суммы двух задержек: задержки τ_C и задержки τ_D элемента D ($\tau_{\text{clock}} \geqslant \tau_C + \tau_D$). Теперь время реакции устройства на упомянутую последовательность равно $(k + p)\tau_{\text{clock}}$.

В качестве модели схемы используется бесконтурный орграф $G = (V, A)$ [5]. Его вершины из множества V представляют логические элементы и входные полюсы схемы, а дуги из множества A показывают направления сигналов от выходов одних элементов к входам других элементов. Каждой вершине $v \in V$ приписан вес $\tau(v)$, представляющий задержку соответствующего элемента. Вершины, соответствующие входам схемы, имеют вес, равный нулю.

Для вычисления задержки элемента сети используется модель, рассмотренная в работе [6], в которой задержка рассчитывается как для положительного, так и для отрицательного фронтов входного сигнала для каждого входного полюса элемента без учета связей.

Далее под числом каскадов (уровней) схемы m понимаем максимальное число элементов схемы на пути от входного полюса к выходному. На входы элементов, принадлежащих i-му каскаду, поступают выходные сигналы только с элементов j-х каскадов, где $0 < j < i$. Каскад является минимально возможным блоком схемы.

Сформируем последовательность каскадов L_1, L_2, ..., L_m, представляющую собой упорядоченное разбиение множества вершин V орграфа G, которое характеризуется следующим свойством. Если вершина v принадлежит полуокрестности исхода $N^+(u)$ вершины u, то эти вершины находятся в разных каскадах и каскад, содержащий вершину u, предшествует в этой последовательности каскаду с вершиной v (не обязательно непосредственно). Если длины путей от входов схемы к ее выходам различны, то такое разбиение не является единственным. Следует выбрать такой вариант разбиения на каскады, чтобы сумма весов всех каскадов была по возможности минимальной. Под весом каскада понимаем максимум весов вершин, принадлежащих данному каскаду.

Можно выделить два типа вершин орграфа G. К одному типу отнесем вершины, которые лежат на самых длинных путях в орграфе G. Они строго распределяются по каскадам и не могут менять свое положение. Их назовем *неподвижными*. Положение в каскадах других вершин, которые назовем *подвижными*, можно менять в определенных пределах, например, от каскада L_l до каскада L_r ($l < r$). Эти пределы устанавливают с помощью алгоритма, подобного алгоритму топологической сортировки [7].

Для окончательного распределения вершин по каскадам так, чтобы сумма весов каскадов была по возможности минимальной, предлагается следующий способ. Удалив из орграфа G неподвижные вершины вместе с инцидентными им ребрами, получим орграф H, в каждой компоненте которого выделим вершину с максимальным весом. Эту вершину поместим в один из допустимых для нее каскадов с максимальным весом. Границы положения вершин при этом изменятся, и некоторые вершины из подвижных перейдут в неподвижные. Дальнейшее распределение по каскадам можно вести для каждой компоненты орграфа H описанным выше способом.

Все пути в орграфе приводятся к единой длине с помощью добавления новых вершин с нулевым весом. Каждому из каскадов соответствует множество значений веса, приписанных вершинам, принадлежащих данному каскаду. Максимальное значение веса в этом каскаде представляет собой задержку прохождения в нем сигнала. Заданную комбинационную схему надо разбить на заданное число блоков с минимизацией задержки в самом медленнодействующем блоке. Каждый блок представляет собой упорядоченное множество каскадов.

Рассмотренная задача решается в системе синтеза конвейерных КМОП-схем.

Архитектура системы ArCon синтеза конвейерных логических схем

Основная задача системы синтеза конвейерных логических схем — это преобразование нерегулярных многоуровневых логических схем в конвейерные структуры путем установки регистров с общим сигналом синхронизации, как это показано на рис. 1. Конвейеризованная схема, имеющая k входов x_1, x_2, ..., x_k, и m выходов $f_1, f_2, ..., f_m$, разбита на n блоков, в каждом из них объединены от одного до нескольких каскадов, исходя из значений их задержек. Чем ближе друг к другу значения задержек блоков, тем эффективнее разбиение. Между блоками вставляют регистры, в данном случае D-триггеры, имеющие библиотечное имя DFF. Все триггеры имеют общий синхросигнал *clk*.

Архитектура системы синтеза конвейерных логических схем ArCon представлена на рис. 2. Система

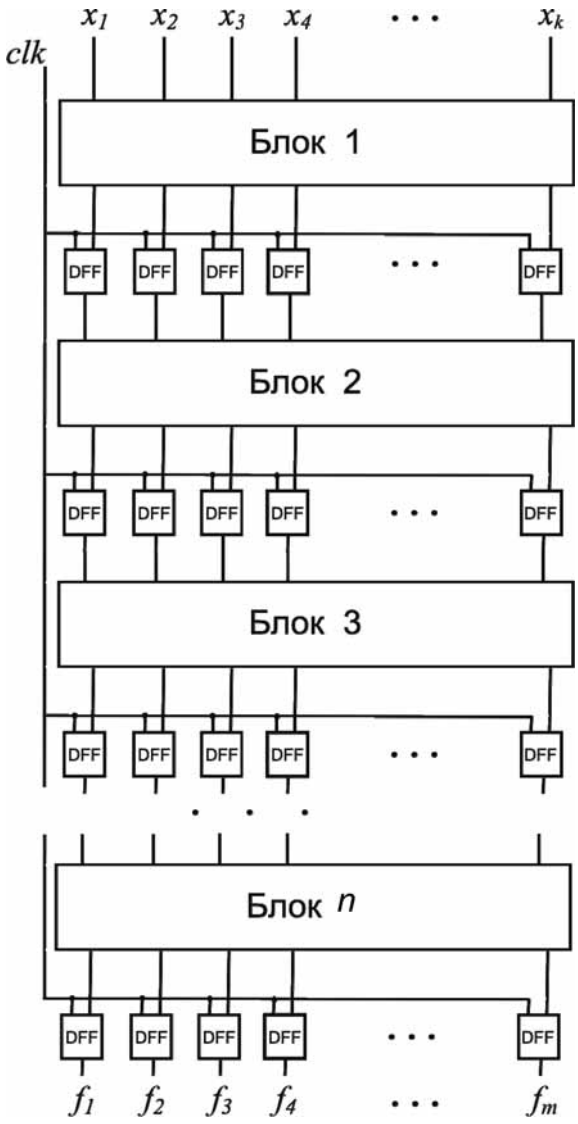


Рис. 1. Схема с конвейерной структурой

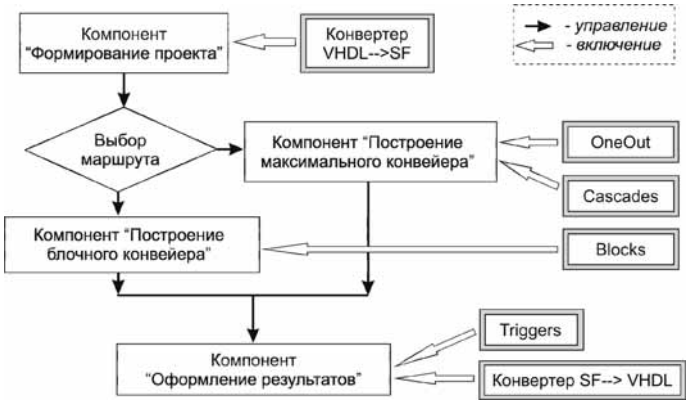


Рис. 2. Архитектура системы синтеза конвейерных логических схем ArCon

состоит из ряда программных компонентов, каждый из которых предназначен для выполнения одного из этапов построения конвейера: формирования внутреннего представления исходной схемы — проекта, построения конвейера тем или иным способом, оформления полученных результатов в требуемой форме.

Формирование проекта осуществляется по VHDL-описанию исходной схемы. Система синтеза использует язык SF [8] в качестве внутреннего языка, на котором проводятся преобразования конвейеризации. Для получения описания схемы на этом языке используется специальный конвертер, на выходе которого схема представляется в форме иерархического описания, в качестве "листьев" которого выступают библиотечные элементы. В рамках компонента "Формирование проекта" осуществляется определение временных задержек для каждого библиотечного элемента, участвующего в описании конвейеризуемой схемы.

В составе системы синтеза конвейерных логических схем реализовано два метода построения конвейера — методы максимальной и блочной конвейеризации.

Метод максимальной конвейеризации предполагает ранжирование элементов схемы по каскадам (топологическую сортировку) и встраивание регистров между элементами каждой пары каскадов, а также перед входами и на выходах схемы. В рамках реализации метода максимальной конвейеризации задействованы программные модули OneOut и Cascades. Следует отметить, что в целях повышения быстродействия схемы и нагрузочной способности элементов модуль OneOut выполняет следующее преобразование. Все элементы схемы, имеющие разветвление на выходе, дублируются столько раз, со сколькими элементами в схеме они связаны. Таким образом, получается вариант схемы, в которой каждый элемент связан только с одним элементом на выходе. Далее для полученного варианта схемы выполняется разбиение элементов на каскады.

Метод блочной конвейеризации предполагает разбиение элементов схемы на заданное число блоков с учетом некоторых критериев. Число блоков не может быть более числа каскадов в схеме. Как правило, блок объединяет несколько соседних каскадов. Элементы блоков также должны быть топологически отсортированы. В рамках реализации этого метода задействован программный модуль Blocks. В отличие от метода максимальной конвейеризации после получения разбиения элементов на каскады выполняется объединение каскадов в блоки с учетом заданных критериев.

На завершающем этапе конвейеризации осуществляется построение описания схемы с триггерами в соответствии с найденными решениями каскадирования или блочного разбиения. Для этого используется программный модуль Triggers и специальный конвертер, преобразующий описание схемы с языка SF на язык VHDL. Программный модуль Triggers в качестве исходных данных использует структурное описание схемы и информацию о разбиении ее элементов на каскады или блоки. В структуру схемы добавляется столько триггеров заданного типа, сколько имеется связей между элементами соседних блоков, — по одному триггеру на каждую связь.

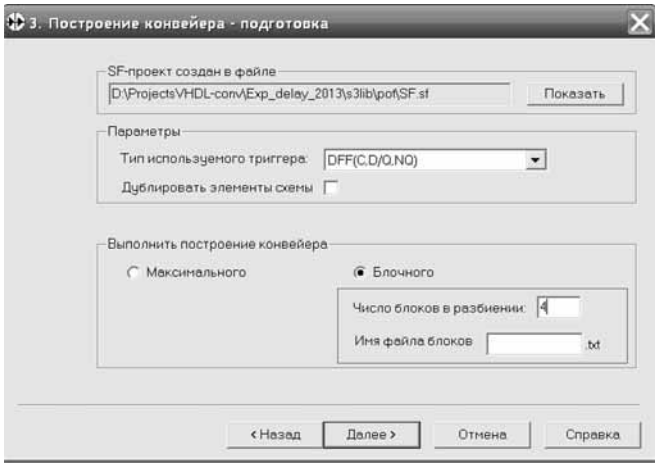


Рис. 3. Закладка одного из состояний сеанса системы ArCon

Конвертер SF → VHDL осуществляет перевод только структурной части описания схемы, поскольку функциональные описания компонентов доступны на языке VHDL в форме VHDL-библиотеки КМОП-элементов, используемой для проектирования заказных СБИС.

Отдельный сеанс работы системы синтеза конвейерных схем строится в форме последовательного предъявления пользователю цепочки заполняемых бланков-закладок, на каждом из которых представлены информационные поля и средства управления. Пример одной из закладок представлен на рис. 3.

Экспериментальное исследование энергопотребления конвейеризованных схем

Быстродействие результирующей конвейеризованной логической схемы определяется задержкой самого "медленного" блока в схеме. Эта задержка и определит длину периода синхросигнала для триггеров. Синхронизация является общей, поэтому в каждом такте на вход начального блока может быть подана следующая комбинация входных сигналов. Таким образом, конвейер позволяет выдавать в каждом такте выходные сигналы.

Было проведено экспериментальное исследование системы ArCon на потоке практических примеров. Схемы для экспериментов выбирались из набора тестовых примеров [9], в качестве библиотеки логических элементов использовалась библиотека КМОП-элементов, представленная в работе [10], в качестве триггеров были выбраны синхронные *D*-триггеры. Целью экспериментов являлось определение характеристик схем после проведения конвейеризации, а именно насколько увеличится сложность схемы и потребляемый ток. Характеристики конвейеризованных схем исследовались в режимах блочного и максимального конвейеров.

Измерение потребляемого схемой тока осуществлялось с помощью схемотехнического моделирования описаний схем на транзисторном уровне в системе Accusim (программный комплекс фирмы Mentor Graphics). Для выполнения схемотехнического моделирования использовались тесты из 32 наборов случайных (с вероятностью 0,5) входных сигналов.

Результаты экспериментов представлены в табл. 1, где:

n — число входных полюсов в схеме;

m — число выходных полюсов в схеме;

L — число элементов в схеме;

p — число блоков схемы, в неконвейеризованной схеме — один блок, пометка (max) означает, что блоками являются каскады схемы;

d — задержка схемы (для конвейеризованных схем — задержка максимального блока схемы);

R — суммарное число транзисторов во всех элементах схемы;

A — средний потребляемый схемой ток.

Результаты эксперимента представлены по восьми примерам. Эксперимент показал, что в конвейеризованных схемах резко возрастает число транзисторов, и, как следствие, энергопотребление. Показателем возрастания энергопотребления назовем отношение значения среднего потребляемого конвейеризованной схемой тока к значению среднего потребляемого неконвейеризованной схемой тока (соответствующие значения среднего потребляемого тока представлены в столбце *A* табл. 1). Показателем увеличения быстродействия назовем отношение значения задержки неконвейеризованной схемы к значению максимальной задержки блока для конвейеризованной схемы (соответствующие значения задержек представлены в столбце *d* табл. 1).

Таблица 1

Экспериментальное исследование энергопотребления *A* конвейеризованных схем

Вид схемы	*n*	*m*	*L*	*p*	*R*	*d*, нс	*A*, мА
b2	15	17	545	**1**	**2878**	**23,98**	**2,5**
			1357	3	28 862	7,78	29,28
			2108	6	52 894	4,5	54,34
			2839	**10**	**76 286**	**2,64**	**76,74**
			7543	20 (max)	233 400	1,77	214,0
Chkn	29	7	278	1	1508	21,5	1,16
			1063	5	26 628	5,43	29,56
			1535	10	41 732	2,55	43,47
			3028	19 (max)	88 204	1,83	94,87

Продолжение табл. 1

Вид схемы	*n*	*m*	*L*	*p*	*R*	*d*, нс	*A*, мА
dist	8	5	252	1	1408	14,99	1,12
			684	3	15 232	5	15,83
			1097	6	28 448	2,55	29,20
			2766	13 (max)	78 478	1,46	84,89
m2	8	16	174	1	844	14,7	0,48
			500	4	11 276	4,97	9,89
			911	9	24 428	2,35	21,71
			1868	14 (max)	53 418	1,59	51,03
m3	8	16	218	1	1084	19,75	0,71
			773	5	18 844	5,06	16,8
			1199	11	32 476	2,42	28,57
			2941	17 (max)	85 044	1,54	82,39
rckl	32	7	99	1	486	18,2	0,46
			496	6	13 190	3,7	15,9
			688	9	19 334	2,5	22,36
			1078	18 (max)	32 902	1,65	36,38
sym10	11	1	118	1	660	14,27	0,85
			507	5	13 108	3,9	15,3
			787	10	22 068	2,09	24,8
			1365	15 (max)	39 602	1,65	46,06
tms	8	16	132	1	626	17,55	0,34
			485	5	11 922	3,49	10,7
			780	10	21 362	2,21	18,99
			1588	16 (max)	46 188	1,83	42,41

Сравним характеристики двух вариантов схемы b2: неконвейеризованного ($p = 1$) и конвейеризованного в режиме блочного конвейера ($p = 10$), они выделены полужирным шрифтом в табл. 1. Показатель увеличения быстродействия рассматриваемого варианта конвейеризованной схемы равен 9,08 (23,98:2,64). Показатель увеличения энергопотребления равен 30,7 (76,74:2,5). Число транзисторов схемы, представленное в столбце *R* табл. 1, увеличивается в 26 раз (76 286:2878).

В табл. 2 представлены значения показателей возрастания энергопотребления и увеличения быстродействия для трех вариантов конвейеризации схемы: максимальной (первая строка для каждого примера), средней (вторая строка) и минимальной конвейеризации (третья строка). Максимальная конвейеризация соответствует режиму максимального конвейера, когда схема разбивается на максимально возможное число блоков. При средней конвейеризации заданное число блоков разбиения равно приблизительно половине от максимального числа блоков. При минимальной конвейеризации заданное число блоков разбиения равно приблизительно одной четвери от максимального числа блоков.

Из табл. 2 видно, что показатель возрастания энергопотребления растет значительно быстрее, чем показатель увеличения быстродействия. Например, для схемы b2 (режим максимального энергопотребления) при увеличении быстродействия схемы в 13,55 раза энергопотребление возрастает в 85,6 раза. Часто такое увеличение энергопотребления может не устроить разработчиков схем. По этой причине реализован режим блочного конвейера, когда можно определить компромисс между быстродействием и энергопотреблением схемы путем варьирования числа блоков в разбиении схемы.

Определим коэффициент возрастания энергопотребления, как отношение показателя роста энергопотребления к показателю увеличения быстродействия схемы. Данные о зависимости коэффициента возрастания энергопотребления от числа блоков разбиения представлены в табл. 2. Динамика изменения коэффициента возрастания энергопотребления для максимального, среднего и минимального режимов конвейеризации исследуемых схем представлена на рис. 4.

Таблица 2

Зависимость коэффициента возрастания энергопотребления от числа блоков разбиения

Вид схемы	Число блоков разбиения	Показатель возрастания энерго-потребления	Показатель увеличения быстро-действия	Коэффициент возрастания энерго-потребления
b2	20	85,60	13,55	6,32
	10	30,70	9,08	3,38
	3	11,71	3,08	3,80
chkn	19	81,78	11,75	6,96
	10	37,47	8,43	4,44
	5	25,48	3,96	6,44
dist	13	75,79	10,27	7,38
	6	26,07	5,88	4,44
	3	14,13	3,00	4,71
m2	14	106,31	9,25	11,50
	9	45,23	6,26	7,23
	4	20,60	2,96	6,97
m3	17	116,04	12,82	9,05
	11	40,24	8,16	4,93
	5	23,66	3,90	6,06
rckl	18	79,09	11,03	7,17
	9	48,61	7,28	6,68
	6	34,57	4,92	7,03
Sym10	15	54,19	8,65	6,27
	10	29,18	6,83	4,27
	5	18,00	3,66	4,92
tms	16	124,74	9,59	13,01
	10	55,85	7,94	7,03
	5	31,47	5,03	6,26

Коэффициент возрастания энергопотребления имеет тенденцию к уменьшению при уменьшении числа блоков разбиения. Задача проектировщика состоит в том, чтобы найти оптимальное разбиение схемы на блоки (с минимальным значением коэффициента возрастания энергопотребления) для заданного значения задержки схемы.

Заключение

Быстродействие нерегулярных логических схем из библиотечных элементов можно повысить, используя конвейерную структуру. Система синтеза конвейерных логических схем ArCon позволяет выполнять преобразование нерегулярных многоуровневых логических схем в конвейерные структуры путем установки триггеров с общим сигналом синхронизации. В качестве подзадач в системе выступают разбиение элементов схемы на блоки и определение функций и временных задержек каждого блока. Экспериментальные исследования показали, что путем конвейеризации можно добиться значительного увеличения быстродействия схемы, однако при этом сильно возрастает сложность схемы и ее энергопотребление. Проектировщик имеет возможность выбрать вариант разбиения схемы на блоки для увеличения ее быстродействия.

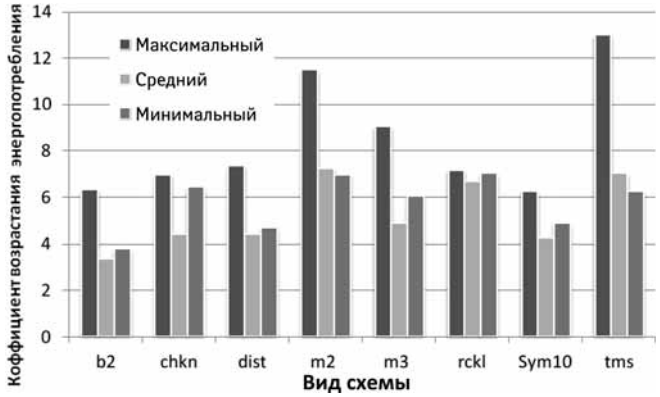


Рис. 4. Динамика изменения коэффициента возрастания энергопотребления для максимального, среднего и минимального режимов конвейеризации

Список литературы

1. **Каляев И. А., Левин И. И., Семериков Е. А., Шмойлов В. И.** Реконфигурируемые мультиконвейерные вычислительные структуры / Под общ. ред. И. А. Каляева. Ростов-на-Дону: Изд-во ЮНЦ РАН, 2008. 320 с.

