ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

# Программная инженерия

ПрИн 4 2013

Учредитель: Издательство "НОВЫЕ ТЕХНОЛОГИИ"

Издается с сентября 2010 г.



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

## СОДЕРЖАНИЕ



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

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

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

УДК 004

**М. В. Пирогов**, канд. физ.-мат. наук, нач. сектора, ФГУП НПО им. С. А. Лавочкина, г. Химки, Московская область
e-mail: pmv_mvp@mail.ru

# Радикальное программирование

*Рассматривается используемая на практике технология программирования и задачи, к необходимости решения которых привело ее применение. Для решения этих задач предлагается новая технология — технология радикального программирования, основанная на концепции среды радикалов и формализме схем радикалов.*

***Ключевые слова:*** *программная инженерия, программное средство, критическая система, интеллектуализация, среда радикалов, схема радикалов*

## Введение

**Информатика — средства и проблемы.** Относительно короткая, но стремительная в своем развитии, история современной информатики производит **двойственное впечатление**. С одной стороны, были созданы многочисленные программно-технические средства (ПТС), нормативные и методические документы, предназначенные для решения самых разных задач. Разработчику в настоящее время доступны возможности структурного, объектно-ориентированного и визуального программирования, систем управления базами данных (СУБД), разнообразные сетевые программные средства (ПС) и многое другое. Мир ПТС продолжает быстро развиваться. С другой стороны, продолжает оставаться неразрешенным ряд проблем принципиального характера. Сложность задач, объектов и процессов, для автоматизации которых применяются ПТС, а также роль таких ПТС на всех этапах жизненного цикла (ЖЦ) сложных систем (СС) быстро возрастают. Положение дел еще более усугубляется применительно к **критическим системам** (КС) — СС, которые характеризуются жесткими требованиями безотказности их функционирования [1]. Причины отказов, с учетом возрастающей роли ПС в КС, заключаются, как правило, в ошибках и недостатках этапа разработки программного обеспечения таких систем. Постоянно появляется информация, свидетельствующая о многочисленных проблемных ситуациях в таких КС, которые приводят к отказам, как следствие, к масштабным потерям, к опасности для жизни людей и даже катастрофам. Таким образом, крайне необходимо создание средств, в том числе ПТС, которые гарантировали бы отсутствие конфликтных ситуаций, приводящих к отказам КС и обеспечивали их безопасность.

Следует отметить, что на создание, сопровождение и модификацию ПС потрачены и продолжают тратиться **огромные ресурсы**. При этом достигнутые с их помощью результаты далеко не всегда удовлетворительны, а значительная часть разработанных в последние годы языков программирования практически не имела реального применения.

Если придерживаться пессимистической точки зрения, то складывается следующая картина. Задачи СС, ПС и ПТС будут быстро усложняться, задачи надежной верификации ПС обеспечения их отказоустойчивости решены не будут. Как следствие, не позднее 2025 г. (как предсказывает B. Boehm [2]) произойдет катастрофа, вызванная дефектами ПС.

**Технология программирования.** За все годы развития современной информатики так и не удалось создать пригодные для практического применения программный инструментарий и технологию программирования, которая позволяла бы в автоматизированном режиме сопровождать весь ЖЦ сложных ПС различного назначения и обеспечивала бы решение задач их верификации и поддержки отказоустойчивости. Такая технология и соответствующие ПС, с учетом возрастающей сложности и объективного единства мира СС (и мира программных СС), жизненно необходима.

Представляется, что одним из выходов в создавшейся ситуации могла бы стать рассматриваемая в настоящей работе новая технология — **технология радикального программирования (ТРП)**. Данная статья не претендует, разумеется, на ее исчерпывающее описание. Рассматриваются лишь некоторые характерные черты ТРП. Создание такой технологии — это, очевидно, масштабная задача со всеми вытекающими от-

сюда последствиями. И тем не менее сложившаяся в информатике (и программировании) ситуация и возможная перспектива ее неблагоприятного развития делают такой подход очень востребованным.