2. **Кухарев Г. А., Шмерко В. П., Зайцева Е. Н.** Алгоритмы и систолические процессоры для обработки многозначных данных. Минск: Навука і тэхніка, 1990. 296 с.

3. **Воеводин В. В.** Математические модели и методы в параллельных процессах. М.: Наука. Гл. ред. физ.-мат. лит., 1986. 296 с.

4. **Капитонова Ю. В., Летичевский А. А.** Математическая теория проектирования вычислительных систем. М.: Наука, 1988. 296 с.

5. **Поттосин Ю. В., Кардаш С. Н.** Повышение быстродействия комбинационной схемы путем конвейеризации // Информатика. 2013. № 1. С. 1—9.

6. **Бибило П. Н., Авдеев Н. А.** VHDL. Эффективное использование при проектировании цифровых схем. М.: СОЛОН-ПРЕСС, 2006. 344 с.

7. **Кнут Д.** Искусство программирования для ЭВМ. Т. 1. Основные алгоритмы. М.: Мир, 1976.

8. **Бибило П. Н.** Кремниевая компиляция заказных СБИС. Минск: Ин-т техн. кибернетики АН Беларуси, 1996. 268 с.

9. **Berkeley PLA** test set [Electronic resorce]. Mode of access: http://www1.cs.columbia.edu/~cs4861/sis/espresso-examples/ex/.

10. **Бибило П. Н., Кириенко Н. А.** Оценка энергопотребления логических КМОП-схем по их переключательной активности // Микроэлектроника. 2012. Т. 41, № 1. С. 65—77.

УДК 004.06

Е. М. Лаврищева, д-р физ.-мат. наук, проф., КНУ имени Тараса Шевченко, г. Киев, Украина, e-mail: Lavrischeva@gmail.com

Развитие идей академика В. М. Глушкова по технологии компьютеров, систем и программ

Статья посвящена 90-летию со дня рождения академика Академии наук Украины и СССР Виктора Михайловича Глушкова, она отражает его вклад в разработку технологии ЭВМ, компьютерных систем и программ. В 1970-х годах он предвидел, что появятся фабрики по созданию компьютеров, систем и программ, которые будут работать по принципу сборки, как в автомобильной промышленности на заводах Г. Форда. Со временем его предвидение оправдалось. Развивая концепцию сборки Глушкова, автор формализовала технологию сборки разнородных модулей в сложные программы и дала классификацию новых дисциплин, способствующих более наукоемкому и обоснованному созданию качественных программных продуктов для массового использования.

Ключевые слова: *технология программирования, технология компьютеров, систем и программ, фабрики программ, индустрия, методы и инструменты*

E. M. Lavrishcheva

Development of Academics Ideas of V. M. Gluchkov from the Technologies of Computers, Systems and Programs

An article is dedicated to 90-year of birth day of academician of NAN Ukraine and USSR Victor Michalovich Gluchkov, to his deposit in development of technology of ECM, computer systems and programs. In the 70-years of past century he foresaw, that the factories of computers, systems and programs, which will work on principle of assembling will appear, as in motor-car industrial factory of Ford. In course of time his foresight were justified. Offered them technologies are adequate to technologies, which appeared simultaneously in Computer Sciences. One of technologies — Software Engineering technology of programming is identical. Developing the Gluchkov's ideas, the author realized the technology of assembling ready components for the heterogeneous systems and gave a new classification of SE disciplines, oriented on industry of software products on the factory.

Keywords: *technology of programming, technology of computers, systems and programs, factories of the programs, industry, methods*

Академик Виктор Михайлович Глушков посвятил свою жизнь созданию отечественной кибернетической школы. Он оставил много парадигм, которые стали основой для формирования новых современных научных направлений, а также концептуальных положений, в значительной степени определивших последующее развитие кибернетики и информатики. Его ученики Ю. В. Капитонова и А. А. Летичевский к 80-летию В. М. Глушкова написали монографию [1], в которой выделили семь научных парадигм Глушкова: математическая теория проектирования ЭВМ; самоорганизация и совершенствование компьютерных систем; теория доказательства теорем, алгебраическое программирование для вычисления математических задач; принципы и методы построения информационных систем, автоматизированных систем управления (АСУ) и АСУ технологическими процессами (АСУ ТП); искусственный интеллект и концепция повышения внутреннего языка ЭВМ до уровня интеллекта человека; принципы диалогового взаимодействия человека с компьютерной и информационной средой; экономические модели и пути их совершенствования в системах управления.

К списку перечисленных парадигм относится и технология ЭВМ, систем и программ, которую академик В. М. Глушков определил в 1960-х гг., как повышение уровня программирования вычислительных задач на ЭВМ и позже как сборочный конвейерный

Академик Виктор Михайлович Глушков

способ постепенного перехода от ремесленного к промышленному производству компьютеров, систем и программ. Автор данной статьи работала непосредственно с В. М. Глушковым по проблематике технологии создания сложных систем и многие годы развивает его сборочный тезис. Технологию он считал двигателем прогрессивного развития любой науки, в том числе теории создания ЭВМ и программного обеспечения ЭВМ и автоматизированных систем (АСУ, АСУ ТП, САПР и др.).

Под руководством В. М. Глушкова в 1957 г. в Институте кибернетики Академии наук Украины начали создавать новые ЭВМ, теории и технологии, связанные с ними, в их числе:

- теория построения ЭВМ, вычислительных машин для инженерных расчетов "Мир", ЭВМ для управления технологическими процессами предприятий, машин со схемной интерпретацией языков программирования "Украина", интеллектуальных, мозгоподобных ("умных"), рекурсивных, макроконвейерных многопроцессорных машин и др.;
- методы автоматизации процессов создания системного программного обеспечения новых ЭВМ (операционные системы, трансляторы, редакторы и подобные им), больших программных систем, пакетов прикладных программ разного назначения (математического, экономического, транспортного и др.);
- теория и практика АСУ в целом и АСУ ТП на предприятиях СССР как общегосударственной сети управления предприятиями, оборудованными ЭВМ и устройствами сбора, обработки данных на автоматизированных рабочих местах (АРМ).

Виктор Михайлович Глушков рассматривал технологию создания ЭВМ как комплексное проектирование вычислительных систем, технических средств и их базового математического и системного обеспечения. Именно по такой технологии разрабатывалась серия машин "Мир" ("Мир-1", "Мир-2", "Мир-3") для инженерных расчетов с использованием математических методов, формульных алгебраических преобразований, математического доказательства теорем. В это время у В. М. Глушкова возникли концепция интеллектуальной "умной" (мозгоподобной) машинной структуры, частично промоделированной в проекте машины "Украина" [2—4] и позже в программно-техническом комплексе "Маяк" (1985—1991 гг.), а также концепция управляющих ЭВМ для АСУ и АСУ ТП, нашедшая отображение в управляющей машине "Днепр-1" (1962—1964 гг.) и в управляющем вычислительном комплексе (УВК) "Днепр-2" (1965—1969 гг.) [5, 6].

Технологии ЭВМ

При изготовлении первой ЭВМ под руководством академика С. А. Лебедева сформировалась технология проектирования и изготовления *универсальных* ЭВМ. Она совершенствовалась в плане унификации элементной базы и методов сборки отдельных элементов в более сложные структурные компоненты ЭВМ. По этой технологии изготавливались в СССР такие ЭВМ, как МЭСМ, Стрела, БЭСМ, М-20 и др. В совместной работе В. М. Глушкова и В. А. Мясникова были предложены новые виды рекурсивных, микро- и макроконвейерных ЭВМ с новой элементной базой для организации высокопроизводительных вычислений, решения сложных математических и народнохозяйственных задач.

Принципы построения высокопродуктивной суперЭВМ макроконвейерного типа и реализации на ней семейства языков программирования для проектирования вычислительных систем В. М. Глушков обосновал на конференции в Новосибирске (1979 г.) и опубликовал в журнале "Кибернетика", 1981, № 1. Эти концепции и принципы были реализованы (1985—1991 гг.) коллективом отдела В. М. Глушкова (А. А. Летичевский, Ю. В. Капитонова, Г. И. Горлачь, Н. М. Мищенко, С. С. Гороховский и др.) в системе "МАЯК" на макроконвейере ЭВМ ЕС 2701, который был принят Государственной комиссией СССР для задачи ядерного взрыва (1990 г.). На этих идеях построены высокопроизводительные многопроцессорные кластеры СКИТ-3 (2010 г.) и СКИТ-4 (2013 г.) для организации больших вычислительных задач и современный интеллектуальный комплекс "Инпарком" (1987—2013 г.).

Тезис конвейерной сборки сложных программ из готовых программных заготовок с использованием фондов алгоритмов и программ В. М. Глушков высказал на научном семинаре института (1975 г.). Этот тезис сначала получил развитие в системе АПРОП [7, 8], далее — в сборочном программировании и на линиях сборки компьютерных продуктов (машин и систем) на фабриках по производству компьютеров и программ для массового применения [9].

Базовые принципы построения управляющих машин с набором новых устройств и средств для связи с внешними производственными объектами с приоритетным управлением или с использованием устройства сбора и обработки данных в АСУ и АСУ ТП реализованы в УВК "Днепр-2" (1965—1969 гг.) [5, 6].

Перспективной тенденцией В. М. Глушков считал переход от однопроцессорных *фоннеймановских* ЭВМ к многопроцессорным и "умным" интеллектуальным структурам машин [1—4].

Первым шагом на пути создания таких "умных" машин была серия машин для инженерных расчетов "Мир". В ней схемно и программно реализован новый алгебраический язык "Аналитик" для проведения аналитических преобразований, доказательства теорем и решения задач численно-аналитическими методами. На настоящее время на заводе ВУМ (г. Киев) совместно с германскими специалистами изготовлена к 90-летию Глушкова новая интеллектуальная машина, развивающая серию "Мир-3" [10, 11]. Она выполнена под руководством известного специалиста Института кибернетики И. Н. Молчанова. В эту машину встроены программы решения систем линейных, нелинейных, интегральных и дифференциальных уравнений, численного интегрирования задач Коши, краевых задач, кратных интегралов и др. Машина разработана в составе "знание-ориентированного" комплекса "Инпарком". Этот комплекс обеспечивает реализацию новых математических методов и организацию параллельных вычислений для решения сложных математических задач с помощью встроенных методов, описанных в статье И. Н. Молчанова из сборника [11].

Язык "Аналитик" и в наше время постоянно развивается в Институте математических машин и систем НАН Украины. В нем активно используют методы компьютерной алгебры, ориентированной на создание математических моделей для исследования технических, компьютерных объектов и процессов обработки информации [4, 12].

Новым по меркам 1970-х гг. видом ЭВМ для управления технологическими процессами на промышленных предприятиях стал УВК "Днепр-2" [6]. Для него было реализовано оригинальное общесистемное программное обеспечение, необходимое для организации разработки систем управления и проведения вычислений задач, возникающих на автоматизированных промышленных предприятиях. В частности:

— ОС с диалоговым, многопультовым режимом разработки, отладки, выполнения программ и управления вычислениями со сбором и обработкой данных в реальном масштабе времени (разработчики Г. Я. Машбиц, В. И. Конозенко, Е. И. Калайда и др.);

— трансляторы с языков Автокод, АЛГАМС и КОБОЛ (разработчики Е. М. Лаврищева, Л. П. Бабенко, Е. Л. Ющенко и др.), разработанные на основе нового СМ-метода анализа программ [7, 8] с табличным представлением грамматик языков программирования и положивший начало построению синтаксисо-семантических управляемых трансляторов. СМ-метод[1] является усовершенствованием известного метода синтаксического контроля Замельсона и Бауэра. С помощью СМ-метода реализованы семантика общего арифметического блока транслятора АЛГАМС и КОБОЛ (разработчики Л. Г. Усенко и С. Л. Берестовая), компилятор с подмножества языка SQL в ГДР (1972 г.) и др.

Техническое и математическое обеспечение УВК "Днепр-2" описано в документации на систему (общее описание, руководство программиста, инструкции и др.). Это обеспечение прошло успешные испытания в государственной комиссии (1967 г.) под председательством директора Вычислительного центра АН СССР академика А. А. Дородницына.

УВК "Днепр-2" был представлен на Первой международной выставке ЭВМ в Москве в 1969 г. Огромный интерес к комплексу проявили специалисты в области вычислительной техники из США, Франции, Германии (IBM, CDC, CDS, Bull Jeneral Electric и др.). Им хотелось получить от разработчиков этого комплекса, которые выступали в роли экскурсоводов, новые сведения, понять оригинальные идеи, реализованные в его архитектуре (концепцию прерывания, средств связи и управления объектами предприятий, режим разделения времени и др.).

После этой выставки был подписан контракт по поставке УВК "Днепр-2" (вычислительный комплекс "Днепр-21" и управляющий комплекс "Днепр-22") для построения АСУ ТП управления прокатом металла на металлургических комбинатах (г. Берлин, г. Лейпциг). Во внедрении и реализации АСУ ТП принимали участие разработчики комплекса и программного обеспечения [2, 5, 6]. Виктор Михайлович Глушков был членом межгосударственной программы ГДР и СССР по внедрению УВК "Днепр-2" в вычислительных центрах ГДР и по разработке АСУ ТП для металлургии на его основе. Он практически руководил данным проектом, который был успешно выполнен. Комплекс "Днепр-2" был надежной программно-технической базой АСУ ТП для управления металлургическими комбинатами ГДР до 1991 г.

По результатам создания ОС, системы программирования для УВК "Днепр-2" и системы управления процессами обработки данными в реальном времени работы АСУ ТП основные разработчики от института Кибернетики защитили кандидатские диссертации: Е. М. Лаврищева — "СМ-метод анализа языка Автокод и АЛГАМС"; Л. П. Бабенко — "Транслятор с языка КОБОЛ"; В. И. Конозенко — "Принципы реализации многопультовой и диалоговой отладки программ и решения задач"; Кухарчук А. Г. — "Принципы построения управляющего комплекса "Днепр-21" и вычислительного комплекса "Днепр-22". Научные результаты, полученные при выполнении этого проекта, вошли в сокровищницу мировой компьютерной науки [4, 5, 10, 11].

[1] См. статью E. M. Lavrishcheva and E. L. Yushchenko. A method of analyzing programs based on a machine language, URL: http://link.springer.com/article/10.1007%2FBF01068491

Теория построения АСУ была изложена В. М. Глушковым в его монографиях [2, 3], которыми руководствуются многие специалисты и в настоящее время. В 1970-х гг. В. М. Глушков уделял огромное внимание созданию первых АСУ в Украине, Болгарии и ГДР. По его методологии создавались АСУ и информационные системы, к числу которых относятся АСУ ТП Лисичанского химкомбината, Донецкого горно-обогатительного комбината, Львовского телевизионного завода, металлургического комбината ГДР и других предприятий.

Виктор Михайлович Глушков в 1974 г. предложил Кабинету министров СССР проект системы для объединения и централизованного управления всеми автоматизированными системами СССР, в том числе проект объединения АСУ промышленных предприятий страны в общую государственную автоматизированную систему — ОГАС [2, 11]. Тогда этот проект не был поддержан, так как требовал огромных финансовых ресурсов на его реализацию. Следует отметить, что идеологически проект ОГАС, по мнению автора, предвосхитил появление Интернет. Основы теории управления и построения информационных систем (ИС) были изложены В. М. Глушковым в его последней монографии "Безбумажная информатика", продиктованной им в больнице (октябрь 1981 г. — январь 1982 г.) [3]. Многие идеи, принципы и методы В. М. Глушкова были устремлены в будущее. Они развиваются его последователями — учеными и практиками с учетом новых условий, которые предоставляют Интернет и возможности современных общесистемных сред (IBM, MS.Net, Grid и др.). В период глобальной информатизации и компьютеризации методы В. М. Глушкова получили дальнейшее развитие в системе электронного документооборота Академии педагогических наук Украины. По материалам идей, связанных с развитием ИС и систем документооборота в государственных органах управления, разработан учебник для студентов высших учебных заведений. В нем изложены основополагающие идеи В. М. Глушкова в ИС, новые современные подходы к обработке деловой информации и управлению ИС через систему Интернет (см. http://lib.iitta.gov.ua/view/creators).

Технология создания программных продуктов

Появлению современной технологии создания программного продукта предшествовало программирование для решения разных математических задач на первых ЭВМ. Описание программ выполнялось машинным языком, адресным языком Е. Л. Ющенко и языками программирования четвертого поколения, включая Ассемблер, Algol-60, Fortran, PL, Модула, Ада и др. Для их практического использования разрабатывались соответствующие программы для ЭВМ. Идеи и концептуальные положения автоматизации программирования были в центре внимания зав. отделом ИК АН УССР Е. Л. Ющенко. Вместе с аспирантом Г. Е. Цейтлиным им были разработаны теоретические аспекты использования алгебраических методов В. М. Глушкова в программировании, концептуальные положения синтаксического и семантического анализа языков программирования, система алгоритмических алгебр (САА) на основе теории автоматов Глушкова (1972 г.) [10, 12]. В этих работах представлена формальная теория универсальных алгебр, автоматных и алгоритмических САА, контекстно-свободных языков программирования и метаязык СМ-грамматик для реализации семантических программ трансляторов. Первые работы в области теории программирования были выполнены А. П. Ершовым, Г. Е. Цейтлиным и Е. Л. Ющенко [10] и опубликованы повторно, в том числе и на английском языке. Заметим, что Г. Е. Цейтлин постоянно развивал САА, создал теорию алгоритмики, подготовил более десяти кандидатов наук и реализовал систему Мультипроцессист, основанную на идее многопроцессорности компьютеров Глушкова.

В 1975 г. В. М. Глушков предложил перспективный сборочный способ для постепенного перехода от "ремесленного" производства к промышленному выпуску компьютеров, программ и аппаратно-программных систем. Индустрия компьютерных программ, по его замыслу, должна была базироваться на технологических линиях конвейерного изготовления продуктов. Технологии создания компьютеров, информационных и программных систем он считал движущей силой прогресса фундаментальных кибернетических и компьютерных наук.

В работе [12] В. М. Глушков сформулировал три перспективных направления технологии программирования:

- модульная система автоматизации производства программ по методу сборки (АПРОП) программных заготовок-модулей в сложные системы [7, 8, 13];
- метод формализованных технических заданий для проектирования сложных программных комплексов с использованием семейства алгоритмических языков для описания отдельных блоков систем на уровнях последовательной детализации компьютерного проекта Маяк [2];
- Р-технология программирования для автоматизации конструирования структур программ и данных для программно-технических систем средствами графического Р-языка.

В результате поисковых и прикладных исследований на этих научных направлениях были разработаны новые методы, технологии и инструментальные средства. К их числу относятся: Р-технология и стандарт Р-языка в ISO/IEC (И. В. Вельбицкий [14]); метод сборки разнородных модулей в АПРОП; технология систем и пакетов прикладных программ, отдельные пакеты программ математического, экономического, статистического типов (И. В. Сергиенко, А. С. Стукало, И. Н. Редько и др. [15]). Этими работами был внесен

существенный вклад в теорию и практику индустрии создания программных продуктов в СССР на основе метода сборки, сформулированного В. М. Глушковым и реализованного в АПРОП ЕС ЭВМ [7, 8, 13].

Сформировалась технология программирования и новый вид программирования — сборочное программирование для объединения разноязыковых модулей средствами системы АПРОП. Такая система разрабатывалась по договору с Институтом приборостроения (г. Москва) в составе технологии создания программ для бортовых систем "ПРОТВА", которая реализовывалась под руководством В. В. Липаева [16]. Главное нововведение системы АПРОП — интерфейс (межмодульный, межъязыковый и технологический) [8, 13, 17], а также библиотека интерфейсных функций преобразования нерелевантных типов данных, описанных на разных языках и платформах. Впервые автором было определено понятие интерфейса и языка его описания (1976 г.) в проекте АПРОП [7]. Идеи обеспечения связи разноязычных модулей в АПРОП во многом опередили появление зарубежных языков интерфейсов (MIL API, IDL, SIDL и др.). Интерфейс начал использоваться в процессе создания новых программных систем из модулей. Понятие "интерфейс" стало общепризнанным после международной конференции "Интерфейс СЭВ-1987".