## Технология серых модулей

**Что возникло на практике.** Эффективная для практического применения технология программирования, охватывающая всю проблемную область и обеспечивающая решение задач верификации и поддержки высокого уровня отказоустойчивости сложных ПС, создана не была. Образовалась определенная пустота, которая начала быстро заполняться другой технологией — технологией чисто прагматического свойства. У этой технологии изначально были серьезные недостатки. Главный из них в том, что она никогда не была настроена на решение принципиальных и, соответственно, трудоемких задач. Однако, как оказалось, эта технология стала удобной для многих, так как человеческий фактор, особенно в интеллектуальной сфере, всегда играл важнейшую роль. Условно назовем ее **технологией серых модулей (ТСМ)**, подчеркивая тем самым следующий аспект программирования. Современные инструментальные ПС не обеспечивают эффективное решение задач верификации и поддержки отказоустойчивости сложных ПС, однако допускают и поддерживают создание (и даже довольно быстрое) таких программных продуктов, о которых никто, включая их авторов, не может сказать (точно и достаточно оперативно), как эти модули устроены и как они работают. Отметим также, что вместо слова "серые" вполне можно было бы употребить слова "темно-серые" или даже "практически черные" модули. В значительной степени именно благодаря появлению и развитию ТСМ появились и стали популярными афоризмы (принципы), согласно которым "каждая сколько-нибудь сложная программа содержит ошибки" и что "легче написать свой код, чем разобраться в чужом". Далее будем рассматривать возможные ситуации в программировании и в целом — в ЖЦ ПС, которые, в принципе, допускают современные инструментальные ПС, и к которым, по существу, подталкивает ТСМ.

**Формальные описания.** Важнейшей характерной чертой ТСМ является **принципиальное отсутствие** достаточно полных, унифицированных и удобных формальных описаний проблемных областей и их объектов, относящихся к разряду (классу) СС, автоматизация которых реализуется с помощью ПС. Нет применимых на практике ПС для создания и ведения таких описаний. Отсутствуют такие описания для целей, доступных ресурсов и конфликтов, которых необходимо избегать. Это предоставляет специалистам, участвующим в сопровождении ПС на всех этапах их ЖЦ, некоторую **свободу действий**. Такая свобода, на самом деле, является **мнимой** и оборачивается в конечном счете против создаваемого ПС, перспектив его успешного применения и модификации. При отсутствии адекватных реалиям и программно реализованных формальных описаний автоматизируемых объектов проблемной области, всерьез говорить о надежной верификации сложных ПС и обеспечении их отказоустойчивости нельзя.

Программно реализованные достаточные формальные описания отсутствуют, зато в изобилии присутствует разнообразная текстовая документация (ТД). И это тоже характерная черта ТСМ. Состав ТД имеет тенденцию к расширению, а ее объем — к увеличению, требования к ее оформлению постоянно совершенствуются. Комплекс работ, связанных с ТД, ее созданием и прохождением (обращением), вовлекает в свою сферу большое число специалистов, тем самым отвлекая их от других более важных содержательных задач. Эти работы требуют значительных временных и иных ресурсов, являются ярким примером действия человеческого фактора в составе ТСМ. Соответствие ТД программе и ее коду автоматически не обеспечивается. Такая документация традиционно неполна и крайне инертна. Она в принципе не могла бы конкурировать с формальными описаниями, реализованными программно и имеющими удобную для практической работы форму. Она просто не успевает, да и не может успеть за постоянными изменениями сложной проблемной области в течение ЖЦ ПС, за частыми модификациями ПС.

**Фактические стандарты.** Применение ТСМ привело к тому, что, так или иначе, устанавливаются соответствующие ей неофициальные стандарты "де-факто", касающиеся разработки и модификации ПС, выпускаемой документации, отношений между специалистами и коллективами специалистов, сопровождающих ПС на различных этапах их ЖЦ. Технология серых модулей существенно влияет на нормативные документы, сопроводительную документацию и литературу методического характера. По существу, ТСМ устанавливает свои собственные "правила игры".

**Крупные проекты.** Многочисленные результаты применения ТСМ налицо. В России по разным каналам в рамках крупномасштабных национальных программ осуществляется финансирование проектов, имеющих в своем составе ПТС. Выделяемые на их выполнение средства значительны, конечные результаты, к сожалению, **неудовлетворительны** [3, с. 5]. Практическое применение созданных в рамках выполнения таких проектов аппаратно-программных средств и систем создает **дополнительные трудности**, а затраты на преодоление этих трудностей соизмеримы, а иногда и превосходят затраты на разработку самих средств.

Отметим, что в условиях ТСМ и соответствующей ТД большой и сложно организованный программный продукт, разрабатываемый и сопровождаемый в рамках крупного проекта, практически становится неверифицируемым (с учетом потребности в оперативной информации и необходимости в объективно частой модификации). Это создает существенные трудности как при разработке ПС, так и при его эксплуатации и

модификации, поскольку часто требуется работать быстро, охватывать весь проект (или значительную его часть) "одним взглядом" то по одному, то по другому поводу, выделять нужные области, принимать решения и осуществлять те или иные действия.

Приведем еще один яркий пример, демонстрирующий результаты применения ТСМ. Это одна из российских проблем настоящего времени, напрямую влияющая на экономику. Проблема связана со сложностями перевода активно используемых на практике больших программных комплексов на современные высокопроизводительные системы — параллельные вычислительные платформы [3, с. 16]. Такие комплексы были предназначены для решения стратегически важных задач в области атомной энергетики, авиакосмического машиностроения и в других, в том числе критически важных секторах национальной экономики. В эти комплексы многие годы вкладывались знания большого числа ученых и инженеров. Однако в ходе разработки, сопровождения и модернизации этих ПС далеко не всегда соблюдались стандарты программной инженерии (еще одна характерная черта ТСМ), и теперь перенос этих ПС вызывает очень большие сложности. Это может означать только одно — применение ТСМ привело к **фактической утрате** значительного объема очень ценных знаний.

Если говорить о ПС как о продуктах индустриального способа производства, то требования к ним должны предусматривать [3, с. 2] их более высокие **функциональные свойства** (при ТСМ они ограничены наличием принципиально важных и нерешенных задач надежной верификации ПС и обеспечения их отказоустойчивости), эффективную **отчуждаемость** ПС от создателя (производителя) и, как следствие, возможность **модификаций** их многочисленными пользователями (потребителями) самостоятельно (с ТСМ не совместимо в принципе — здесь разработчики всеми силами стремятся "привязать" ПС к себе); а также высокое качество выполняемых функций (при ТСМ это опять-таки ограничено возможностями верификации).

**Инструментальные программные средства.** В условиях ТСМ возникли многочисленные и разнообразные инструментальные ПС. Например, появилось большое число языков программирования. Характерная черта инструментальных ПС ТСМ — предоставление программисту значительной **свободы**, которая оборачивается для него дополнительными **проблемами**. Никто не требует от программиста формального описания (с помощью инструментального ПС) всех целей создаваемого ПС, всех доступных ресурсов и потенциально возможных конфликтов, которых уже на ранних стадиях ЖЦ следует избегать. Не надо фиксировать все используемые модели. Вовсе не требуется достаточного (а лучше бы избыточного — на случай нештатных ситуаций) описания всей проблемной области. Получается, что все это программист должен держать, с учетом необходимости оперативного доступа, в своей голове.

Рассматривая ПС в процессе ее разработки или модификации (да и после ее завершения) без участия автора, объективно трудно бывает понять, куда, к каким целям движется программист, и чего он на самом деле достиг. Программа при ТСМ есть цель самой себя, которая заключается в выполнении (или, наоборот, невыполнении) своих операторов. При таком подходе трудно, на самом деле, надеяться, что в конце процесса разработки (модификации) "все сойдется" и "будет, как надо".

Важнейшей задачей эффективного программирования для СС с обеспечением должного набора функций и высокого качества их выполнения была и остается задача создания адекватных потребностям практики средств автоматизированного накопления и использования **программистских знаний** [3, с. 12]. Исследования здесь особенно важны для разработки сложных программных комплексов — для реализации крупных проектов с территориально распределенными на сетевой среде группами исполнителей. Должно быть обеспечено, в том числе наличие средств, позволяющих в автоматизированном режиме поддерживать **положения программной инженерии** на всех этапах разработки программ, включая средства эффективного описания спецификаций, верификации, тестирования и др. Программисту должны быть доступны сущности, которые необходимо учитывать и анализировать при разработке, эксплуатации и модернизации сложных программных комплексов, комплексов, предназначенных для массового использования на различных платформах (в различных операционных средах). Кроме того, для создания и развития ПС, поддерживающих совместную работу территориально распределенных коллективов, крайне важным является выбор отвечающих их потребностям сервисов, которые хранятся в сетевой среде. Относительно этих задач можно с уверенностью сказать, что их успешное решение в принципе не совместимо с ТСМ, с характерным для нее отсутствием достаточных и программно реализованных формальных описаний проблемной области и широким использованием ТД в современном ее виде.