Технология сборочного программирования начала формироваться при В. М. Глушкове как средство программирования и сборки сложных прикладных систем, а также семейств трансляторов. Основные положения этой технологии были связаны с выделением общих средств в языках программирования операционной системы единой системы ЭВМ (ОС ЕС), их программной реализацией и сборкой из них трансляторов в системе "Терем" в отделе В. М. Глушкова. В этой системе методом сборки разработанных общих компонентов языковых процессов в классе языков ОС ЕС реализовано семейство языков МАЯК [18].

Становление в СССР сборочного программирования поддерживали академик АН СССР А. П. Ершов [19] и проф. В. В. Липаев в проекте "ПРОТВА" [16]. А. П. Ершов считал, что сборочное программирование эффективно, поскольку готовые программные модули позволяют быстро решить многие задачи из определенной проблемной области.

В дальнейшем сборка стала важным технологическим решением для индустрии создания программных продуктов. Это обстоятельство было подтверждено защитой в 1989 г. автором настоящей статьи докторской диссертации "Методы, средства и инструменты сборочного программирования".

Таким образом, сборочное программирование сложных систем из готовых программных модулей было де-факто сформулировано как новый подход к программированию и, в соответствии с концепцией академика В. М. Глушкова, как основа фабрик для конвейерной разработки программ.

Индустриальная технология сборки программных продуктов

Методика создания технологических линий (ТЛ) предложена учениками В. М. Глушкова, включая автора, в 1987 г. [20]. Она апробирована на шести линиях автоматизированной ИС "Юпитер-470" Института кибернетики АН УССР для военно-морского флота СССР (1983—1991 гг.). Эти ТЛ стали первой практической реализацией идеи сборки ТЛ из процессов и операций, которые отображаются в маршрутной схеме ТЛ.

Фабрики программ разрабатывают как инфраструктуру создания в промышленном режиме программных продуктов с заданными функциями, структурой и качеством. В основе фабрики — большое число разного рода готовых к использованию ресурсов и средства разработки из них продуктов различного уровня сложности и назначения. Основные механизмы фабрики — ТЛ, задающие порядок разработки сложной продукции из готовых ресурсов, которые находятся в хранилище (складе) фабрики или, при необходимости, подбирают из разных библиотек и репозитариев Интернет.

Исходя из опыта автоматизированной сборки разнородных программ в АПРОП и анализа современных зарубежных фабрик индустриального типа [21], автор сформулировала общий набор элементов, которые характеризуют любую фабрику программ:

- готовые ресурсы (артефакты, программы, сервисы, многократные компоненты и т. п.);
- спецификаторы интерфейсов (паспортных данных готовых программных ресурсов), которые описывают в одном из языков интерфейса (IDL, API, SIDL и др.);
- операционная среда, которая содержит программные средства и инструментарий для системной сборки разнородных ресурсов;
- технологические и продуктовые линии производства программной продукции;
- метод проектирования, разработки и сборки готовых ресурсов;
- сборочный конвейер производства программ [21—23].

Следует отметить, что к настоящему времени сложились необходимые условия для решения научных и технических задач в рамках Европейских проектов Grid. В рамках этих проектов функционируют фабрики программ системного и прикладного характера с представленными выше элементами производства программных продуктов [24—27].

Построение ТЛ выполняется на этапе технологической подготовки работ для создания специальной схемы линии (технологического маршрута) с помощью процессов и операций, которые обеспечивают продуцирование элементов системы средствами языков программирования или комплекса соответствующих средств [27, 28]. Линии комплектуют из процессов ТЛ, которые отвечают задачам реализации будущей предметной области, стандартных инструментальных средств, технологических модулей и комплекса соот-

ветствующего нормативно-методического обеспечения. При этом могут дополнительно подбираться готовые прикладные ресурсы, инструментальные средства для реализации отдельных функций или элементов программ. Согласно положениям SWEBOK процессами являются анализ требований, конструирование, разработка, тестирование, эксплуатация и модернизация. С помощью соответствующих технологических ресурсов их поддержки и видов обеспечения формируется технологический маршрут линии для выполнения отдельных задач разработки и сборки программных продуктов.

Все ресурсы и процессы связаны между собой технологическим маршрутом, который устанавливает порядок операций и процессов, выполняемых разными видами обеспечения (информационным, методическим, математическим и программным). На конечной операции маршрута выполняется операция оценивания качества продукта по принятой методике. Набор процессов формируется с учетом международных стандартов ISO/IEC 12207—96, 2007 и ГОСТ 3918—99. Процессы ТЛ описывают специальным языком со ссылками на соответствующие инструментальные средства, технологические модули и правила управления деятельностью специалистов по выполнению отдельных операций таких процессов.

Построенные ТЛ стали первым вариантом промоделированного сборочного конвейера Глушкова. С помощью ТЛ было создано более 500 программ обработки данных для разных объектов автоматизированной ИС "Юпитер".

Позже (2004 г.) появился альтернативный подход — линии продуктов Института программной инженерии США (http://sei.cmu.edu/productlines/frame_report/). Этот подход основан на интеграции ранее разработанных программных продуктов в семейство продуктов с помощью модели характеристик, общей для членов этого семейства. Подход используется при коммерческой сборке готовых программных продуктов, которые написаны на "старых" традиционных языках программирования. Такие продукты, как правило, сложны для понимания и их трудно модифицировать и эксплуатировать. По этой причине инициаторы этого подхода выдвинули концепцию вариабельности программного продукта (способности продукта к изменениям, адаптация продукта). Согласно ее положениям, к уже созданным продуктам применяют методы реинженерии и реверсной инженерии программных продуктов. Такие методы помогают решать некоторые вопросы, обусловленные сложностью больших программных продуктов, созданных с использованием традиционных технологий. Уменьшению сложности реализации больших программ может способствовать их сборка из готовых стандартизованных программных элементов и объектный подход Г. Буча. В подобной сборке могут быть использованы готовые ресурсы, находящиеся в репозиториях фирм производителей программных продуктов или в сети Интернет. Использование этих подходов значительно упрощает разработку больших систем и снижает сложность их реализации.

Следует отметить, что объектно-ориентированные языки программирования (ООП) и системы программирования в современных операционных средах еще не получили должного распространения в режимах промышленной реализации. Более распространенным подходом при производстве больших систем стал сборочный конвейер — *continuos integration* М. Фаулера (2007 г.), который поддерживается в ряде коммерческих проектов компании ЕПАМ. Такие продукты не способны к автоматизированному управлению изменениями, так как создаются не с использованием ООП, а традиционными методами.

Хотелось бы отметить, что вопросу индустрии программных и информационных систем большое внимание уделяет правительство Украины. По его инициативе и при поддержке 17—18 ноября 2011 г. и 25—27 октября 2012 г. были проведены два международных научных конгресса по инфраструктуре электронного правительства и индустрии программных продуктов. Отмечено, что эта индустрия развивается в основном зарубежными фирмами, которых в Украине более 1000. Они создают продукты силами студентов университетов и институтов. Такие продукты находят потребителей, в том числе и в Украине [28]. На конгрессе поддержана концепция отечественной сборочной технологии и продемонстрирована экспериментальная студенческая фабрика программ Киевского национального университета им. Т. Шевченко (КНУ).

Анализ фабрик программ показал [28, 29], что в мире разработан спектр технологий, претендующих на индустрию программных продуктов. Это мультитехнология К. Чернецкого и К. Айзенекера с лейтмотивом "от ручного труда к конвейерной сборке"; технология И. Бея с автоматизированным взаимодействием разноязычных программ; потоковая сборка — *use case UML*-фабрики программ Дж. Гринфильда (США), Г. Ленца (Германия) и С. Авдошина (Россия); сборочный конвейер М. Фаулера и компании ЕПАМ и др. Общее, что их объединяет — линии (схемы) сборки различных видов программ для массового использования. Для поднятия уровня индустриального производства программных продуктов в отделе "Программная инженерия" Института программных систем НАН Украины (ИПС НАНУ) в рамках фундаментальных проектов и диссертационных исследований были разработаны следующие новые теоретические положения и технологии [21, 22, 30—32]:

- объектного моделирования предметных областей с использованием теории Г. Буча и определения новых функций объектного анализа и алгебры операций;
- компонентного проектирования программных систем с помощью оригинальных моделей компонентов, сред и систем, а также внешней и внутренней компонентной алгебры;
- моделей и методов генерации, трансформации и конфигурации готовых компонентов с точками вариантов в интерфейсах в вариабельные и интеропера-

бельные семейства систем для выполнения в гетерогенных средах [31];

- сервисно-компонентного проектирования распределенных прикладных систем с использованием моделей SOA и SCA;
- онтологического представления жизненного цикла (ЖЦ) программной системы стандарта ISO/IEC 12207, общих типов данных стандарта ISO/IEC 11404 для их реализации в целях генерации новых вариантов ЖЦ и данных для конкретных предметных областей;
- тестирования отдельных элементов программ и процессов, а также их оценки на зрелость по модели CMM и качества по стандартным моделям качества.

Основные аспекты технологий представлены линиями (спектром линий) в инструментально-технологическом комплексе (ИТК) ИПС. Каждая из этих линий повышает производительность изготовления как отдельных элементов продукта, так и продукта в целом, улучшает условия работы исполнителей, сокращает число сборщиков и снижает себестоимость выпускаемой продукции [22].

Фабрика программ к 90-летию академика В. М. Глушкова

Концепция сборочного конвейера Глушкова реализована с участием студентов факультета кибернетики КНУ и МФТИ в виде экспериментальной фабрики программ (URL: http://programsfactory.univ.kiev.ua). Автор осуществляла научное руководство реализацией этого проекта на практических занятиях.

Министерством образования Украины в соответствии с программой Curricula-2004 в 2006 г. было введено обучение студентов вузов Украины по дисциплине "Программная инженерия" наряду с курсом "Технология программирования" [22, 33, 34]. Студенты КНУ и филиала кафедры МФТИ изучают основы этих дисциплин с помощью электронного учебника, представленного студентами на веб-сайте фабрики. Они выполняют лабораторные работы по тематике технологии программирования программной инженерии [24, 26, 35]. Для обучения студентов программной инженерии автором разработаны учебники на русском и украинском языках [20, 33, 34]. Ряд студентов приняли участие в разработке фундаментального проекта ИПС НАН Украины по теории и технологии программных продуктов (2002—2011 гг.). В рамках проекта разработаны новые теоретические положения объектного, компонентного, генерирующего программирования. Усовершенствована практика разработки систем из объектов, компонентов и сервисов на фабриках. Отдельные результаты этой деятельности представлены в журнале "Кибернетика и системный анализ" и переведены на английский язык [9]. По итогам конкурса учебников по программной инженерии, проведенной фирмой Майкрософт-МГУ в Москве в 2006 г., учебник Е. М. Лаврищевой и В. А. Петрухина [33] опубликован при поддержке Министерства образования России и представлен в Интернете на сайте www.intuit.ru

Страна или регион	Посещения	Посещения, %
1. Ukraine	714	61,93 %
2. Russia	291	25,24 %
3. Kazakhstan	25	2,17 %
4. Belarus	24	2,08 %
5. (not set)	22	1,91 %
6. India	11	0,95 %
7. United States	9	0,78 %
8. Moldova	6	0,52 %
9. Germany	5	0,43 %
10. Latvia	5	0,43 %

просмотреть весь отчет

Google-статистика веб-сайта ИТК (II квартал 2013 г.)

Фабрика программ демонстрировалась на ряде международных конференций [29, 36, 37]. Она включена в состав комплекса ИТК ИПС веб-сайта http://sestudy.edu-ua.net

Фабрика оборудована линиями, созданными студентами. В их числе технологии программирования на языках C#VS.Net, Java; построения программных артефактов и компонентов для информационных и программных систем, их сертификация согласно стандарту W3C и сохранение в репозитории; сборки готовых компонентов и компонентов повторного использования (КПИ) в сложные структуры программных систем; трансформации передаваемых типов данных между КПИ; метрического анализа и оценки качества составных элементов и программных систем, обеспечивающие взаимодействие систем между собой (VS.Net-Java, Java-Corba, VS.Net-Corba); поддерживающие процессы обучения теоретическим и прикладным аспектам программной инженерии по электронному учебнику. С декабря 2011 г. к фабрике обратилось более 10 000 пользователей из разных стран (см. рисунок).

Участвующие в работе по этой тематике студенты выполнили дипломные работы, написали магистерские диссертации и статьи [23, 28, 37, 38].

С участием студентов фабрика ИТК пополнена такими новыми линиями, как генерация прикладных систем в языке DSL (Domain Specific Language); онтология ЖЦ стандарта ISO/IEC 12207, трансформация общих типов данных GDT стандарта ISO/IEC 11404 к фундаментальным типам данных FDT; веб-сервисы для построения распределенных программных систем из готовых ресурсов, включая сервисы.

Результаты развития идей В. М. Глушкова

Технология программирования, инициированная академиком В. М. Глушковым, стала главной стратегической линией исследований и разработок отдела

Программная инженерия" (с 1980 г.) в ИПС НАН Украины. Сотрудниками отдела, аспирантами и студентами сформулирован ряд новых научных концепций и методов, используемых в технологии программирования (на основе которых написаны пять кандидатских и одна докторская диссертация), опубликовано более 50 статей. Веб-сайты ИТК ИПС и фабрики программ уникальны в плане представления фундаментальных основ понятия "Программная инженерия" и реализации концепций, моделей, методов и технологий создания программных систем.

Список литературы

1. **Капитонова Ю. В., Летичевский А. А.** Парадигмы и идеи академика В. М. Глушкова. Киев: Наукова думка, 2003. 355 с.

2. **Глушков В. М.** Кибернетика, ВТ, информатика (АСУ). Избр. тр. в 3-х томах. Киев: Наукова думка, 1990. 262 с., 267 с., 281 с.

3. **Глушков В. М.** Основы безбумажной информатики. М.: Наука, 1982. 552 с.

4. **Системы** компьютерной алгебры семейства АНАЛИТИК. Теория, реализация, применение: сб. науч. тр. / под ред. А. А. Морозова, В. П. Клименко, А. Л. Ляхова. Киев: НПП Интерсервис, 2010. 762 с.

5. **Лаврищева Е. М., Никитин А. И., Усенко Л. Г.** и др. АКД — Автокод машины "Днепр-2". Киев: Ин-т кибернетики АН УССР, 1969. 97 с.

6. **Управляющая** вычислительная система "Днепр-2" / под ред. А. Г. Кухарчука и В. М. Египко. Киев: Наукова думка, 1972. 260 с.

7. **Глушков В. М., Лаврищева Е. М., Стогний А. А.** и др. Система автоматизации производства программ (АПРОП). Киев: Ин-т кибернетики АН УССР, 1976. 134 с.

8. **Лаврищева Е. М., Грищенко В. Н.** Связь разноязыковых модулей в ОС ЕС. М.: Финансы и статистика, 1982. 127 с.

9. **Lavrischeva K. M.** Theory and practice of software factories // Cybernetics and Systems Analysis. 2011. Vol. 47, Is. 6. P. 961—972.

10. **Глушков В. М., Цейтлин Г. Е., Ющенко Е. Л.** Алгебра. Языки. Программирование. Киев: Наукова думка, 1974. 318 с.

11. **В. М. Глушков:** прошлое, устремленное в будущее. Сб. трудов. Киев: Академпериодика, 2013. 290 с.

12. **Глушков В. М.** Фундаментальные основы и технология программирования // Программирование. 1980. № 2. С. 3—13.

13. **Лаврищева Е. М., Грищенко В. Н.** Сборочное программирование. Киев: Наукова думка, 1991. 213 с.

14. **Вельбицкий И. В., Ходаковский В. Н., Шолмов Л. И**. Технологический комплекс автоматизации программ на машинах ЕС ЭВМ и БЭСМ-6. М.: Финансы и статистика, 1980. 253 с.

15. **Редько В. Н., Сергиенко И. В., Стукало А. И.** Пакеты прикладных программ. Киев: Наукова думка, 1992. 317 с.

16. **Липаев В. В., Позин Б. А., Штрик А. А.** Технология сборочного программирования. М.: Машиностроение, 1992. 272 с.

17. **Лаврищева Е. М., Грищенко В. Н**. Сборочное программирование. Основы индустрии программных продуктов. Киев: Наукова думка, 2009. 319 с.

18. **Мищенко Н. М.** О сборочном программировании языковых процессоров // Интеллектуализация программного обеспечения информационно-вычислительных систем. Сб. трудов. Киев: Ин-т кибернетики им. В. М. Глушкова АН УССР, 1990. С. 45—52.

19. **Ершов А. П.** Научные основы доказательного программирования. Научное сообщение в Президиуме АН СССР на звание академика СССР, 1984. 11 с.

20. **Бабенко Л. П., Лавріщева К. М.** Основи програмної інженерії. Киев, Знання, 2001. 269 с.

21. **Андон П., Лавріщева К.** Розвиток фабрик програм в інформаційному світі // Вісник НАНУ. 2010. № 10. С. 15—41.

22. **Лавріщева К. М., Коваль Г. І., Бабенко Л. П.** і др. Нові теоретичні засади технології виробництва сімейств ПС у контексті ГП. Електронна монографія. ДНТІ України, ВИНИТИ Росії та ДНТБ, 2012. 277 с.

23. **Аронов А. О., Дзюбенко А. І.** Підхід до створення студентської фабрики програм // Проблеми програмування. 2011. № 3. С. 87—93.

24. **Лавріщева К. М.** Компонентне програмування. Теорія і практика // Проблеми програмування. 2012. № 1. С. 3—14.

25. **Лавріщева К. М.** Генерувальне програмування ПС і сімейств // Проблеми програмування. 2009. № 1. С. 3—16.

26. **Лаврищева Е. М., Зинькович В. М., Колесник А. Л.** и др. Инструментально-технологический комплекс разработки и обучения приемам производства программных систем, (укр.). Гос. служба интеллектуальной собственности Украины. Свид. о регистрации авторского права. № 45292, от 27.08.2012. 103 с.

27. **Лавріщева К. М.** Онтологічне подання життєвого циклу ПС для загальної лінії виробництва програмних продуктів // X Международная научно-практическая конференция TAAPSD-2013. Теоретические и прикладні аспекты разработки программных систем. 25 травня — 2 червня, 2013 р. Україна, Ялта. Кіровоград: ПП "Центр оперативної поліграфії "Авангард", 2013. С. 81—90.

28. **Андон П. І., Лавріщева К. М.** Методологія побудови ліній виробництва програмних продуктів і їх практичне застосування // Міжнародний науковий конгресс "Інформаційне суспільство в Україні". 24—25 жовтня, 2012. Україна. Кіев. Держ агентство з питань науки, інновацій та інформатизації України. URL: http://www.ict-congress.com.ua/attachments/article/102

29. **Kolesnyk A., Clabospitskaya O.** Tested Approach for Variability Management Enhancing in Software Product Line // ICTERI 2012, the 8th International Conference on ICT in Education, Research, and Industrial Applications. 6—10 June 2012. Ukraine, Kherson. URL: http://ceur-ws.org/Vol-848/ICTERI-2012-CEUR-WS-paper-31-p-155-162.pdf

30. **Лавріщева К. М.** Програмна інженерія. Киев: Академперіодика, 2009. 371 с.

31. **Андон Ф. И., Коваль Г. И., Коротун Т. М.** и др. Основы инженерии качества программных систем. Киев: Академпериодика, 2007. 680 с.

32. **Лаврищева Е. М.** Основы технологической подготовки разработки и прикладных программ СОД. Препринт 87-5 ИК АН УССРЮ. 1987. 30 с.

33. **Лаврищева Е. М., Петрухин В. А.** Методы и средства инженерии программного обеспечения. М.: МОН РФ, 2007. 415 с.

34. **Лаврищева Е. М.** Методы программирования. Теория, инженерия, практика. Киев: Наукова думка, 2006. 451 с.

35. **Лавріщева К. М., Слабоспицька О. О., Коваль Г. І., Колесник А. О.** Теоретичні аспекти керування варіабельністю в сімействах програмних систем // Вісник КГУ, серія фіз.-мат. наук. 2011. № 1. С. 151—158.