**Тестирование.** В этом аспекте также существуют недостатки [3, с. 12], главные из которых заключаются в отсутствии доказательной **строгости и "полноты покрытия"** программы испытаний, т**рудности составления и реализации** программы испытаний для сложных объектов. В итоге возникает сложность сопоставления результатов тестовых испытаний, полученных разными группами исследователей. Необходимость повышения эффективности, снижения затрат и сокращения времени испытаний диктуют необходимость их автоматизации на основе применения **формальных математических моделей**. Однако используемые в настоящее время подходы, как правило, **нереализуемы на практике** для тестирования сложных систем, состоящих из большого числа иерархически организованных компонентов. Последнее обстоятельство связано с проблемой корректности декомпозиции. Причем

**задача верификации декомпозиции требований в настоящее время общего решения не имеет**. В целом больших практически значимых результатов в разработке доказательно строгих формальных моделей, адекватно описывающих процессы тестовых испытаний, пока не получено, и многие вопросы остаются без ответов. На практике проблема тестирования частично решается с помощью контрольных примеров. В какой степени, вообще говоря, точно неизвестно.

Путь к решению проблемы тестирования видится в переходе к постепенному, **пошаговому тестированию ПС** в процессе его создания. Утрируя, можно сказать, что "качество надо производить, а не контролировать". Такой подход возможен только на основе достаточной формализации проблемной области, что при ТСМ недостижимо. Это одна из задач новой, отвечающей современным требованиям технологии программирования.

**Модификации.** Модификации ПС практически **неизбежны**. Их осуществляет программист и обычно в сжатые сроки. Здесь многое зависит непосредственно от конкретного специалиста (и, конечно, от требуемой модификации). Функциональность модификации не предусматривается заранее, иначе она была бы реализована и составляла часть функциональности ПС. Отсюда, вообще говоря, **чужеродность** модификации для уже разработанного ПС. Правда, опытный программист может предвидеть некоторые варианты развития событий и разработать ПС в соответствии с ними. Однако предусмотреть можно далеко не все. Кроме того, надо обеспечить **целостность** системы в смысле необходимости выполнения ею новых функций, что при ТСМ представляет серьезную трудность. В любой момент можно ожидать требования модификации, осуществить которую не удастся, по крайней мере, в заданные сроки. Можно ли в принципе осуществить требуемую модификацию при имеющихся ресурсах? Как это сделать? На эти вопросы ТСМ, вообще говоря, ответа не дает.

**Ресурсы.** Выше были приведены два примера, демонстрирующих результаты применения ТСМ: современные проекты в рамках крупномасштабных российских национальных программ и перевод больших отечественных программных комплексов на современные высокопроизводительные системы. Эти примеры ярко демонстрируют еще одну особенность ТСМ. Ее применение практически неизбежно приводит к **перерасходу ресурсов**, причем весьма масштабному. Перерасход ресурсов может быть явным, а также скрытым, трудно определяемым. Он может быть отсроченным — проявляющимся через некоторый, возможно значительный, промежуток времени. Он может быть постоянным и временным, может прерываться и возобновляться.

При ТСМ отсутствуют достаточные и реализованные программно формальные описания проблемной области, имеющей сложное строение и постоянно изменяющейся. Соответственно, нет достаточно точных **оценок, моделей использования ресурсов** всех необходимых видов для всех уровней проблемной области и всех этапов ЖЦ ПС. Вместо этого ТСМ предлагает ТД, искусство ее создания и использования. В результате, проблемная область, включая растекающиеся в различных направлениях ресурсы разных видов и объемов, оказывается плохо обозримой и слабо контролируемой. Создаются благоприятные условия для неоправданного роста потребляемых ресурсов (одновременно со стремлением к минимизации прилагаемых усилий). Как результат — при ТСМ тратились и продолжают тратиться разными путями и способами огромные ресурсы на создание, а затем и эксплуатацию ПС и ПТС **заведомо низкого качества**. Применение ТСМ ведет к образованию в проблемной области ПС своеобразных "черных дыр", буквально поглощающих ресурсы разных видов.

**Пользовательский интерфейс.** К пользовательскому интерфейсу ПС, созданных в условиях ТСМ, вполне применимы характеристики "сложный", "громоздкий" и "непонятный". Конечно, его никак нельзя назвать по-настоящему дружественным и тем более интеллектуальным. Изучение и освоение такого пользовательского интерфейса требует значительных усилий. Постоянно возникают, вопросы, например: какими функциональными возможностями обладает ПС, как до них добраться, как реально воспользоваться? И ответы на эти вопросы получить совсем не просто. Общепринятый и лучший способ их получения — попросить помощи у того, кто в них уже разобрался (правда, такого специалиста может рядом и не оказаться), а вовсе не использование кнопки "Помощь" и чтение руководств, инструкций и книг, также носящих яркий отпечаток ТСМ. Большинство таких источников, кажется, вовсе не торопится снабдить читателя знаниями, необходимыми для практической работы и представленными в удобной для поиска и восприятия форме, а вместо того преследуют какие-то свои цели.

**Код и модели.** О программном коде следует сказать особо. При использовании ТСМ "программа — это код". Фактически, в коде достаточно сложного ПС разбирается только его автор — программист. Никто другой, без крайней необходимости, читать этот код, а тем более вмешиваться в него не будет. Такие попытки, если они и предпринимаются, даже в лучших случаях имеют, как правило, весьма ограниченный успех. Любая работа с кодом **крайне трудоемка**. Вероятность ошибки здесь не просто велика. Как показывает практика, ошибки здесь совершенно неизбежны, и речь может идти лишь о недопущении фатальных ошибок. Комментарии вполне могут быть плохо понятными (совсем непонятными), недостаточно полными, они могут вообще противоречить коду. Кроме того, далеко не все можно и удобно выразить с помощью комментариев.

При этом многие модели, для реализации которых писался код, не фиксируются и практически утрачиваются. Работая с кодом, такие модели трудно (а зачастую практически невозможно) обнаружить. Они "не видны"

в коде. Отметим также, что при ТСМ модель — это одно, а **программно реализованная модель** — уже нечто другое. В ЖЦ ПС модели, по крайней мере некоторые, необходимо часто **корректировать**, и это требование **не соответствует** возможностям, предоставляемым кодом. В ТСМ, благодаря использованию, в том числе кода, получаются плохо обозримые, слабо контролируемые, "жесткие", трудно модифицируемые ПС. При модификациях ПС изменения в коде имеют характер "заплаток". Они чужды разработанному ранее коду, с трудом, поспешно "втиснуты" в него. При частых модификациях таких "заплаток" становится слишком много. Код, который и раньше был не слишком прозрачным, все больше "затуманивается", становится все более громоздким, трудночитаемым и, соответственно, совсем трудно модифицируемым. "Серый" модуль постепенно превращается сначала в "темно-серый", а затем — в "практически черный" модуль.

Код (коды) — существенная составляющая ТСМ — неудобен, неприемлем для человека в сколько-нибудь значительных объемах и должен быть заменен другим формализмом.

**Программирование как искусство.** В России разработка ПС сложилась и преобладает в настоящее время на основе подхода "программирование как искусство" [3, с. 4]. Добавим, что этот стиль программирования можно было бы определить (если употреблять сам термин исключительно в отрицательном смысле) как ТСМ, доведенную до предела. Программирование как искусство — это работа по принципу "делаю, как хочу" и "код мой и предназначен он только для меня". Если руководитель проекта потребует, "чтобы были комментарии", то комментарии будут добавлены, однако должного понимания для постороннего взгляда они все равно не обеспечат. В условиях информационного общества при разработке программных продуктов массового использования или критически важных программных комплексов такой подход неэффективен и **неприемлем**.