36. **Lavrischeva E., Ostrovski A., and Radetskyi I.** Approach to E-Learning Fundamental Aspects of Software Engineering // ICTERI 2012, the 8th International Conference on ICT in Education, Research, and Industrial Applications. 6—10 June, 2012. Ukraine, Kherson. URL: http://ceur-ws.org/Vol-848/ICTERI-2012-CEUR-WS-paper-17-p-176-187.pdf

37. **Lavrischeva E., Dzubenko A., Aronov A.** Conception of Programs factory for Representation and E-learning Disciplines of Software Engineering // ICTERI 2013, the 9th on ICT in Education, Research, and Industrial Applications. 19—22 June, 2013. Ukraine, Kherson. URL: http://ceur-ws.org/Vol-1000/ICTERI-2013-p-252-263.pdf

УДК 004.94

Д. Н. Кобзаренко, д-р техн. наук, зав. лаб., ФГБУН Институт проблем геотермии Дагестанского научного центра РАН, г. Махачкала,
e-mail: kobzarenko_dm@mail.ru

К созданию средств автоматизации выборки данных ветромониторинга с сервера "Погода России"

Рассматриваются подходы к автоматизации процессов съема, обработки и анализа данных ветромониторинга с сервера "Погода России", которые выполняются с помощью специально разработанного для этих целей программного обеспечения. Его функциональные возможности демонстрируются на примере данных по Республике Дагестан.

***Ключевые слова**: ветромониторинг, автоматизация выборки данных*

D. N. Kobzarenko

Automation for Sample Data of Wind Monitoring from the "Russia's Weather" Server

The questions related to the preliminary analysis and processing of wind monitoring data from the "Russia's Weather" server, with the help of specially developed software are considered.

***Keywords:** wind monitoring, automation for sample data*

Введение

Задача подсчета ресурсов ветровой энергии в целях определения территорий, наиболее перспективных для их промышленного использования, требует наличия исходных данных. Основными исходными данными для такого определения являются временные ряды распределения скоростей и направлений ветра в различных географических точках. Чем плотнее при этом данные покрывают изучаемую территорию, тем более надежными получаются результаты анализа. Для расчета удельной мощности и удельной энергии ветра необходимо также знать температуру окружающей среды [1].

Во многих публикациях, например, в работе [2], при анализе распределения ресурсов энергии солнца и ветра в качестве исходных данных берут данные из информационной базы *NASA Surface meteorology and Solar Energy*. Особенностью этой базы является то, что для любой географической точки предоставляются только среднегодовые и среднемесячные значения метеорологических параметров за весь период наблюдения с 1983 г. по настоящее время. Однако при этом в текущей базе данных получить информацию по конкретным годам, месяцам и дням не представляется возможным.

При построении общероссийских атласов с картами среднегодовых и среднемесячных значений параметров, связанных с возобновляемыми энергоресурсами, применение базы данных *NASA Surface meteorology and Solar Energy* вполне оправдано. Однако при выполнении работ по исследованию ветроэнергетического кадастра в региональном масштабе [3], которые предполагают анализ таких характеристик, как повторяемость скоростей и направлений ветра, максимальная скорость ветра и т. д., в качестве исходных данных необходимо использовать временные ряды с измерениями несколько раз в сутки.

Исходные данные по скоростям и направлениям ветра в виде временных рядов можно получить из архивов метеорологических станций. Такие архивы в цифровом виде можно найти и безвозмездно (в режиме свободного доступа) скачать на сервере "Погода России" [4] в сети Интернет. В разделе "Архив погоды" содержатся ссылки на архивы метеорологических данных почти 5000 метеостанций по всему миру, начиная с декабря 1998 г. Для каждой интересующей пользователя метеостанции можно получить текстовый файл (определенного формата) со всеми метео-

рологическими параметрами по временной выборке либо за весь имеющийся период.

Использование данных сервера "Погода России" для последующих расчетов и составления ветроэнергетического кадастра в региональном масштабе предполагает решение ряда задач, что требует использования методов статистической обработки данных и современных средств автоматизации этих процессов. Возникновение некоторых задач обусловлено тем обстоятельством, что во многих временных рядах существуют пробелы в измерениях различной длительности, а также тем, что иногда во временных рядах встречаются некорректные значения измеряемых параметров. Решение таких задач предполагает:

- определение метеостанций, имеющих отношение к исследуемому региону;
- получение текстовых файлов с данными этих метеостанций за весь временной период;
- предварительную обработку данных — выборку временных рядов интересующих пользователя параметров, исключение некорректных значений;
- определение временных периодов и наборов метеостанций с минимумом пропущенных значений;
- выборку данных в соответствии с предыдущим пунктом и перевод их в цифровую форму для выполнения дальнейших расчетов.

Для решения представленного перечня задач было разработано рассматриваемое в данной работе специализированное программное обеспечение, которое позволяет оперативно и точно выполнить работы по отбору данных ветромониторинга с сервера "Погода России".

Инженерия программного обеспечения

Рассмотрим аспекты, связанные с проектированием программного обеспечения средства автоматизации выборки данных ветромониторинга с сервера "Погода России". К ним относятся: структура базы данных, формат данных, организация работы с данными, функциональная и интерфейсная части программного обеспечения.

Структура базы данных. Для объединения данных ветромониторинга в единую форму, удобную для работы, разработана иерархическая структура базы данных (рис. 1).

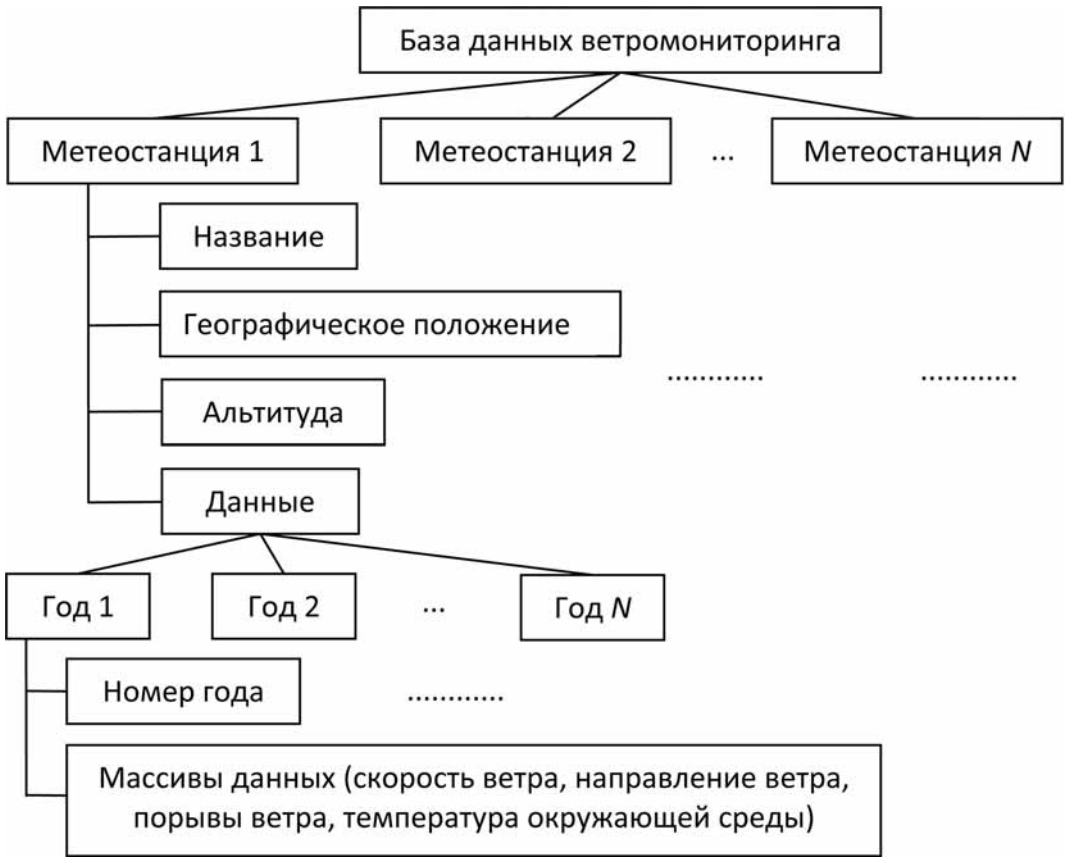


Рис. 1. Структура базы данных ветромониторинга

На самом верхнем уровне иерархии данные рассматривают как единое целое. Уровнем ниже идет разделение по метеостанциям. Для уровня "метеостанция" существуют поля, описыв ающие название станции, координаты географического положения, значение альтитуды и данные. Данные разделяются по годам, что представляет собой последний уровень иерархии. Для уровня иерархии "год" существуют поля, описывающие номер года и массивы данных, хранящие значения четырех интересующих параметров.

Формат данных базы данных. Для хранения данных на жестком диске в соответствии с представленной структурой базы данных разработан двоичный формат файла. Первые шесть байт в файле хранят *ASCII*-коды символов "*WindDB*", представляющих собой идентификатор файла. Следующие четыре байта являются целым числом *N* (число метеостанций). В остальной части файла хранятся *N* блоков данных для всех имеющихся в базе метеостанций.

Блок *i*-й метеостанции состоит из следующей последовательности данных: название станции в *ASCII*-строке (размер зависит от длины); широта (четыре байта — вещественное число); долгота (четыре байта — вещественное число); альтитуда (четыре байта — целое число); число лет (четыре байта — целое число). Далее следует *M* блоков данных, хранящих информацию по годам для *i*-й метеостанции.

Блок *j*-го года состоит из следующей последовательности данных: номер года (два байта — целое число), массив данных о скоростях ветра (5952 байт); массив данных о направлениях ветра (5952 байт); массив данных о порывах ветра (2976 байт); массив данных о температуре (2976 байт).

Длины массивов определены из следующих соображений. Для удобства расчетов доступа к данным длина каждого из 12 месяцев принята равной 31 день. Максимальное число измерений в день равно 8, через каждые 3 часа (на основе информации сервера "Погода России"). Итого $31 \times 12 \times 8 = 2976$ измерений в году. Для хранения значения скорости ветра используется два байта: старший байт — целая часть, младший — дробная. Для хранения значения направления ветра (диапазон 0...355) также используется два байта. Для хранения информации о порывах ветра с точностью до 1 м/с и значения температуры окружающей среды с точностью до 1 °С достаточно одного байта.

Управление базой данных. Программная реализация функций управления базой данных выполнена на основе объектно-ориентированного подхода. В среде программирования *Delphi* разработаны три управляющих класса.

Класс *TWindProc* управляет базой данных как единым целым. В свойствах класса содержится информация о числе метеостанций и временном диапазоне данных (годы). Возможности класса состоят из следующих функций: чтение БД из файла, запись БД в файл и импорт данных из группы файлов сервера "Погода России". Через векторное свойство класса *Stations[i]* осуществляется доступ к управлению данными для *i*-й метеостанции. Это свойство является экземпляром класса управления данными метеостанции — *TWindStation*.

В свойствах класса *TWindStation* содержится информация о параметрах метеостанции. Существует функция добавления нового календарного года к данным по текущей метеостанции. Через векторное свойство *Years[Number]* класса осуществляется доступ к управлению данными для *Number*-го календарного года. Это свойство является экземпляром класса управления данными календарного года — *TWindYear*.

В свойствах класса *TWindYear* содержится информация о номере года и о том, является ли год високосным. Через векторные свойства класса *Speeds[Month, Day, Num]*, *Directions[Month, Day, Num]*, *Gusts[Month, Day, Num]*, *Tempers[Month, Day, Num]* осуществляется доступ к данным по скоростям ветра, направлениям ветра, порывам ветра и температуре окружающей среды соответственно месяца *Month*, дня *Day*, измерения *Num*. В этом классе присутствуют также свойства и методы для выполнения статистического анализа данных в рамках текущего года.

В рамках **функциональной части ПО** разработан класс *TDiagram*, выполняющий графическое построение так называемых диаграмм полноты данных, которые более подробно описаны далее.

Интерфейсная часть ПО разработана с применением стандартных средств среды программирования *Delphi*. Предусмотрены возможности табличного и графического представления данных ветромониторинга и набор кнопок для управления программой.

Результаты работ программного обеспечения описаны далее применительно к территории Республики Дагестан.

Импорт данных с сервера "Погода России"

Для исследования распределения ресурсов энергии ветра в Республике Дагестан были выбраны десять метеостанций, находящихся на ее территории. К их числу относятся метеостанции "Ахты", "Буйнакск", "Дербент", "Кизляр", "Кочубей", "Махачкала-Уйташ", "Сергокала", "Хасавюрт", "Южно-Сухокумск", "о. Тюлений", а также еще девять метеостанций, расположенных на соседних территориях; "Артезиан", "Грозный", "Закаталы", "Кварели", "Комсомольский", "Лагодехи", "Рощино", "Телави", "Тианети". Для каждой метеостанции с сервера "Погода России" [4] был сгенерирован и скачан текстовый файл с данными за весь период наблюдения до конца 2012 г.

Исходные данные с сервера "Погода России" в виде группы текстовых файлов с помощью опции "Импорт" были импортированы в базу данных.

Просмотр данных и редактирование некорректных значений

После того как выполнен импорт данных, они сохраняются в формате иерархической базы данных и с ними можно осуществлять дальнейшие действия.

На следующем этапе выполняется просмотр всех данных по метеостанциям и годам в табличном виде (рис. 2) для выявления и исправления некорректных значений. Некорректными значениями скорости ветра в контексте данной работы считаем значения соответствующие (по оценке эксперта) явно нереальной скорости в данный период времени и, в частности, ее не реально большое значение по отношению к соседним значениям временного ряда. Анализ всех имеющихся данных по Республике Дагестан показал, что во всех случаях некорректных данных в значениях пропущена запятая (или точка). Например, число 40, которое соответствует в реальности числу 4,0. На рис. 2 показан пример некорректного значения скорости ветра — 83 м/с во временном ряде 2011 г. метеостанции "Махачкала". Логически можно предположить, что корректное значение будет 8,3 м/с, если связать в единое целое предыдущее значение 7 м/с и последующее 6 м/с.

Просмотр всех данных показал, что подобных некорректных значений может быть до десяти за календарный год, и во всех таких значениях между цифрами, скорее всего, пропущена запятая (или точка).

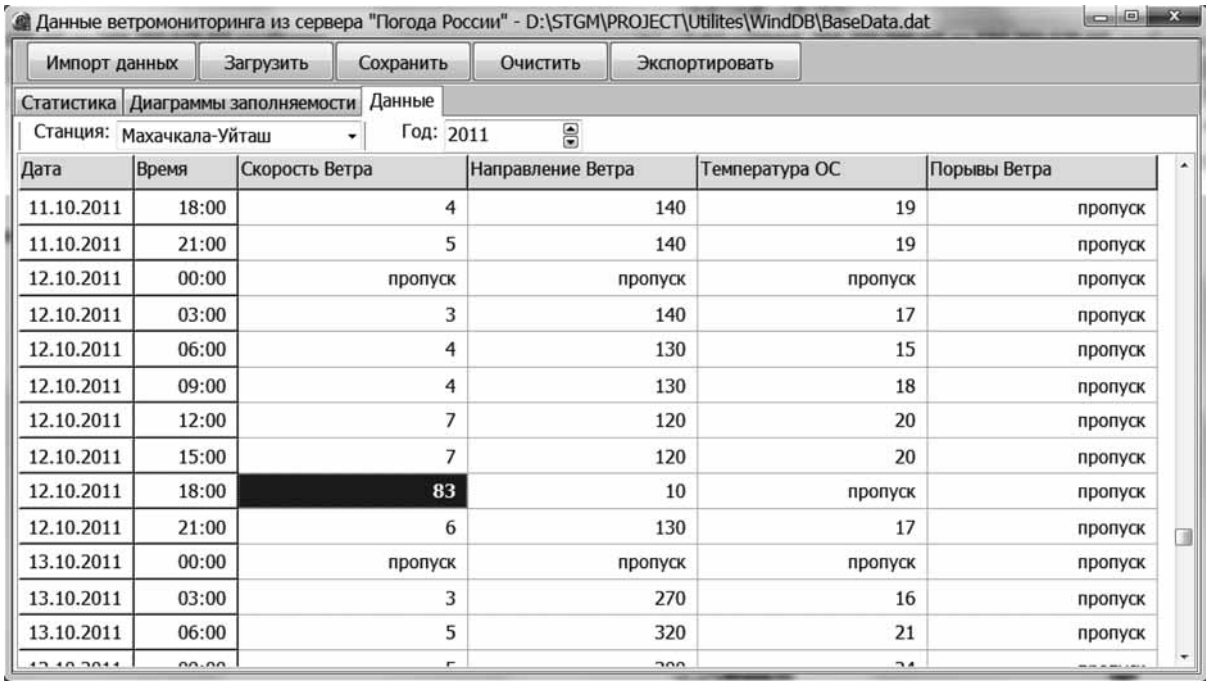
Так как просмотр всех данных трудоемкий процесс, было решено предоставить пользователю возможность корректировать значения в полуавтоматизированном режиме. Согласно шкале Бофорта ветер со скоростью выше 20 м/с является девятибалльным штормом (более 32 м/с — ураган в максимальные 12 баллов). Исходя из этого факта, определена формула логического выражения R, принимающего значение ИСТИНА в случае, когда текущее значение скорости ветра V_i попадает под подозрение как некорректное:

$$R(i) = (V_i > 20)\ \text{И}\ (V_i/V_{i-1} > 2)\ \text{И}\ (V_i/V_{i+1} > 2), \quad (1)$$

где V_{i-1} — предыдущее показание скорости ветра; V_{i+1} — последующее показание скорости ветра.

Алгоритм полуавтоматизированной корректировки значений следующий. Для всех значений скорости ветра в базе данных выполнить: если $R(i)$ = ИСТИНА, то вывести диалоговое окно, в котором указываются текущее, предыдущее и последующее значения скоростей ветра, и предлагается отредактировать текущее значение.

Право выбора — "редактировать" или "пропустить" — остается за пользователем.



Дата	Время	Скорость Ветра	Направление Ветра	Температура ОС	Порывы Ветра
11.10.2011	18:00	4	140	19	пропуск
11.10.2011	21:00	5	140	19	пропуск
12.10.2011	00:00	пропуск	пропуск	пропуск	пропуск
12.10.2011	03:00	3	140	17	пропуск
12.10.2011	06:00	4	130	15	пропуск
12.10.2011	09:00	4	130	18	пропуск
12.10.2011	12:00	7	120	20	пропуск
12.10.2011	15:00	7	120	20	пропуск
12.10.2011	18:00	83	10	пропуск	пропуск
12.10.2011	21:00	6	130	17	пропуск
13.10.2011	00:00	пропуск	пропуск	пропуск	пропуск
13.10.2011	03:00	3	270	16	пропуск
13.10.2011	06:00	5	320	21	пропуск

Рис. 2. Корректировка данных ветромониторинга

Определение временных периодов и наборов метеостанций с минимумом пропущенных значений

Чтобы было легко определить временные периоды и наборы метеостанций с минимумом пропущенных значений, в программном обеспечении был спроектирован и внедрен механизм построения так называемых диаграмм полноты данных. В диаграмме по горизонтали расположена временная ось, по вертикали — ряды, связанные с метеостанциями, цветом на диаграмме показана степень полноты данных в процентах. Полнота в 100 % означает, что в базе данных имеются все восемь значений за сутки.

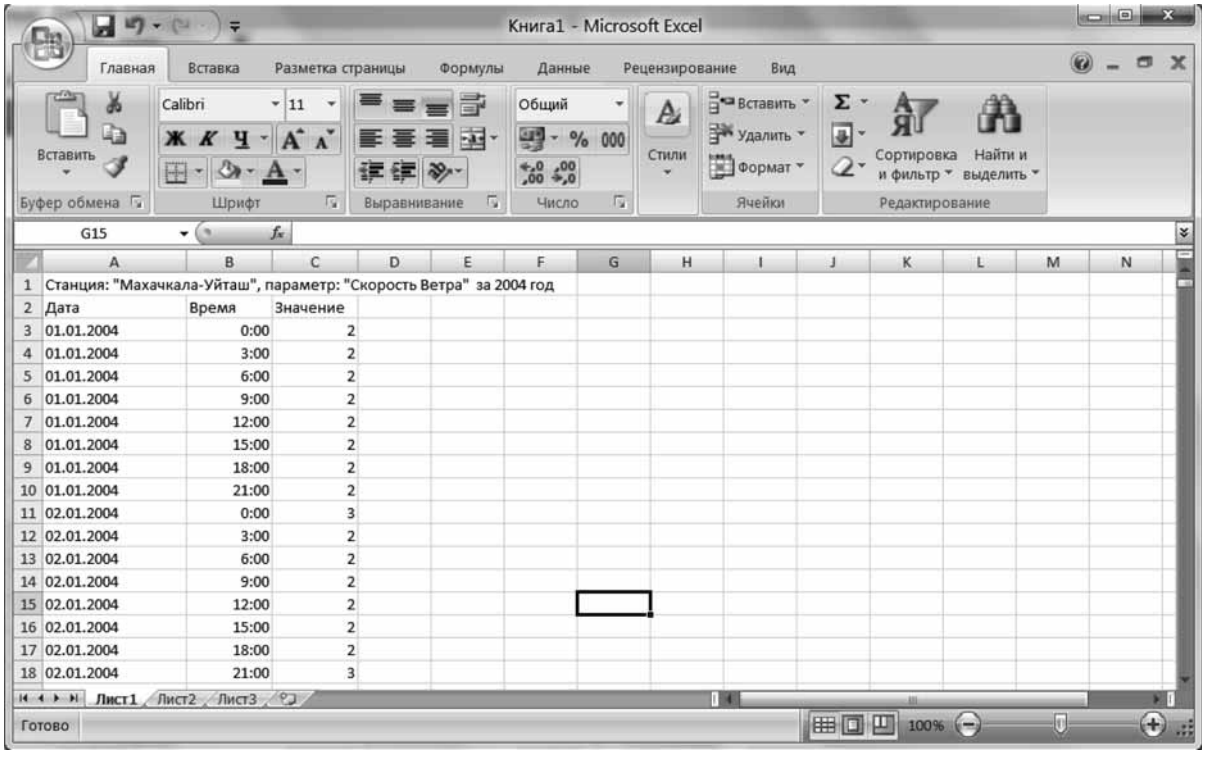
Диаграммы полноты данных являются очень информативными, что будет показано примерами далее. Они помогают быстро определить временную структуру наполненности базы данных по метеостанциям. Имеются три типа диаграмм: среднесуточная с выборкой по годам, среднегодовая с выборкой по годам и месяцам и диаграмма за весь период.