**Передача модуля.** При ТСМ единственный способ передать модуль (модули) "старого" программиста "новому" программисту — это организовать (если удастся) их достаточно продолжительную совместную работу. Предполагается, что в процессе такой работы "старый" программист будет активно передавать (для этого нужны стимулы) и, в конце концов, передаст все свои знания "новому" программисту. Возможно, что-то и передаст, но за исключением тех знаний, о которых он забыл, либо по какой-то причине не хотел передавать. Либо он уже сосредоточен на совершенно других проблемах и ему не до передачи знаний. Таким образом, при ТСМ никогда нельзя быть уверенным, что подобный процесс передачи знаний прошел достаточно успешно. "Новый" программист вполне может оказаться беспомощным, оставшись один на один с ПС в некоторой ситуации, которая может наступить в любой, в том числе самый неподходящий момент.

**Необходимость изменений.** Что же следует предпринять в связи с проблемами, возникшими в условиях ТСМ? Здесь существуют разные точки зрения. Например, следующая. **Ничего особенного делать не следует.** Надо продолжать работать по-прежнему. К установившемуся порядку вещей все уже привыкли. А издержки — они есть везде, они неизбежны. Вообще, все на самом деле не так уж плохо. Отдельные недостатки и даже проблемы, конечно, имеются. Но, во-первых, так было всегда, а, во-вторых, есть и старые проверенные меры, которые кажутся вполне естественными для борьбы с этими недостатками. Есть, например, всевозможные организационные и финансово-экономические рычаги. Их надо использовать, и использовать более решительно. Надо проводить реорганизации, надо больше и жестче требовать с программистов. Они должны работать быстрее, писать больше операторов в единицу времени, оперативно выпускать требуемую ТД, которая должна строго соответствовать действующим нормативным документам. Они обязаны постоянно (и лучше самостоятельно) повышать свою квалификацию — изучать все новые и новые ПС. Нет сомнения, совсем скоро за рубежом будут созданы новые, еще более совершенные и многочисленные программные инструменты и ПТС, обещано решение многих проблем. Надо будет быстро осваивать эти средства, работать с ними. И ситуация, конечно, улучшится. А если созданное ПС подведет и произойдет событие, которое приведет, например, к утрате значительных ресурсов, то и здесь почти все понятно. Виноватым, скорее всего, окажется программист, допустивший "программную ошибку". Это, конечно, неприятно, но такое случается. И с этим, на самом деле, трудно что-либо поделать в принципе. Ведь это — человеческий фактор, и соответствующие события трудно предсказуемы. Но, конечно, этот отрицательный опыт будет учтен, и все необходимые меры будут приняты.

Существует и другая точка зрения — противоположная. Согласно ей — **изменения необходимы и это должны быть изменения принципиальные и качественные**. Проблемная область программирования для СС **быстро усложняется**. Возрастает сложность решаемых задач, возникает потребность в новых средствах и системах автоматизации. Требуются частые модификации разработанных ПС. Возрастает цена ошибки. Ужесточается ситуация с ресурсами всех видов. Растут сложность и разнообразие программных инструментов. Работая в проблемной области одной СС, необходимо все в большей степени учитывать проблемные области других СС, объективное единство мира СС, а также окружающую природную среду. В этих условиях отчетливо проявилась принципиальная **неспособность** ТСМ справиться с усложняющейся ситуацией, что вовсе не является неожиданностью. Настрой и направленность ТСМ были понятны уже давно. Их можно выразить в принципах "выбирать задачи с умом" и "не делать лишней работы". В полном соответствии с этими принципами, большая часть усилий