Среднесуточная диаграмма полноты данных по скоростям ветра на рис. 3 (см. вторую сторону обложки) за весь временной период иллюстрирует, что частота в восемь измерений в сутки имеет место только для пяти метеостанций: "Ахты", "Грозный", "Дербент", "Кочубей" и "Махачкала". Для остальных метеостанций характерно, что выполняются только четыре измерения в сутки. Поэтому при анализе суточного хода ветра наиболее полную картину могут дать данные только указанных пяти метеостанций.

Среднегодовая диаграмма полноты данных по скорости ветра на рис. 4 (см. вторую сторону обложки) за весь временной период показывает, что лучшую полноту имеют данные с метеостанций "Махачкала-Уйташ" и "Кочубей". Далее по полноте данных за весь имеющийся период можно выделить метеостанции "Ахты", "Дербент", "Телави" и "Закатала".

Самой информативной является диаграмма полноты данных с развернутой осью времени по всему временному периоду базы данных. Именно благодаря этой диаграмме можно выделить временные периоды и наборы метеостанций с минимумом пропущенных значений. На рис. 5—8 (см. вторую и третью стороны обложки) по частям показана диаграмма полноты базы данных за период с 1998 по 2012 г.

На рис. 5 (см. вторую сторону обложки) можно видеть, что с марта (03) 2000 г. начинается временной интервал, для которого в базе данных имеется относительно полная информация по всем 19 метеостанциям. Этот временной интервал заканчивается в июне (06) 2004 г., что видно на рис. 6 (см. третью сторону обложки). С июля (07) 2004 г. в базе данных имеются значения временных рядов только по двум метеостан-



	A	B	C
1	Станция: "Махачкала-Уйташ", параметр: "Скорость Ветра" за 2004 год		
2	Дата	Время	Значение
3	01.01.2004	0:00	2
4	01.01.2004	3:00	2
5	01.01.2004	6:00	2
6	01.01.2004	9:00	2
7	01.01.2004	12:00	2
8	01.01.2004	15:00	2
9	01.01.2004	18:00	2
10	01.01.2004	21:00	2
11	02.01.2004	0:00	3
12	02.01.2004	3:00	2
13	02.01.2004	6:00	2
14	02.01.2004	9:00	2
15	02.01.2004	12:00	2
16	02.01.2004	15:00	2
17	02.01.2004	18:00	2
18	02.01.2004	21:00	3

Рис. 10. Экспорт данных в формат Excel

циям — "Махачкала-Уйташ" и "Кочубей". Такая ситуация продолжается вплоть до 2009 г., образуя таким образом пятилетний пробел в данных.

Начиная с марта (03) 2009 г., можно выделить начало второго временного интервала, в котором имеется хорошая полнота значений временных рядов по шести метеостанциям — "Ахты", "Дербент", "Закатала", "Кочубей", "Махачкала-Уйташ" и "Телави" (рис. 7, см. третью сторону обложки). Этот временной интервал продолжался до конца 2012 г. (данные за 2013 г. пока не обработаны). В конце 2012 г. к шести метеостанциям добавляются данные еще по трем метеостанциям — "Грозный", "Кизляр" и "Комсомольский" (рис. 8, см. третью сторону обложки). Остается надеяться, что последний временной интервал будет продолжен в 2013 г., и в последующих годах процесс измерений метеорологических параметров с других метеостанций будет восстанавливаться.

На приведенных рисунках показаны диаграммы полноты значений по параметру "скорость ветра". Сопоставляя эти диаграммы с диаграммами полноты данных значений параметров "направление ветра" и "температура окружающей среды" (рис. 9, см. четвертую сторону обложки), несложно прийти к выводу, что они идентичны. Это означает, что пропуски в данных синхронны по всем трем параметрам. Что касается параметра "порывы ветра", то его значения в основном зафиксированы во временном интервале с марта 2000 г. по июнь 2004 г.

Экспорт данных

После предварительной обработки и корректировки данных ветромониторинга удается ответить на вопрос, наборы данных каких метеостанций и в какие временные периоды представляют интерес. Далее можно воспользоваться опцией "Экспорт данных" и экспортировать данные для любого года и метеостанции в формат Excel (рис. 10).

Заключение

На примере данных ветромониторинга по Республике Дагестан показано, что разработанное программное обеспечение для предварительной обработки и анализа данных ветромониторинга с сервера "Погода России" позволяет структурировать исходные данные, быстро скорректировать ошибки в значениях и проанализировать полноту данных во времени. На основе анализа полноты данных уточняются временные периоды и метеостанции, имеющие достаточную наполненность данными для выполнения работ по исследованию потенциала ресурсов энергии ветра в регионе.

Программное обеспечение не связано со спецификой конкретного региона и может быть использовано для любых наборов метеостанций с сервера "Погода России".

Список литературы

1. **Виссарионов В. Э.** Методы расчета ресурсов возобновляемых источников энергии. М.: МЭИ, 2009. 144 с.
2. **Попель О. С., Фрид С. Е., Коломиец Е. Г., Киселева С. В., Терехова Е. Н.** Атлас ресурсов солнечной энергии на территории России. М.: ОИВТ РАН, 2010. 84 с.
3. **Минин В. А., Дмитриев Г. С., Иванова Е. А., Морошкина Т. Н., Никифорова Г. В.** Ресурсы ветровой энергии Мурманской области и возможности их промышленного использования. URL: http://www.kolasc.net.ru/russian/sever06/ sever_9.pdf
4. **Сервер** "Погода России". URL: http://meteo.infospace.ru

ИНФОРМАЦИЯ

18—19 апреля 2014 г. в Москве пройдет
Пятнадцатая международная конференция в области обеспечения качества ПО
"Software Quality Assurance Days"

SQA Days — это платформа общения и обмена опытом для людей, вовлеченных в сферу тестирования ПО. Ведущие профессионалы смогут рассказать о своих достижениях, показать, как эффективно использовать инструменты, методики и методологии. Для начинающих — это отличный шанс приобрести новые полезные знакомства в профессиональной среде.

Конференция охватит широкий спектр профессиональных вопросов в области обеспечения качества, ключевыми из которых являются:

- методики и инструменты тестирования ПО;
- автоматизация тестирования ПО;
- подготовка, обучение и управление командами тестировщиков;
- процессы обеспечения качества в компании;
- управление тестированием и аутсорсинг;
- совершенствование процессов тестирования и инновации.

Подробности см. на сайте конференции: http://sqadays.com/article.sdf/sqadays/sqa_days14/about

УДК 004.912

В. А. Васенин, д-р физ. мат. наук, проф., зав. отделом, Институт проблем информационной безопасности МГУ, e-mail: vasenin@msu.ru,
С. А. Афонин, канд. физ.-мат. наук, вед. науч. сотр., НИИ механики МГУ имени М. В. Ломоносова, e-mail: serg@msu.ru,
Д. С. Панюшкин, аспирант МГТУ им. Н. Э. Баумана, вед. инженер-программист, ЗАО "Expert Solutions", email: dspan@yandex.ru

Модели распространения информации в социальных сетях

На основе анализа подходов к созданию путей распространения информации в социальных сетях и web-блогах предлагается модифицированная математическая модель, расширяющая возможности применения существующих методов решения этой задачи.

Ключевые слова: *социальные сети, пути распространения информации, модели информационных потоков, источники информации*

V. A. Vasenin, S. A. Afonin, D. S. Panushkin

Models of Information Dissemination in Social Networks

The modified mathematical model, which enhances the capabilities of existing methods of solving the problem of information dissemination in social networks, is proposed. The proposal is based on the analysis of approaches to the development of ways to disseminate information in social networks.

Keywords: *social networks, information dissemination ways, traffic models, data sources*

Введение

Процессы обмена информацией играют огромную роль в жизни человечества. Исследование законов и внутренней структуры этих процессов представляет собой одну из важнейших и интереснейших задач, которая может найти применение во многих областях практической деятельности. На протяжении последних веков процессы такого обмена постоянно ускорялись и принимали все большие масштабы. Такое положение дел было связано с научно-техническим прогрессом, в частности, с развитием средств коммуникации, а также с ростом населения планеты и объективными потребностями в эффективном обмене информацией. Вместе с перечисленными изменениями происходили и качественные изменения в структуре таких процессов. Например, всего несколько десятков лет назад главную роль в распространении новостей играли телевидение и радио. С появлением же метасети Интернет обмен информацией вышел на новый уровень. В него оказались вовлечены миллионы людей уже не в качестве пассивных слушателей, а в качестве активных субъектов обмена, которые могут как получать, так и передавать информацию. Процессы информационного обмена, определяющие эффективное развитие многих важных сфер жизни общества, стали предметом исследования. Однако исследование структуры процессов обмена информацией без привлечения современных методов и средств автоматизации не представляется возможным. Таким образом, актуальной стала задача исследования процессов распространения информации с помощью программных средств автоматизации.

В последнее десятилетие стал проявляться особый интерес не только к наблюдению за потоками информации в сети Интернет, но и к активному их формированию, а также к управлению такими потоками. Исследования на уровне подсчета статистических параметров процессов распространения информации в метасети не могут в должной мере ответить на вопросы, связанные с управлением ее потоками. Для этого требуется глубокое понимание строения структур, участвующих в процессе обмена информацией, на микро- и макроуровнях. Это в свою очередь приводит к необходимости моделирования и прогнозирования поведения информационных потоков.

Множество субъектов обмена информацией удобно представить в виде некоторой сети, между узлами которой могут передаваться сообщения. Тогда структура обмена будет отражаться с помощью интенсивности передачи сообщений между различными участками этой сети. Необходимо упомянуть, что в реальных сетях при этом может не происходить именно посылки/принятия сообщений. Например, узлы мо-

гут считавать информацию друг с друга, что эквивалентно посылкам сообщений от узлов-источников к узлам-читателям. Вследствие того что на настоящее время не существует достаточно развитых средств распознавания и, прежде всего, анализа звуковых и графических образов, представляется целесообразным автоматизировать лишь исследования путей распространения текстовой информации.

Следует отметить, что каждой отдельной подсети в сети Интернет соответствует свой характер обмена информацией. Например, по web-документам информация распространяется сравнительно медленно и может не покидать тематических ресурсов. А в сетях микроблогов короткие сообщения распространяются почти мгновенно и не концентрируются в узлах сети с определенной тематикой. По этой причине задача исследования распространения информации в Интернет как в единой сети является слишком общей и в практическом плане неразрешимой.

На настоящее время большинство людей имеют достаточно быстрый доступ к сети Интернет. В связи с этим широкое распространение получили личные публикации. Личные онлайн-дневники или блоги управляются простыми в использовании программными пакетами, которые позволяют одним щелчком мыши осуществить публикацию ежедневных записей. В отличие от способов распространения информации, не связанных с сетью Интернет, блоги обычно открыты для общего наблюдения, что позволяет свободно заниматься их исследованием.

Широкое распространение получили социальные сети, например, MySpace, Facebook, Flickr, Вконтакте, основное свойство которых — наличие окружения пользователя (друзей). Исходя из этого свойства, к социальным сетям можно отнести также сервисы обмена электронной почтой и сервисы голосовой связи, такие как Skype и Google Talk. Сообщения в социальных сетях делятся на личные и широковещательные. К первому типу относится личная переписка пользователей. Ко второму типу — объявления пользователя, доступные всем его друзьям. Отслеживание личной переписки технически затруднено и, как правило, незаконно. Однако широковещательные объявления пользователей в этом плане ничем не отличаются от записей в блогах. Очень часто широковещательные сообщения применяют для распространения вирусной рекламы или оказания информационного воздействия. На вирусную рекламу в настоящее время тратятся огромные денежные средства, а вопросами информационного воздействия общество интересовалось, пожалуй, с самых ранних этапов своего существования. По причинам, перечисленным выше, исследования в области социальных сетей представляются очень важными.

В последнее время широкую популярность приобрели так называемые микроблоги — сети распространения мгновенных сообщений, в частности, Twitter. Они представляют особый интерес для исследований ввиду большой скорости обмена сообщениями, а также их небольшой длины. В этих условиях, а также по причине относительно высокой скорости обмена, достаточно быстро накапливается большой статистический материал для исследований. Twitter служит для распространения новостей, позволяя узнавать их почти мгновенно из свидетельств очевидцев, минуя СМИ. В силу перечисленных причин исследование структуры распространения информации в микроблогах требует особого внимания. Отметим характерные вопросы изучения структуры социальных сетей и блогов, к числу которых относятся следующие:

- как изменяется популярность сообщений со временем;
- существуют ли шаблоны распространения информации;
- насколько широко и с какой скоростью распространяется информация;
- какое влияние оказывают блоги друг на друга;
- на каких пользователей сети нужно оказать воздействие, чтобы вызвать "цепную реакцию"?

Не следует забывать, что обмен информацией происходит, прежде всего, между людьми, и уже во вторую очередь — между узлами компьютерной (или какой-либо другой рассматриваемой) сети. Поэтому наблюдать и исследовать распространение информации можно только по косвенным признакам. Например, в большинстве случаев нельзя отследить сам факт передачи сообщения между конкретными узлами сети. Можно наблюдать лишь появление нового сообщения в одном из узлов.

Целью исследования, результаты которого представлены в данной статье, является разработка методов и программных средств, автоматизирующих исследование процессов распространения информации в некоторой абстрактной сети, между узлами которой идет обмен сообщениями, наблюдаемый только по появлению различных сообщений в ее узлах, а также применение этих методов к блогам, микроблогам и социальным сетям.

Результаты применения таких методов и инструментальных средств к реальным сетям могут быть актуальны в следующих практических задачах:

- прогнозирование распространения информации по сети;
- выявление связей между узлами сети;
- поиск источников конкретной информации;
- разбиение сети на группы, внутри которых обмен наиболее интенсивен;
- нахождение узлов сети из окружения пользователя, наиболее подверженных информационному воздействию или вирусной рекламе;
- определение такого минимального множества узлов, появление информации в котором приведет к ее распространению на всю сеть;
- выявление наиболее активных источников распространения информации (либо наиболее авторитетных авторов) по косвенным признакам.

Исследования социальных сетей непосредственным образом связаны с вопросами социологии, маркетинга, безопасности, а также вопросами в сфере политики. Они позволяют оптимизировать решение таких задач, как распространение рекламы, поиск источников информации экстремистской направленности, PR, распространение новостей, прогнозирование социальных процессов и др.

Для того чтобы более полно отразить особенности рассматриваемой задачи, следует привести результаты некоторых статистических исследований.

В работе [1] было проведено исследование сети блогов на основе анализа перекрестных ссылок. За август—сентябрь 2005 г. было проиндексировано 21 млн блогов, из них лишь 45 тыс. оказались активными. Было проиндексировано свыше 2,2 млн сообщений, на которые приходилось всего 200 тыс. ссылок, в 98 % случаев сообщения оказались изолированными. Отсюда следует, что методы исследования, основанные лишь на анализе перекрестных ссылок, упускают большую часть наблюдаемого материала.

Интерес представляют результаты исследования сети Flickr, представленные в работе [2]. За период с декабря 2006 г. по май 2007 г. было проиндексировано 11 млн фотографий от 2,5 млн пользователей. Основные результаты этого исследования:

- значительное (до нескольких месяцев) время передачи сообщения между друзьями;
- высокий уровень лояльности — более 50 % всех пользователей узнают о новых фотографиях от своих друзей;
- низкая скорость распространения — фотография становится популярной за один—два года;
- более 80 % пользователей, отметивших фотографию, находятся на расстоянии ⩽2* от ее автора.

Сопутствующие подходы

Существует достаточно большое число источников, содержащих результаты исследований о распространении информации, которые используют математические модели, разработанные для анализа других сфер жизнедеятельности человека, в первую очередь, в области эпидемиологии, маркетинга и социологии. Все представленные далее подходы моделируют процесс распространения сообщений как случайный процесс с дискретным временем. На очередном шаге его реализации каждый узел сети может принять сообщение при условии наличия соседей по сети, уже принявших это сообщение.

Эпидемиология. Многие проведенные ранее исследования были основаны на аналогии распространения информации через сеть и распространении заболеваний. Они привносят огромный опыт исследования механизмов эпидемиологии в вопросы распространения информации [3].

К классическим моделям распространения заболеваний относятся так называемые SIR-, SIS- и SIRS-модели. В этих моделях индивид последовательно проходит следующие стадии:

- S — восприимчив (*susceptible*);
- I — инфицирован (*infected*);
- R — выздоровел/удален (*recovered*/*removed*).

В рамках данных моделей восприимчивый индивид u, контактируя с инфицированным другом v с некоторой вероятностью p может быть инфицирован, затем некоторое время $t_{infected}$ пребывать в болезни, с возможностью передать ее другим индивидам. Далее, в зависимости от модели, индивид может либо вновь стать восприимчивым (модель SIS), либо приобрести иммунитет (модели SIR, SIRS). Две последние модели отличаются тем, что в модели SIRS после прохождения периода иммунитета $t_{immunity}$ индивид вновь становится восприимчивым.

В работе [4] рассматривается особая модель SIR, в которой индивид после выздоровления получает иммунитет к болезни, но остается восприимчив к любым ее модификациям. В сферу блогов эту модель можно перенести следующим образом:

1) блоггер, прочитав сообщение, копирует его себе;

2) по прошествии времени блоггер удаляет сообщение;

3) еще через некоторое время блоггер вновь возвращается к старой теме, претерпевшей изменения.

Такому поведению отвечают темы, популярность которых включает периоды вспышек и затишья.

В работах [5, 6] были предприняты попытки вычислить такую минимальную вероятность передачи болезни, что от одного индивида заразился бы фиксированный процент пользователей сети. В ранних работах по эпидемиологии было показано, что в условиях SIS-модели, если граф контактов обладает свойством $P(deg(v) = k) \sim k^{\alpha}$, где $P(deg(v) = k)$ обозначает вероятность того, что степень вершины $v = k$, $k \in \mathbb{N}$, а α — отрицательная константа, то обязательно будет заражен фиксированный процент популяции. Однако в блогосфере большинство сообщений распространяется не создавая эпидемии, что делает SIS-модели несостоятельными. Это объясняется высоким локальным коэффициентом кластеризации сети блогов (две вершины, соседние для данной, с большой вероятностью тоже окажутся соседними), а также убыванием вероятности заражения при удалении от источника эпидемии.

В работах [6, 7] модели SIR и SIS были различным образом уточнены для применения к таким сетям, как блогосфера и электронная почта.

Социология. Распространению информации в сети Интернет можно также поставить в соответствие распространение инноваций, которое изучает социология. Как инновацию можно представить некую полезную информацию, распространяемую по социальной сети. Пользователь может либо принять инновацию, либо не принять.

В социологии рассматривают две представленные далее основные модели распространения инноваций.

- Каскадная модель [8]. Всякий раз, когда соседний узел v принимает инновацию, данный узел u с некоторой вероятностью $p_{v,u}$ тоже ее принимает. Изначально инновацию принимает непустое множество узлов. Затем инновацию принимает некоторое, возможно пустое, множество узлов, и так далее, пока новые принятия не перестанут происходить. Следует отметить, что если узел u не принял инновацию от узла v на следующий же шаг после принятия инновации узлом v, то v больше никогда не передаст ее u. В этом заключается

* В этой социальной сети есть понятие "дружба" и расстояние измеряется в числе переходов между друзьями. Расстояние ⩽2 означает, что это либо друзья автора, либо друзья друзей.

отличие этих моделей от SIS, и SIRS-моделей. Различают независимую каскадную модель — Independent Cascade Model [8], в рамках которой каждый узел u из соседей узла v принимает от него инновацию независимо от других соседей узла v, и обобщенную каскадную модель — General Cascade Model [9], в которой вводится зависимость между соседями узла v.

- Пороговая модель [10]. Каждый узел u выбирает порог $t_u \in [0, 1]$. Каждой вершине v соседней с u приписана степень влияния $\omega_{v,\,u}$ (с условием, что сумма $\omega_{v,\,u}$ по v не превосходит единицы). Множество $A(u)$ состоит из соседних с u вершин v, принявших инновацию. Узел u принимает инновацию, если и только если $t_u \leqslant \sum_{v \in A(u)} \omega_{v,\,u}$.

Отметим, что в обоих представленных подходах сеть моделируется ориентированным графом со взвешенными ребрами.

Теория игр и маркетинг. Следует рассмотреть такую ситуацию: блоггер начинает совместное обсуждение интересной темы внутри своего круга общения, что приносит ему удовольствие или повышает статус. В работах [11, 12] была предложена модель, позволяющая описать эту ситуацию.

За каждый шаг по времени каждый узел социальной сети выбирает тип 0 или 1, где 1 интерпретируется как принятие сообщения. У каждого узла есть свой индивидуальный критерий определения ценности сообщения, однако при этом кроме прибыли от собственного принятия сообщения узел получает прибыль от каждого соседнего узла одного с ним типа. Критерием принятия или непринятия сообщения является максимизация прибыли, зависящей от этого поступка.

В работах [9, 13, 14] также представлена модель, восходящая к маркетингу. Она позволяет оценить вероятность того, что пользователь примет сообщение ("купит"), при условии, что определенный процент его окружения уже принял это сообщение. В этих работах решается задача выбора такого множества пользователей, что воздействие на этих пользователей приведет к максимальному охвату сети.

Постановка задачи

В настоящей работе под *процессом распространения информации* понимается обмен *сообщениями* между некоторым конечным множеством субъектов, его осуществляющих, которые далее именуют источниками информации. Каждое сообщение имеет свою *тему*. Темы являются классами эквивалентности на множестве сообщений.

О процессе обмена делают следующие предположения:

- обмен сообщениями проводится в дискретном времени;
- источники могут создавать свои сообщения, читать чужие сообщения и копировать их;
- источник не может получить дважды сообщение, относящееся к одной и той же теме;
- все сообщения и источники наблюдаемы (модель замкнутого мира);
- при чтении или копировании сообщения источником B из источника A разница во времени между появлением сообщения в A и появлением его в B строго больше нуля, т. е. не происходит "мгновенной" передачи сообщений;
- сообщения, относящиеся к одной и той же теме и появившиеся в разных источниках с небольшой разницей во времени, могут считаться одновременными;
- источники не могут удалять сообщения;
- о каждой теме сообщения известно лишь ее содержание и время появления относящихся к ней сообщений в каждом из источников, в которых они наблюдались.

В качестве ребра *графа контактов* между источниками принимается не формальное отношение "дружбы" между пользователями социальной сети, а возможность перехода сообщений от одного источника к другому.

Под *схемой распространения информации* понимается оценка интенсивности передачи сообщений между каждыми двумя источниками. Под *схемой распространения сообщений* понимается оценка вероятности передачи сообщения, относящегося к каждой конкретной теме, между любыми двумя источниками, в которых эта тема наблюдалась.

Таким образом, основной задачей в рамках изложенного выше является разработка методов и программных средств построения и анализ схем распространения информации и схем распространения сообщений, наиболее соответствующих действительности.

Математическая модель

К основным положениям, характеризующим модель распространения информации, относятся перечисленные далее.

1. *Совокупность источников информации* отождествляется с множеством $V = \{v_1, ..., v_{NV}\} \subset \mathbb{N}$, т. е. каждому источнику присваивается порядковый номер.

2. Темы сообщений разбивают все множество сообщения на классы. *Совокупность тем сообщений* представляется в виде множества $M = \{m_1, ..., m_{NM}\} \subset \mathbb{N}$, где m_i является номером темы.

3. Для определения процесса передачи сообщений между источниками вводится функция τ: $M \times V \to T$, $\tau(m, v) = t$, где $T = \{0, ..., t_{\max}\}$ — это *время* в рассматриваемом процессе распространения сообщений, а $t \in T$ — переменная, отражающая *время появления сообщения* с темой m в вершине v; $t_{\max}$ обозначает момент окончания наблюдения за сетью.

4. Таким образом, *наблюдаемое сообщение* блоггера представляет собой тройку $a_i = (m, v, t)$. *Картина наблюдения* $A = \{a_1, ..., a_{NA}\}$ определяется как множество таких троек.

5. *Схему распространения сообщений* можно представить в виде помеченного ориентированного мультиграфа $G = (V, E, \rho(\cdot))$, вершинами которого являются источники $v \in V$, множество ребер $E = V \times V \times M$, а $\rho: E \to [0, 1]$, $\rho(u, v, m) = p_{u,\,v}(m)$ — функция меток, харак-

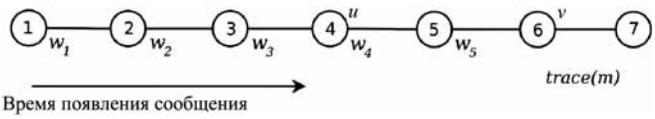


Рис. 1

теризующая оценку $p_{u,\,v}(m)$ вероятности перехода сообщения с темой m от источника u к источнику v.

6. *Схемой распространения информации* будет называться граф

$$G_{\min} = (V,\, E_{\min},\, p_1(\cdot),\, \ldots\, p_{N_p}(\cdot),\, q_1(\cdot),\, \ldots\, q_{N_q}(\cdot)),$$

$$p_i(\cdot): E_{\min} \to [0,\, 1],\; i = 1,\, \ldots,\, N_p,$$

$$q_i(\cdot): V \to [0,\, 1],\; i = 1,\, \ldots,\, N_q,$$

где $E_{\min} \subset V \times V$ — по возможности минимальное множество ребер, по которым могли бы пройти сообщения, создав ровно ту же картину наблюдения A;

$p_i(u,\, v)$, $i = 1,\, \ldots,\, N_p$ — оценки параметров, характеризующих вероятность перехода сообщений от u к v;

$q_i(u)$, $i = 1,\, \ldots,\, N_q$ — оценки параметров, характеризующих вероятность перехода сообщений в узел u независимо от конкретного соседнего узла.

В принятой терминологии основной задачей является построение алгоритмов нахождения наиболее соответствующей действительности схемы распространения сообщений G и схемы распространения информации $G_{\min}$ на основе картины наблюдения A, а также анализ точности этих алгоритмов и выбор критерия остановки (в случае итерационного алгоритма).

Следует отметить, что схема распространения информации служит для прогнозирования будущего поведения сети, а схема распространения сообщений — для выявления путей уже наблюдаемых сообщений.

Обзор известных результатов

Наиболее значимым результатом в методологии, близкой к поставленной в настоящей работе задаче, является работа [15]. В своей работе авторы вводят следующую модель сети, являющуюся производной от SIR-модели и независимой каскадной модели:

- сеть представляет собой полный ориентированный граф $G = (V,\, E)$, $E = V \times V$;
- ребра имеют метки $r_{u,\,v}$ — вероятность прочтения блоггером v сообщений у блоггера u в течение одного дня и $k_{u,\,v}$ — вероятность копирования блоггером v сообщения у блоггера u при условии его прочтения;
- для каждых двух вершин u, v и темы m вводится вероятность $p_{u,\,v}(m)$ перехода сообщения с темой m от блоггера u к блоггеру v.

Для нахождения схемы распространения информации строится следующий двухэтапный итерационный процесс.

Этап 1. Вычисление вероятностей $p_{u,\,v}(m)$. Для каждой темы $m \in M$ строится последовательность вершин $trace(m) \subset V$, в которых побывали сообщения данной темы (рис. 1). Затем последовательность $trace(m)$ сортируется по возрастанию времени обнаружения.

Для каждой вершины последовательности $v \in trace(m)$ и находящейся раньше нее в этой последовательности вершины u строится вероятность

$$p_{u,\,v}(m) = \frac{r_{u,\,v}(1 - r_{u,\,v})^{\delta_{u,\,v}(m)} k_{u,\,v}}{\sum\limits_{w < v} r_{w,\,v}(1 - r_{w,\,v})^{\delta_{w,\,v}(m)} k_{w,\,v}},$$

где $\delta_{u,\,v}(m)$ — время в днях между появлением сообщения в u и появлением в v.

Выражение $r_{u,\,v}(1 - r_{u,\,v})^{\delta_{u,\,v}(m)} = Pr_{geom}[x = \delta_{u,\,v}(m)]$, где Pr_{geom} — геометрическое распределение, означающее вероятность того, что блоггер v не читал сообщения блоггера u в течение $\delta_{u,\,v}(m)$ дней, а затем прочитал данное сообщение. Таким образом, домножение на $k_{u,\,v}$ дает безусловную вероятность копирования сообщения m. Выражение в знаменателе используется для нормировки вероятностей, так как обмен информации между блоггерами может проходить вне рамок сети, и тогда сумма вероятностей будет меньше единицы.

Этап 2. Обновление параметров $r_{u,\,v}$ и $k_{u,\,v}$. Для каждых двух фиксированных вершин u и v строится множество S_1 таких тем сообщений m, что m *сначала* наблюдалась в вершине u, а после этого в вершине v. Для них же строится множество S_2 таких тем сообщений m, что тема m наблюдалась в вершине u, но никогда не наблюдалась в вершине v.

Теперь параметры $r_{u,\,v}$ и $k_{u,\,v}$ последовательно вычисляют по следующим формулам:

$$r_{u,\,v} = \frac{\sum\limits_{m \in S_1} p_{u,\,v}(m)}{\sum\limits_{m \in S_1} p_{u,v}(m)\delta_{u,\,v}(m)},$$

$$k_{u,\,v} = \frac{\sum\limits_{m \in S_1} p_{u,\,v}(m)}{\sum\limits_{m \in S_1 \cup S_2} Pr_{geom}[r_{u,\,v} \leqslant \delta_{u,\,v}(m)]}.$$

В последней формуле, если тема m не достигла вершины v, то значение $\delta_{u,\,v}(m)$ считается равным разности момента окончания наблюдения за сетью $t_{\max}$ и момента появления сообщения m в вершине u. Выражение $Pr_{geom}[a \leqslant b] = 1 - (1 - a)^b$ — вероятность того, что геометрическое распределение с параметром a примет значение, не превосходящее константы b. При вычислении $k_{u,\,v}$ оно оценивает вероятность того, что сообщение m из вершины u будет прочитано вершиной v не более чем за $\delta_{u,\,v}(m)$ дней. В множество $S_1 \cup S_2$ входят все сообщения, которые u не мог прочитать из v, а для таких сообщений определяется вероятность прочтения их блоггером v у блоггера u. Таким образом, параметр $k_{u,\,v}$ определяется как отношение ожидаемого числа тем, скопированных из u в v, к ожидаемому числу тем, прочитанных блоггером v из u.

Шаг основного алгоритма представляет собой последовательное выполнение этапа вычисления вероятностей и этапа обновления параметров. Начальные значения параметров принимают равными произвольным константам из интервала [0, 1].

В работе [15] утверждается, что на их тестовых данных за 5—6 итераций изменение векторов $r_{u,\,v}$ и $k_{u,\,v}$ в течение одной итерации алгоритма уменьшается до 1 % в норме L_2*.

Методы решения

В настоящем разделе проанализированы достоинства, недостатки и рамки применения предложенной в работе [15] модели распространения информации. Представлено расширение (модификация) этой модели, устраняющее отмеченные недостатки, а также алгоритмы работы с расширенной моделью.

Анализ рамок применения существующей модели

Модель, предложенная в работе [8], позволяет достаточно точно прогнозировать распространение "обычных" сообщений, т. е. интересных только для пользователя, создавшего новую тему, и ближайшего к нему окружения в сети. Однако в реальных социальных сетях распространяются также сообщения, интересные большому числу пользователей. К этому типу можно отнести, например, новости и "вирусные" сообщения. При прочтении такого сообщения пользователю не важно, откуда он получил эту информацию, а вероятность копирования определяется лишь самой темой сообщения. Заметим, что именно распространение сообщений-новостей представляет наибольший интерес. Также существующая модель не учитывает кластерную структуру социальной сети, в которой критерии копирования сообщений-новостей изменяются при переходе новости в каждый следующий кластер сети.

Тестирование алгоритма, представленного в работе [15], на синтетических данных показало, что он восстанавливает большинство ребер графа, по которому строился тест. Вместе с тем алгоритм выдает также и большое число несуществующих ребер графа. Лишние ребра удается отделить на основе анализа вероятностей чтения и копирования, которые им приписаны, сравнивая вероятности ребра со средними вероятностями по сети или по участку сети. Однако отметим, что это возможно только на модельных данных, в которых вероятность копирования одинакова по всему графу. В реальной же ситуации есть возможность потерять ребра, по которым пришли сообщения. Следует заметить, что реальный человек не может читать сразу большое число источников. Отсюда все-таки следует необходимость провести удаление ребер.

Существующий алгоритм при определении вероятностных параметров чтения и копирования работает с полным графом. Вычислительную сложность одной его итерации удается понизить, удалив из графа ребра с вероятностями, близкими к нулю. Тогда сложность итерации будет линейно зависеть от средней степени вершины графа. Однако удаление ребер, когда их вероятности еще не стабилизировались, может привести к появлению ошибки в окончательном результате. Таким образом, появляется необходимость определения минимального множества ребер графа, согласованного с картиной наблюдения сообщений.

Построение расширенной модели

Расширение существующей модели основано на изложенных ниже предположениях.

- Для каждого источника информации множество авторов, сообщения которых он копирует, должно быть по возможности минимальным. Данное предположение основано сразу на двух фактах. Во-первых, человек не способен воспринимать одновременно большое число объектов. Во-вторых, при минимизации множества авторов, в него войдут преимущественно те авторы, набор сообщений которых в наибольшей степени совпадает с набором сообщений наблюдаемого источника. Этот факт иллюстрирует "жадный" алгоритм построения минимального покрытия множества системой подмножеств.
- Сеть имеет кластерную структуру. В каждом кластере существуют свои критерии копирования сообщений.
- Существуют сообщения, для которых критерий копирования пользователем не зависит от источника сообщения.
- Поведение человека во многом совпадает с поведением его окружения. Это означает, что в случае построения графа отношений между источниками информации близкие вершины будут копировать сообщения преимущественно у одних и тех же авторов.

Описание модели

- Сеть представляет собой ориентированный граф $G_{\min} = (V, E_{\min})$, $E_{\min} \subset V \times V$, обладающий минимальным множеством ребер, которое сохраняет возможным переход сообщений между пользователями согласно картине наблюдения A.
- Сообщения сети делятся на два типа: *обычные* и *новостные*. Для каждой темы m к первому типу относятся сообщения, располагающиеся в вершинах v, удаленных от вершины u, создавшей первое сообщение данной темы, на расстояние $s_{u,\,v} \leqslant s_{\max}$, где $s_{u,\,v}$ — минимальная длина пути от u до v в графе $G_{\min}$, а $s_{\min}$ — некоторая константа. Соответственно, ко второму типу относятся остальные сообщения.
- В графе $G_{\min}$ выделяют кластеры вершин. При переходе сообщения из кластера F в кластер H первоисточник сообщения $u \in F$ отождествляется со всем кластером F.
- Распространение конкретных тем характеризуется *схемой распространения сообщений* $G = (V, E, p(\cdot))$, где $p_{u,\,v}(m)$ — вероятность перехода темы m из u в v.
- Интенсивность потока *обычных* сообщений между двумя вершинами u и v по-прежнему отражается вероятностями чтения $r_{u,\,v}$ и копирования $k_{u,\,v}$, но вероятность копирования конкретного сообщения

* L_2 — корень из суммы квадратов (матанализ).

уже не определяется независимо от близких к v по графу $G_{\min}$ вершин.

- Интенсивность потока *новостных* сообщений между двумя вершинами u и v отражается вероятностью чтения $r_{u,v}$ и *толерантностью* k_u — вероятностью копирования новостного сообщения при условии его прочтения из любого источника.
- Для отражения зависимости между двумя вершинами u и v вводится функция $c: V \times V \to [0, 1]$, $c_{u,v} = c_{v,u}$, определяемая на основе графа $G_{\min}$ и участвующая в вычислении вероятности перехода конкретного сообщения.

Алгоритм работы с моделью

Алгоритм приведения модели в соответствие с картиной наблюдения также является производным от алгоритма, представленного в работе [15], и имеет итерационную структуру. Начальные данные полагают равными некоторым константам и впоследствии уточняются. Все вероятности $p_{u,v}(m)$ следует положить равными константе в случае появления m сначала в u, потом в v, и нулю в противном случае. Каждая итерация делится на шесть представленных далее этапов.

Этап 1. Построение графа $G_{\min}$. Для каждой вершины v строится множество вершин $B_v \subset V$, в которых сообщения из v наблюдались раньше, чем появились в v. Таким образом, B_v — множество всех возможных источников сообщений для v. Теперь каждая вершина $u \in B_v$ представляется в виде множества своих сообщений $u|_v = \{m_u^1, ..., m_u^{N_u}\}$, которые впоследствии наблюдались в v. Каждому сообщению из u ставится в соответствие его вероятность перехода $p_{u,v}$. Далее строится множество авторов $D_v \subset B_v$, имеющее минимальную мощность и обладающее свойством покрытия:

$$\bigcup_{u \in D_v} \left(\bigcup_{m_u \in u} m_u \right) = \bigcup_{m \in v} m_v.$$

Более подробно вопрос о построении такого множества на основе вероятностей $p_{u,v}$ рассматривается далее. И наконец, для всех вершин $u \in D_v$ в граф $G_{\min}$ добавляются ребра $u \to v$.

Этап 2. Кластеризация графа $G_{\min}$. Граф $G_{\min}$ разбивается на *кластеры* — группы вершин $F \subset G_{\min}$, которые определяются следующим свойством: любая вершина $v \in F$ имеет много больше ребер внутри группы F, чем во множестве $G_{\min} \backslash F$. В действительности такие кластеры соответствуют частично изолированным сообществам пользователей, внутри которых информационный обмен идет гораздо интенсивнее, чем с внешней сетью. Подробнее вопрос кластеризации рассмотрен в следующем подразделе.

Этап 3. Вычисление расстояний между вершинами $s_{u,v}$. Для определения типа сообщений строится $s_{u,v}$ — функция расстояний между вершинами по графу $G_{\min}$. Нет необходимости строить эту функцию для каждых двух вершин. Нужно лишь ответить на вопрос $s_{u,v} \leqslant s_{\min}$? Также, исходя из предположения, что при переходе сообщения из некоторого кластера F в некоторый кластер H первоисточник сообщения $u \in F$ отождествляется со всем кластером F, функцию $s_{u,v}$ можно строить отдельно для каждого кластера F. Областью определения функции $s(\cdot)|_F$ будет множество вершин F, определяемое как объединение F с множеством соседних для F вершин.

Исходя из значения $s_{\min}$ и размера кластера F можно поступить следующими двумя способами.

1. Провести для каждой вершины $u \in F$ подобие алгоритма "Волна". В рамках данного алгоритма ребра хранятся в индексированной куче. Строится последовательность множеств E^1, $E^1 \cup E^2$, ..., $E^1 \cup ... \cup E^{s_{\min}}$, где E^i — множества пар вершин (u, v) с расстоянием $s_{u,v} = i$. Каждое следующее множество строится по приведенной ниже формуле:

$$E^{i+1} = \{(u, v) \in F \times F : \exists (u, w) \in E^i \ \& \ \exists (w, v) \in E^1\}.$$

Ребра $(u, v) \in E^{i+1}$, уже содержащиеся в $E^1 \cup ... \cup E^i$, отбрасывают при объединении. Неформально построение следующего элемента последовательности похоже на операцию соединения отношений реляционной алгебры. Этот алгоритм применим при малых значениях $s_{\min}$ и больших размерах кластера F.

2. При больших значениях $s_{\min}$ проще оказывается провести поиск расстояний между всеми вершинами множества F по алгоритму Флойда — Уоршелла.

Для кластера $F = \{v_1, ..., v_{NF}\}$ строится матрица смежности $\mathbf{M}_{NF \times NF}$, заполняемая единицами в случае наличия ребра и ∞ в случае его отсутствия. Далее матрица заполняется числами по следующему правилу: для всех h, последовательно принимающих значения от 1 до N, вычисляется $M_{i,j} = \min(M_{i,j}; M_{i,h} + M_{h,j})$, $i = 1, ..., NF$, $j = 1, ..., NF$. По завершении процесса $M_{i,j}$ становится равным расстоянию от v_i до v_j. Этот алгоритм имеет вычислительную сложность $O(NF^3)$. При небольших размерах кластера такая сложность является допустимой.

Этап 4. Построение функции зависимости двух вершин $c_{u,v}$. Для каждой вершины u строится множество Env_u вершин графа $G_{\min}$, из которых исходят ребра, направленные к u. Далее для каждой пары вершин u и v определяется мера зависимости $c_{u,v}$:

$$c_{u,v} = \frac{card(\cap_{w \in Env_u \cap Env_v} w|_u)}{card(\cap_{w \in Env_u} w|_u)},$$

где *card* — мощность множества.

В числителе находится оценка числа сообщений, скопированных вершиной u у тех же авторов, у которых копирует v (рис. 2). В знаменателе стоит оценка числа всех скопированных u сообщений. Данная фор-

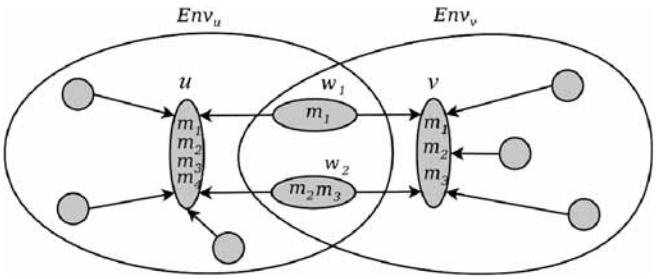


Рис. 2

мула вычисления похожести обусловлена тем, что просто близкие по графу $G_{\min}$ блоггеры могут иметь разные интересы. Очевидно, что автор и его читатель совершенно не обязаны быть похожими. Однако люди, копирующие сообщения у одних и тех же авторов, как правило, имеют общий круг интересов.

Вычисление $c_{u,\,v}$ следует проводить только для вершин, расстояние $s_{u,\,v}$ между которыми не превосходит 2. Для остальных пар вершин $c_{u,\,v} = 0$.

Этап 5. Вычисление вероятностей $p_{u,v}(m)$. Как и в предыдущей модели, для каждой темы m строится след сообщения — последовательность вершин $\mathrm{trace}(m) \subset V$, в которых побывали сообщения данной темы. Затем вершины этой последовательности разбиваются по кластерам, к которым они принадлежат. Дальнейшая обработка сообщения происходит отдельно для каждого кластера F (рис. 3).

Обработка следа сообщения проводится на вершинах $\overline{F}$ расширения кластера посредством соседних вершин. Последовательность $trace(m)|_{\overline{F}}$ сортируется в порядке возрастания времени обнаружения сообщений. Самая первая вершина y называется *источником*. Для всех вершин $v \in trace(m)|_{\overline{F}}$ считается определенное ранее расстояние $s_{y,\,v}$. Для вершин, удаленных от источника на расстояние $s_{\min}$ тема m обозначается как *новостная*, а для остальных — как *обычная*.

Вычисление вероятностей копирования обычной темы из u в v проводится по следующей формуле:

$$p_{u,\,v}(m) = \frac{\alpha r_{u,\,v}(1 - r_{u,\,v})^{\delta_{u,\,v}(m)} k_{u,\,v} + (1-\alpha) c_{v,\,w} p_{u,\,v}(m)}{\sum\limits_{x < v} p_{x,\,v}(m)},$$

где $w = \mathrm{argmax}_{u < w < v} c_{v,\,w}$.

Первое слагаемое числителя строится аналогично тому, как это делается в алгоритме, представленном в работе [15]. Второе слагаемое отражает зависимость вероятности копирования вершиной v сообщения m из u от вероятности копирования того же сообщения вершиной w из u. Вершина w, в свою очередь, определяется как создающая максимальную зависимость v от себя. Константа $0 < \alpha < 1$ характеризует силу зависимости пользователей от окружения для каждого вида сетей и определяется экспериментально. В знаменателе стоит нормирующий делитель.

Вычисление вероятностей копирования новостной темы из u в v отличается от базового алгоритма лишь самим наличием вероятности k_v вместо $k_{u,v}$:

$$p_{u,v}(m) = \frac{r_{u,\,v}(1 - r_{u,\,v})^{\delta_{u,\,v}(m)} k_v}{\sum\limits_{x < v} p_{x,\,v}(m)}.$$

Отсутствие слагаемого $c_{v,w} p_{u,w}(m)$ здесь обусловлено независимостью распространения новостей от отношений пользователей друг с другом.

Этап 6. Вычисление параметров $r_{u,v}$, $k_{u,v}$ и k_u. Вычисление вероятностей $r_{u,v}$, $k_{u,v}$ происходит по тем же формулам, что отмечены ранее, однако лишь для вершин, между которыми присутствуют ребра графа $G_{\min}$:

$$r_{u,\,v} = \frac{\sum\limits_{m \in S_1} p_{u,\,v}(m)}{\sum\limits_{m \in S_1} p_{u,\,v}(m) \delta_{u,\,v}(m)};$$

$$k_{u,\,v} = \frac{\sum\limits_{m \in S_1} p_{u,\,v}(m)}{\sum\limits_{m \in S_1 \cup S_2} Pr_{geom}[r_{u,\,v} \leqslant \delta_{u,\,v}(m)]}.$$

Множества сообщений S_1 и S_2 выбирают только из множества обычных сообщений вершин u и v.

Вероятность k_u вычисляется по следующей формуле:

$$k_u = \frac{card(S_3)}{\sum\limits_{v \in Env(u)} \left[\sum\limits_{m \in S_4 \cup S_5} Pr_{geom}[r_{u,\,v} \leqslant \delta_{u,\,v}(m)] \right]}.$$

В данной формуле S_3 — множество всех новостных сообщений вершины u; $Env(u)$ — множество вершин графа $G_{\min}$, из которых исходят ребра, направленные к u; множества S_4 и S_5, определяемые для каждой $v \in Env(u)$, — это множества всех новостных сообщений, которые либо достигли u после v, либо, никогда не достигнув u, все же наблюдались в v соответственно. Метод построения данной формулы совпадает с предыдущим подходом, и формула пред-

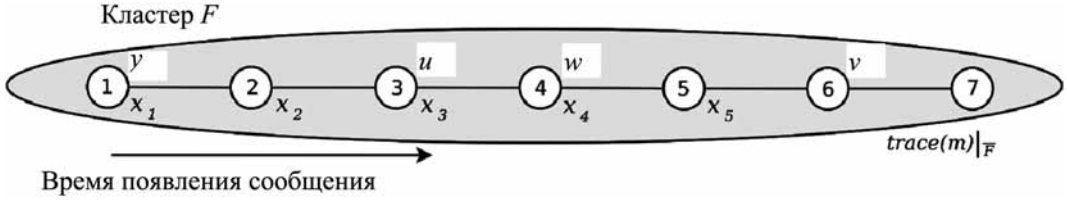


Рис. 3

ставляет собой отношение числа скопированных сообщений к оценке числа прочитанных.

Критерий остановки. Как и в модели, представленной в работе [15], критерием остановки будет расхождение значений $r_{u,\,v}$, $k_{u,\,v}$ и k_u предыдущей и текущей итераций алгоритма не более чем на 1 % в норме L_2.

Задача о минимальном покрытии

Задача о минимальном покрытии множества системой подмножеств рассматривается во многих работах, например [16, 17], и состоит в следующем. Пусть даны множество $M = \{1, ..., m\}$ и набор его подмножеств $M_1, ..., M_n$ таких, что $M_1 \cup M_2 \cup ... \cup M_n = M$. Совокупность подмножеств M_j, $j \in J \subseteq \{1, ..., n\}$ называется покрытием, если их объединение равно M. Каждому M_j приписан вес c_j, например, мощность этого множества. Требуется найти покрытие минимального суммарного веса.

В работе [18] показано, что задача NP-сложна. Вместе с тем существует большое число приближенных алгоритмов решения разной сложности и степени точности, например, описанные в работах [16, 19]. Анализ существующих решений поставленной задачи показал, что для гарантирования высокой точности решения потребуется высокая вычислительная сложность. Однако для применения к задаче восстановления графа $G_{\min}$ требуется прежде всего быстрый алгоритм, потому что исполняться он будет для каждой вершины. Следует заметить, что при восстановлении $G_{\min}$ точность нахождения минимального покрытия уступает по значимости тому факту, что реально присутствующие в сети связи не должны быть удалены. Это делает допустимым применение быстрых алгоритмов низкой точности, обеспечивающих определенный "запас" созданных ребер, который будет усечен на следующих итерациях алгоритма нахождения схем распространения информации.

Кластеризация графа контактов

Известно много различных алгоритмов кластеризации графов, но большинство из них работает с матрицей смежности. Это обусловливает вычислительную сложность порядка $O(N^3)$, где N — число вершин графа. Высокими оказываются также и требования к занимаемой памяти. Все это делает неприемлемым использование такого рода алгоритмов в итерационном процессе отслеживания распространения информации.

Решением вопросов повышения производительности может стать использование алгоритмов кластеризации типа "наводнение", производным от которого является и алгоритм "Иерархического Роста", предлагаемый в работе [20].

Алгоритм предназначен для кластеризации неориентированных невзвешенных графов. Он основан на последовательном добавлении к кластеру соседних с ним вершин либо обрезании ребра между кластером и вершиной в случае "отторжения" ее кластером. Таким образом обеспечивается последовательный рост кластеров.

Перед началом кластеризации проводится подготовительный этап — вычисление локальных коэффициентов кластеризации [6]. *Локальный коэффициент кластеризации* — это коэффициент, приписанный каждой вершине графа $v \in G_{\min}$. Он определяется как отношение числа треугольников из ребер таких, что v является вершиной треугольника, к числу двухреберных цепочек таких, что v стоит посередине цепочки (рис. 4).

Далее вводятся следующие два понятия, определенных для каждого кластера F.

- *Первое окружение* кластера S_1 — множество вершин графа, имеющих ребро, связывающее их с кластером F.
- *Второе окружение* кластера S_2 — множество вершин графа, имеющих ребро, связывающее их с первым окружением.

На основе этих понятий для каждой вершины $v \in S_1$ определяется множество $S_2(v)$ соседних с v вершин, которые принадлежат к S_2, и следующие величины:

- $k_{in1}(v)$ — число ребер, соединяющих v с кластером F или первым окружением S_1;
- $k_{out1}(v)$ — число ребер, соединяющих v с оставшейся сетью $G_{\min}\backslash(F \cup S_1)$;
- $k_{in2}(v)$ — число ребер, соединяющих $S_2(v)$ с S_1;
- $k_{out2}(v)$ — число ребер, соединяющих $S_2(v)$ с оставшейся сетью $G_{\min}\backslash(F \cup S_1 \cup S_2)$.

Алгоритм выглядит следующим образом.

1. Берется следующая вершина $v \in G_{\min}$ с наибольшим локальным коэффициентом кластеризации и отождествляется с кластером F.

2. Рассматривается первое окружение S_1: для всех $v \in S_1$ вычисляются $k_{in1}(v)$ и $k_{out1}(v)$. Если $k_{in1}(v) \geqslant k_{out1}(v)$, то v добавляется к F на следующем шаге итерации, если этого не происходит — выполняется шаг 3.

3. Рассматривается второе окружение S_2 и для всех v, не принятых в F по первому критерию, вычисляются $k_{in2}(v)$ и $k_{out2}(v)$. Если $k_{in1}(v) \geqslant \alpha_{clust} k_{out1}(v)$, где $\alpha_{clust} > 1$ — некоторая константа, то v добавляется к F на следующем шаге итерации, если этого не происходит, то между v и F удаляются все ребра.

4. Если за предыдущую итерацию, начавшуюся с шага 2, кластер F вырос, то повторяется шаг 2. Если кластер перестал расти, он добавляется к ответу и удаляется из рассмотрения. При наличии оставшихся в сети вершин повторяется шаг 1.

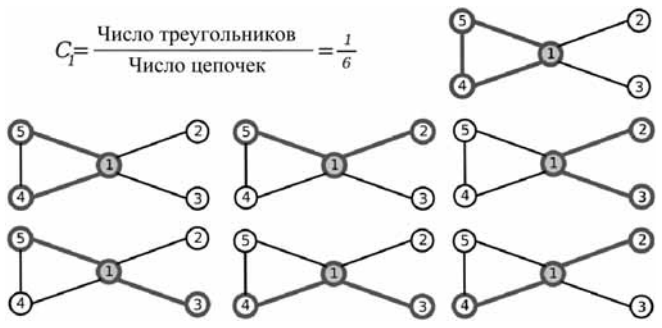


Рис. 4

Заключение

Проведен анализ существующих на настоящее время подходов к решению задачи определения путей распространения информации в социальных сетях и web-блогах. Выявлены достоинства, недостатки и рамки их применения.

Предложены модификации математических моделей, отражающих динамику распространения сообщений в социальных сетях. Такие модификации позволяют расширить возможности применения существующих методов решения задачи.

Предложены алгоритмы, позволяющие улучшить точность или производительность вычислений на базе существующих методов решения задачи.

В следующей публикации по рассматриваемой тематике предполагается демонстрация и анализ результатов тестирования программного комплекса, реализующего предложенные в настоящей статье модели и алгоритмы.

Работа выполнена при поддержке гранта РФФИ № 13-07-00582.

Список литературы

1. **Leskovec J., Krause A., Guestrin C., Faloutsos C., VanBriesen J., Glance N.** Cost-effective outbreak detection in networks // Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. 2007. P. 420—429.

2. **Cha M., Mislove A., Gummadi K. P.** A measurement-driven analysis of information propagation in the flickr social network // Proceedings of the 18th international conference on World wide web. 2009. New York: ACM, 2009. P. 721—730.

3. **Bailey N.** The Mathematical Theory of Infectious Diseases and its Applications. Griffin, 1975.

4. **Girvan M., Callaway D. S., Newman M. E. J., Strogatz S. H.** Simple model of epidemics with pathogen mutation // Physical Review E. 2002. Vol. 65. ID.031915.

5. **Moore C., Newman M. E. J.** Epidemics and percolation in small-world networks // Physical Review E. 2000. Vol. 61, Is. 5. P. 5678—5682.

6. **Watts D. J., Strogatz S. H.** Collective dynamics of `small-world' networks // Nature. 1998. Vol. 393. P. 440—442.

7. **Newman M. E. J., Forrest S., Balthrop J.** Email networks and the spread of computer viruses // Physical Review E. 2002. Vol. 66, Is. 3. ID. 035101.

8. **Libai B., Goldenberg J., Muller E.** Talk of the network: A complex systems look at the underlying process of word-of-mouth // Marketing Letters. 2001. Vol. 12 (3). P. 211—223.

9. **Kempe D., Kleinberg J., Tardos É.** Maximizing the spread of influence through a social network // Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. 2003. P. 137—146.

10. **Granovetter M**. Threshold Models of Collective Behavior// American Journal of Sociology. 1978. Vol. 83, Is. 6. P. 1420—1443.

11. **Morris S.** Contagion // Review of Economic Studies. 2000. Vol. 67, Is. 1. P. 57—78. URL: http://ideas.repec.org/a/bla/restud/v67y2000i1p57-78.html.

12. **Peyton H. Young.** The Diffusion of Innovations in Social Networks. 2000. URL http://ideas.repec.org/p/jhu/papers/437.html.

13. **Richardson M., Domingos P.** Mining knowledge-sharing sites for viral marketing // Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. 2002. P. 61—70.

14. **Goyal A., Bonchi F., Lakshmanan L. V. S.** Learning influence probabilities in social networks // Proceedings of the third ACM international conference on Web search and data mining. 2010. P. 241—250.

15. **Gruhl D., Guha R., Liben-Nowell D., Tomkins A.** Information diffusion through blogspace // Proceedings of the 13th international conference on World Wide Web. 2004. P. 491—501.

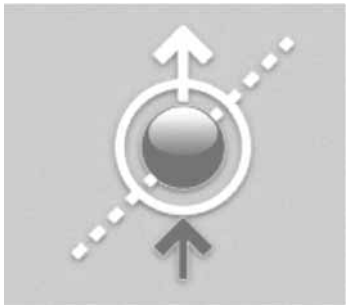
16. **Khuller S.** Algorithms column: the vertex cover problem // Sigact News. 2002. Vol. 33, Is. 2. P. 31—33.

17. **Еремеев А. В., Заозерская Л. А., Колоколов А. А.** Задача о покрытии множества: сложность, алгоритмы, экспериментальные исследования // Дискретный анализ и исследование операций. Серия 2. 2000. Том 7, № 2. С. 22—46.

18. **Журавлев Ю. И.** Теоретико-множественные методы в алгебре логики // Проблемы кибернетики. 1962. Вып. 8. С. 5—44.

19. **Bar-Yehuda R., Even. S.** A linear time approximation algorithm for the weighted vertex cover problem // Journal of Algorithms. 1981. N 2. P. 198—203.

20. **Rodrigues F. A., Travieso G., Costa L. da F.** Fast Community Identification by Hieraichical Growtn // International Journal of Modern Physics C. 2007. Vol. 18, Is. 6. P. 937—948.



VIII Международная научная конференция "Параллельные вычислительные технологии" (ПаВТ'2014)

31 марта — 4 апреля 2014 г.,
Южный федеральный университет (г. Ростов-на-Дону)

ПаВТ — серия международных научных конференций, представляющих собой авторитетный и престижный форум в области применения параллельных вычислительных технологий в различных областях науки и техники.

Учредителями конференции являются Российская академия наук и Суперкомпьютерный консорциум университетов России.

В первый день работы конференции будет объявлена ***20-я редакция списка Top50*** самых мощных компьютеров СНГ.

Официальный сайт конференции: http://ПаВТ.РФ

УДК 004.656

К. Е. Селезнёв[1,2], канд. техн. наук, вед. инж.-программист, e-mail: skostik@relex.ru,
А. А. Владимиров[1], аспирант, e-mail: alcobass@gmail.com
[1]Воронежский государственный университет,
[2]ЗАО НПП РЕЛЭКС, г. Воронеж

Морфологические словари на основе бит-векторов

Рассматривается новый способ построения модулей морфологического анализа. Предлагаемый подход незначительно уступает уже существующим реализациям по скорости и суммарному объему используемой памяти. Однако он позволяет проводить морфологический анализ слов, содержащих опечатки и преднамеренные ошибки. Такие функциональные возможности востребованы в системах, при использовании которых авторы текстов заинтересованы в неправильной обработке своих документов. Ярким примером подобных систем являются спам-фильтры.

Ключевые слова: *морфологический анализ, опечатки, ошибки, искажения*

K. E. Seleznyov, A. A. Vladimirov

Morphological Dictionaries Implementation Based on Vectors of Bit

The paper describes new approach of morphological analysis implementation. It is based on dictionaries, but uses bitmaps for data indexing and searching inside dictionaries. It doesn't allow to improve analysis quality and speed, but it allows to process words which contains errors and misprints.

Keywords: *morphological analysis, errors, misprints*

Введение

Современные системы интеллектуальной обработки текстовых документов немыслимы без модулей морфологического анализа [1,2], которые на входе принимают текст слова, а на выходе возвращают его основную словоформу и набор граммем. При неоднозначном анализе слова возвращаются все его возможные трактовки в виде пар "основная слово-форма — набор граммем". Согласно логике конкретного естественного языка граммемы объединяют в грамматические категории (часть речи, время, род и т.д.), при этом различные граммемы одной категории несовместимы друг с другом. Это означает, что у одного слова может быть только одна граммема из каждой категории.

Существуют три принципиально разных подхода к построению модулей морфологического анализа. Первый заключается в использовании одного большого морфологического словаря, который для каждого слова содержит ссылку на его основную словоформу и информацию о граммемах. Для русского языка этот словарь строится на основе словаря Зализняка [3]. Достоинствами данного подхода являются его сравнительная простота и универсальность. Простота заключается в том, что давно разработаны и исследованы различные динамические структуры данных для работы с множеством слов. В качестве наиболее известных примеров подобных структур следует упомянуть B-деревья и хэш-таблицы [4]. Универсальность рассматриваемого метода заключается в том, что его можно использовать практически для любых естественных языков.

Недостатки морфологического анализа на основе словаря напрямую следуют из его же достоинств. При использовании универсальных структур данных никак не учитывается специфика исходных словарей. Например, известно, что в русском языке в большинстве случаев все формы одного и того же слова различаются только окончаниями. Если для каждой формы хранить только текст окончания, то это позволит существенно сократить суммарный объем информации. Кроме того, при использовании универсальных структур данных не учитывается тот факт,

что словарь генерируется один раз, а потом используется только в режиме чтения.

Более существенным недостатком является то обстоятельство, что построенные на основе словарей модули морфологического анализа способны корректно обрабатывать только те слова, которые явно указаны в используемых словарях. Если же слово не содержится в исходном словаре, то модуль морфологического анализа возвратит пустой результат. На практике часто приходится иметь дело со словами, которые в принципе не могут быть известны при построении словаря. Яркими примерами таких случаев являются собственные имена, слэнговые выражения, а также слова с преднамеренными ошибками и искажениями.

Второй подход к построению модулей морфологического анализа заключается в создании системы правил, которая позволяет по исходному слову определить его возможные морфологичекие характеристики. Правила могут быть разработаны экспертом-лингвистом или выведены с помощью алгоритма машинного обучения. Однако и в том, и в другом случае этот подход хорошо применим только в языках с "бедной" морфологической моделью и строгой системой правил словообразования. Например, в английском языке имена существительные изменяются только по числам (единственное—множественное). При этом заранее известны как четкие правила написания той или иной формы, так и список слов-исключений.

Подход на основе правил плохо применим для языков со сложной морфологической моделью и правилами словообразования. Причина в том, что, во-первых, трудно создать исчерпывающую систему правил и, во-вторых, гарантировать ее точную работу. Именно поэтому часто морфологический анализ на основе системы правил называют неточным и противопоставляют его анализу на основе словарей, называемому точным. Если же к модулю морфологического анализа предъявляются дополнительные требования по обработке слов с ошибками и преднамеренными искажениями, то сложность разработки системы правил возрастает еще сильнее.

Третий подход к морфологическому анализу заключается в комбинировании двух предыдущих. Так, сначала проводится анализ с помощью словарей, и если он не дает результатов, то используется система правил морфологического анализа для предсказания возможных характеристик слова.

Недостатком этих трех подходов является то обстоятельство, что они ориентированы на работу лишь с правильно написанными словами и плохо приспособлены для анализа слов с ошибками. Более того, данные подходы практически неприменимы для слов с преднамеренно допущенными ошибками и искажениями.

В данной статье предлагается новый подход к построению модулей морфологического анализа, имеющих следующие характеристики:

1) анализ осуществляется на основе предзаданного словаря;

2) на вход модуля анализа поступает отдельное слово;

3) на выходе модуль анализа возвращает все возможные характеристики данного слова (род, число, падеж и т. д.);

4) анализ может осуществляться с учетом опечаток и преднамеренного искажения слов;

5) помимо текста слова на вход могут подаваться дополнительные ограничения, например "искать имена существительные в родительном падеже";

6) внутреннее построение и алгоритмы анализа основаны на манипулировании битовыми векторами;

7) скорость предлагаемых модулей ниже, чем модулей, функционирующих на основе хэширования и деревьев поиска, однако предлагаемые модули имеют достаточно высокое быстродействие;

8) предлагаемый подход применим для различных языков.

Далее в статье дается формальная постановка задачи, затем обсуждаются точный и неточный морфологический анализ, и далее описываются алгоритмы морфологического анализа с ограничениями на граммемы искомых слов.

Формальная постановка задачи

Пусть морфологическая модель естественного языка подразумевает n возможных граммем g_1, ..., g_n, а словарь языка содержит слова w_1, ..., w_m. Тогда для каждого слова w_i известна следующая информация:

$$\mathbf{W}_i = (w_i;\ b_i;\ w^*_{i,1}, \ldots, w^*_{i,m}), \qquad (1)$$

где b_i — это индекс слова, соответствующего начальной форме слова w_i, а $\mathbf{w}^*_i = (w^*_{i,1}, \ldots, w^*_{i,n})$ — вектор характеристик слова w_i, состоящий из n нулей и единиц. Значение $w^*_{i,j} = 1$ тогда и только тогда, когда слово w_i имеет граммему g_i.

В русском языке одно слово может иметь множество различных словоформ, например, слово "дом" имеет формы "дома", "домом", "домами" и т. д., при этом одна из словоформ является основной формой слова. Граммемы — это возможные характеристики слов языка. Например, в русском языке есть граммемы "имя существительное", "имя прилагательное", "мужской род", "женский род" и т. д. Для любого сло-

ва языка всегда известен набор граммем, описывающих характеристики этого слова.

Слово w_k является начальной формой слова w_i в том случае, если выполняется условие $b_i = k$, при этом слово может быть своей же начальной формой, т. е. допустимо $b_i = i$. Если слово w_i является начальной формой другого слова w_j, то w_j должно быть начальной формой самого себя, т. е., если выполняется $b_i = k$, $k \neq i$, то должно выполняться $b_k = k$. Например, слово "домом" является формой слова "дом", а слово "дом" — формой самого себя. Возможна ситуация, когда слово является своей начальной формой, но при этом не имеет никаких других форм. Например, предлог "на" не имеет никаких словоформ.

Морфологический анализ слова w без учета опечаток заключается в нахождении множества всех значений i, для которых выполняется:

$$w = w_i.$$

Морфологический анализ слова w с учетом опечаток заключается в нахождении множества всех значений i, для которых выполняется:

$$R(w;\ w_i) \leqslant r, \qquad (2)$$

где $R(a;\ b)$ — функция, которая должна удовлетворять свойствам метрики, показывая, можно ли считать слово a словом b, написанным с опечатками или преднамеренными ошибками. Чем сильнее отличаются a и b, тем большее значение должна давать функция $R(a;\ b)$ на выходе, и, наоборот, $R(a;\ b) = 0$ тогда и только тогда, когда слова a и b полностью совпадают по написанию. Функция $R(a;\ b)$ может быть построена на основе метрики Левенштейна для сравнения текстовых строк, но лучше использовать специально подобранную функцию, работающую с учетом вероятных и маловероятных опечаток.

Параметр r в формуле (2) задает максимально допустимое значение метрики $R(w;\ w_i)$ и позволяет регулировать степень неточности проводимого морфологического анализа. При $r = 0$ результаты точного и неточного анализа совпадают.

Точный анализ

В случае русского языка для каждого слова w_i можно построить набор логических признаков $a_{i,1}$, ..., $a_{i,33}$, где признак $a_{i,k}$ включен, если слово содержит k-ю букву алфавита. Для других языков максимальное значение k зависит от числа букв в алфавите. Все введенные признаки a можно объединить в вектор:

$$\mathbf{A}_i = (a_{i,1},\ ...,\ a_{i,33}). \qquad (3)$$

Вектор **A** можно рассматривать как ключ поиска векторов $\mathbf{W}_i$ (см. формулу (1)), при этом одному ключу поиска **A** может соответствовать множество различных значений $\mathbf{W}_i$. Разумеется, одному и тому же ключу могут соответствовать различные слова, состоящие из одного и того же набора букв, поэтому далее потребуются дополнительные проверки.

Возникает классическая задача индексации и поиска информации, которую можно решить с помощью B-деревьев [4], многомерного поиска [5], хэширования или каких-либо других соответствующих методов. Далее будет рассмотрен альтернативный способ, основанный на бит-векторах.

Набор векторов $\mathbf{A}_i$ можно представить в виде матрицы **A**, состоящей из нулей и единиц, у которой строки соответствуют словам, а столбцы — отдельным логическим признакам. По поданному на вход слову w можно определить, какие логические признаки в нем присутствуют, затем взять нужные столбцы матрицы **A** и перемножить их соответствующие элементы. В результате выполненных действий будет получен вектор **F**.

$$\mathbf{F}_i = \mathbf{A}_{i;k1} \times ... \ \mathbf{A}_{i;kL}, \qquad (4)$$

где k_1 ... k_L — это номера столбцов матрицы **A**, соответствующих признакам слова w. Полученный вектор $\mathbf{F}_i$ состоит из нулей и единиц, и обладает важным свойством: если $\mathbf{F}_i = 1$, то слово w_i обладает теми же логическими признаками, что и анализируемое слово w. Если же $\mathbf{F}_i = 0$, то слово w_i гарантированно отличается от w. Кроме того, следует отметить, что число ненулевых элементов вектора $\mathbf{F}_i$ будет, как правило, невелико.

Пошаговый алгоритм точного морфологического анализа выглядит следующим образом.

1. Для анализируемого слова w определяют значения его логических признаков.

2. Из матрицы **A** берут столбцы, соответствующие логическим признакам слова w. Матрицу **A** имеет смысл хранить "по столбцам", точнее, каждый ее столбец записать в отдельный файл с использованием сжатия [6]. Это решение позволит сократить объем ввода-вывода и читать из внешней памяти только реально необходимую часть **A**.

3. У полученных на предыдущем шаге столбцов перемножают соответствующие элементы, в результате чего формируются вектор **F**. При выполнении данного шага можно применять, по меньшей мере, три вида оптимизации. Во-первых, для вычисления $\mathbf{F}_i$ можно использовать не целочисленную арифметику, а побитовые операции "И", поскольку все операнды равны 0 или 1. Если модуль морфологического анализа работает на 32-разрядной архитектуре, то указан-

ный прием дает выигрыш в 32 раза, на 64-разрядной архитектуре — в 64 раза. Во-вторых, не обязательно распаковывать используемые столбцы целиком, т. е. содержимое каждого файла можно дораспаковывать по мере необходимости. В-третьих, вместо вектора **F** можно хранить только номера его ненулевых элементов.

4. Рассматривают все $\mathbf{F}_i = 1$ и соответствующие им w_i сравнивают с исходным w. Если оказывается $w_i = w$, то вектор $\mathbf{W}_i$ добавляется к результату анализа. Если в исходном словаре w_i отсортированы по алфавиту, то для выполнения данного шага можно использовать модифицированный алгоритм бинарного поиска. В целях оптимизации данный шаг можно объединить с предыдущим, т. е. сразу после вычисления $\mathbf{F}_i$ осуществлять проверку условия $w_i = w$, и в случае успеха добавлять $\mathbf{W}_i$ к результату анализа.

Рассмотренный алгоритм обладает хорошим быстродействием. Формальные рассуждения показывают, что вычислительная скорость алгоритма линейно зависит от числа слов в словаре. Однако такой факт будет наблюдаться только в том случае, если какая-то буква присутствует во всех без исключения словах, т. е. столбец матрицы **A** целиком состоит из одних единиц. На практике буква языка встречается не более чем в 70 % слов. Результаты для русского языка будут приведены позднее при обсуждении эксперимента.

Рассмотренный алгоритм предъявляет сравнительно невысокие требования к ресурсам (расход памяти будет оценен ниже) и может быть существенно оптимизирован. Однако описание технических деталей выходит за рамки данной статьи. Более важным свойством предложенного алгоритма является тот факт, что он открывает возможности выполнения неточного морфологического анализа, рассматриваемые далее.

Неточный анализ

Между столбцами матрицы **A** можно ввести взаимосвязи, основанные на информации о наиболее частых заменах символов или ошибках. Например, столбец для буквы "а" связан со столбцом для буквы "о", поскольку эти буквы часто путают или преднамеренно взаимозаменяют. Данный факт можно учитывать при вычислении вектора **F** следующим образом. Пусть у анализируемого слова w присутствует признак a_i, связанный с признаками $a_{i,1}$, ..., $a_{i,k}$. Тогда при построении $\mathbf{F}_i$ вместо a_i можно использовать логический признак $\mathbf{a}_i^*$, элементы которого вычисляются по следующему простому правилу:

$$\mathbf{a}_i^* = a_i \text{ или } a_{i,1} \text{ или ... или } a_{i,k}.$$

Таким образом, для выполнения неточного морфологического анализа в рассмотренном выше алгоритме необходимо изменить шаг 3. Он должен состоять из двух этапов: вычисления значений $\mathbf{a}_i^*$, а затем вычисления значений $\mathbf{F}_i$. В результате такого преобразования шаг 3 алгоритма анализа по-прежнему будет основываться на высокоскоростных битовых операциях.

В силу рассмотренного изменения шага 3 на последующем за ним шаге 4 становится необходимо просматривать абсолютно все w_i, у которых $\mathbf{F}_i = 1$, и вычислять значение $R(w; w_i)$, показывающее степень сходства слов w и w_i. Если окажется $R(w; w_i) < r$, то w помещается в результат. На завершающем этапе требуется отсортировать найденные результаты по возрастанию значений $R(w; w_i)$.

Анализ с ограничениями

Рассмотренный выше метод и его алгоритмы позволяют проводить морфологический анализ слов с ограничением на морфологические характеристики этих слов. Для этого векторы $\mathbf{A}_i$ из формулы (4) должны быть расширены элементами из $\mathbf{w}_i^*$, т. е. в матрицу **A** должны быть добавлены новые столбцы, соответствующие граммемам. Этот прием позволит одновременно искать слова и на основе букв, и на основе граммем. Например, если нужны все имена существительные, в которых встречается буква "а", то нужно взять столбец "имя существительное", столбец "а" и выполнить логическое "И". Полученный бит-вектор даст искомые слова.

Добавленные столбцы не отличаются от остальных столбцов матрицы **A**, и, следовательно, могут участвовать при определении значений $\mathbf{F}_i$ по формуле (4). Именно благодаря этому факту становится возможным указание условий вида "может ли слово w быть существительным", для чего в формуле (4) нужно просто умножать полученное $\mathbf{F}_i$ на значение $A_{i,k}$, где k — номер столбца, соответствующего граммеме "имя существительное".

Важно отметить, что рассмотренный прием может быть использован как для точного, так и неточного морфологического анализа. Второй случай более интересен, поскольку позволяет, например, выполнять запросы следующего вида: "можно ли заданное слово считать искаженной формой некоторого существительного, прилагательного и т. д.". На предполагаемые

результаты анализа могут накладываться любые ограничения в терминах граммем языка.

Эксперименты

Целью экспериментов, описанных в статье, является оценка объемов памяти, необходимой для хранения морфологического словаря согласно изложенному выше подходу, а также оценка числа операций, необходимых для обработки одного слова. За исходные данные было взято множество характеристик всех словоформ, построенное на основе словаря Зализняка [3], состоящее из 8 033 800 слов и 52 граммем. Таким образом, расширенная матрица **A** состоит из 8 033 800 строк и 33 + 52 = 85 столбцов, что в результате перемножения дает 682 млн элементов. Для их хранения требуется чуть более 80 Мбайт памяти. Если тексты слов хранить в однобайтной кодировке и для каждого слова использовать 10 байт, то для хранения всех слов потребуется примерно 80 Мбайт памяти. Таким образом, в сумме потребуется примерно 80 + 80 = 160 Мбайт, что по современным меркам вполне приемлемо.

Указанные ниже оценки даны без учета возможностей сжатия матрицы **A**. В таблице приведена вероятность встречаемости букв русского алфавита в словах, а параметр "сжатие" показывает отношение размера сжатого столбца матрицы к изначальному размеру в процентах. Для сжатия использовался архиватор zip, работающий по алгоритму LZW [6].

Аналогичные данные для векторов, соответствующих граммемам, в данной статье не приведены в силу большого объема соответствующей таблицы. Итоговый размер всей матрицы **A** после сжатия равен 2,8 Мбайт, что составляет примерно 2 % от исходного размера.

Тексты слов w_i тоже можно хранить в сжатом виде, предварительно разбив их на блоки, каждый из которых будет содержаться в отдельном файле. Например, 8 033 800 слов можно разбить на 81 блок по 100 000 слов. Если необходим текст i-го слова, то сначала проводится чтение и декомпрессия нужного блока, а уже оттуда извлекается текст самого слова.

Хранение словаря в сжатом виде оказывается крайне эффективным с точки зрения расхода памяти, но требует дополнительного процессорного времени при выполнении анализа. Вследствие этого в высоконагруженных системах, выполняющих многократный морфологический анализ, словарь целесообразно хранить в распакованном виде. В "маленьких" системах, требующих обработки лишь нескольких слов, матрицу **A** можно хранить в упакованном виде. Возможно и компромиссное решение, когда часто используемые столбцы матрицы **A** хранятся в исходном виде, а редко используемые распаковываются по мере необходимости.

Буква	Вероятность	Сжатие, %	Буква	Вероятность	Сжатие, %
а	67,72	9,06	р	51,63	2,47
б	13,76	1,09	с	51,76	5,31
в	61,62	7,84	т	35,39	6,49
г	15,89	6,98	у	29,08	9,79
д	19,43	1,89	ф	2,85	0,56
е	64,85	9,53	х	21,62	9,6
ж	7,31	1,45	ц	3,38	1,48
з	18,99	1,51	ч	13,28	2,89
и	67,76	0,6	ш	36,33	6,01
й	16,45	0,13	щ	21,4	4,26
к	24,88	3,94	ъ	0,39	0,16
л	32,84	5,71	ы	32,51	8,37
м	44,56	9,84	ь	6,18	8,41
н	41,96	6	э	0,8	0,18
о	62,24	7,2	ю	23,37	0,07
п	34,23	1,03	я	33,03	1,06

Как было отмечено выше, каждому слову w_i соответствует бит-вектор $\mathbf{A}_i$ из формулы (3), но такое утверждение в обратную сторону неверно. Одному и тому же вектору $\mathbf{A}_i$ могут соответствовать разные слова. Более формально это означает, что существуют $i \neq j$, такие, что $\mathbf{A}_i = \mathbf{A}_j$, но $w_i \neq w_j$. Например, слова "улыбок" и "кобылу" состоят из одних и тех же букв, и соответствующие им строки матрицы **A** полностью совпадают. Ниже приведена статистическая информация о числе слов с одинаковым значением $\mathbf{A}_i$.

Число различных значений $\mathbf{A}_i$ 838 521
Минимальное число слов с одинаковым $\mathbf{A}_i$ 1
Среднее число слов с одинаковым $\mathbf{A}_i$ 9,6
Максимальное число слов с одинаковым $\mathbf{A}_i$ 1428

Исходя из полученных результатов можно сделать вывод, что на заключительном шаге морфологического анализа потребуется выполнять сравнительно небольшое число проверок $w = w_i$ при точном морфо-

логическом анализе или проверок $R(w; w_i) < r$ при неточном анализе.

Заключение

Рассмотрен новый способ построения морфологических словарей на основе бит-векторов и соответствующих матриц. Предложенный подход позволяет проводить как точный, так и неточный морфологический анализ. Дополнительная возможность рассмотренного подхода — анализ слов с учетом ограничений на их предполагаемые характеристики.

Есть основания полагать, что применение рассмотренного метода позволит существенно ускорить работу и повысить точность более высокоуровневых методов обработки текста, и, прежде всего, синтаксического анализа, однако для подтверждения этого обстоятельства требуются дополнительные исследования. Во-первых, предложенный набор логических признаков основан на присутствии в слове отдельных букв. Необходимо разработать более эффективный набор признаков, который будет более точно характеризовать слово и его типичные искажения. Это означает, что при намеренном или случайном искажении слова набор присутствующих в нем логических признаков не должен меняться, или должен меняться предсказуемым образом. Во-вторых, необходимо разработать математический аппарат для неточного сравнения слов с точки зрения случайных или преднамеренных искажений. Например, в русскоязычных социальных сетях типичным искажением слова является замена окончания "ся" на "ца" с многократным повторением буквы "ц". Для эффективной обработки таких ситуаций набор логических признаков должен иметь отдельные признаки "ся" и "ца", связанные таким образом, что если слово имеет "ца" на конце, то в словаре необходимо проводить поиск слов, оканчивающихся на "ся". Аналогично метрика сравнения слов должна быть построена таким образом, чтобы слова, оканчивающиеся на "ся" и "ца", считались почти идентичными.

Список литературы

1. **Ножов И. М.** Реализация автоматической синтаксической сегментации русского предложения. дисс. ... канд. тех. наук. М.: РГГУ, 2003.
2. **Селезнёв К. Е.** Обработка текстов на естественном языке // Открытые системы. 2003. № 12. URL: http://www.osp.ru/os/2003/12/183694/
3. **Зализняк А. А.** Грамматический словарь русского языка. Словоизменение. М.: "АСТ-ПРЕСС", 2008. 794 с.
4. **Кормен Т., Лейзерсон Ч., Ривест Р.** Алгоритмы: построение и анализ. М.: Вильямс, 2006.
5. **Гулаков В. К., Трубаков А. О.** Многомерные структуры данных. Брянск: Изд-во БГТУ, 2010.
6. **Ватолин Д., Ратушник А., Юкин В.** Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. М.: Диалог-МИФИ, 2002.

ИНФОРМАЦИЯ

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

Дизайнер *Т. Н. Погорелова.* Технический редактор *Е. М. Патрушева.* Корректор *М. Г. Джавадян*

Сдано в набор 05.12.2013 г. Подписано в печать 22.01.2014 г. Формат 60×88 1/8. Заказ PI214
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1.