направлялась на решение разнообразных (и часто похожих) локальных задач. Упорно работали над "понятными" частными задачами, решение которых обещало быстрый успех. Принципиальными проблемами, в том числе задачами верификации и повышения уровня отказоустойчивости сложных ПС, занимались существенно меньше. В результате они так и не были решены. Вначале, когда требования к разрабатываемым ПС по большей части были не так критичны, а мир ПС не был таким насыщенным и разнообразным, отсутствие решений для этих проблем не так бросалось в глаза. Теперь, с ужесточением требований, недостатки ТСМ проявились в полной мере. Для ТСМ это тупиковая ситуация. Те ПС, которые (относительно) успешно работают в проблемных областях различных СС, работают так **не благодаря** ТСМ, но **вопреки** ей. Работают благодаря высокой квалификации, опыту, изобретательности, предусмотрительности и чувству ответственности программистов и других специалистов, имеющих непосредственное отношение к ПС. В сложившейся ситуации, в мире быстро усложняющихся ПС, этих замечательных положительных качеств уже недостаточно. Практика постоянно подтверждает, что в рамках ТСМ в принципе не могут быть решены проблемные вопросы верификации и отчуждаемости, модифицируемости и высокой отказоустойчивости ПС. Локальные успехи возможны, но в целом положение будет только ухудшаться. Свойственная ТСМ прагматичность не позволит даже подступиться к решению перечисленных принципиальных задач. Никакие организационные и финансово-экономические меры, направленные на поддержку ТСМ не помогут, а выделенные ресурсы будут задействованы, но какой-либо пользы для дела не принесут. В сущности ТСМ не соответствует требованиям современной информатики, не имеет никакой перспективы, изжила себя и должна быть **заменена** другими технологиями. Необходимо отметить, что создалась ситуация, когда ТСМ, благодаря своей прагматичности (в худшем смысле этого слова) вовлекла в свою сферу большое число самых разных специалистов, связанных между собой на разных этапах ЖЦ ПС отношениями, которые совершенствовались годами и десятилетиями и стали уже привычными. Это обстоятельство является самым серьезным препятствием на пути поиска радикальных средств выхода из создавшегося положения.

С учетом изложенного выше, несмотря на серьезные препятствия, необходимы новые технологии программирования, лишенные недостатков ТСМ и имеющие достоинства, которые позволили бы справиться с проблемами современных ПС. Они должны быть научно обоснованы и, в то же время применимы на практике для всех этапов ЖЦ ПС. Эти принципы ориентируют разработчиков на создание и внедрение в практику реальных секторов экономики инновационно перспективных и **опережающих** традиционные методов, средств и систем автоматизации. Эти принципы изначально предполагают использование **положений программной инженерии** на всех этапах ЖЦ программного обеспечения таких систем — положений, учитывающих, в том числе экономические факторы, а также психологические аспекты, связанные с коллективом исполнителей проектов, и т. п. Методы и средства программной инженерии должны быть развиты и усовершенствованы, модернизированы путем применения современных подходов к их реализации, путем максимального использования математического аппарата и формальных моделей, описывающих семантику программ, поддающихся надежной верификации в автоматизированном режиме [3, с. 10]. **Стандарты программной инженерии** должны [3, с. 10]: полностью охватывать все этапы ЖЦ ПС; соответствовать современным темпам развития средств вычислительной техники и телекоммуникаций; удовлетворять потребностям практики, особенно в части программного обеспечения систем автоматизации сложно организованных объектов, которые остро востребованы реальной экономикой.

Добавим к этому, что положения и стандарты программной инженерии должны быть не только существенно доработаны (переработаны) и расширены. Они должны быть формализованы, и формализованы адекватно проблематике СС, а затем реализованы с помощью ПТС.

## Технология радикального программирования

**Программное средство как сложная система.** Принимая во внимание изложенные выше соображения, необходимы новые подходы и основанные на них технологии программирования, направленные на устранение отмеченных недостатков технологий существующих. В качестве одной из альтернатив традиционным подходам предлагается **технология радикального программирования (ТРП)**. Слово "радикальное" указывает на радикальный разрыв с основами господствующей в настоящее время технологии серых модулей. При этом все полезное, созданное вопреки ТСМ, в ТРП будет аккумулировано. Есть еще одна причина использования слова "радикальное" в названии новой технологии программирования. Она будет изложена несколько позже.

Характерной особенностью ТРП является то, что в ней **ПС на любом этапе своего ЖЦ рассматривается как СС (КС)**. В мире искусственных СС существуют недостатки и проблемы (и это необходимо учитывать), но есть и успехи. Так, например, Московский метрополитен, безусловно, являющийся критически сложной системой, вот уже многие десятилетия успешно эксплуатируется (в самых разных условиях) и при этом модифицируется (расширяется) и верифицируется на предмет соответствия предъявляемым к нему требованиям. С этих позиций ТРП ПС представляет собой СС. Это означает, что оно должно создаваться, эксплуатироваться и модифицироваться также (похоже), как и другие сложные искусственные (тех-

нические) системы. Существуют самые разные искусственные СС, однако им присуще и нечто общее. Например, наличие цели (целей), которую необходимо достичь; ограниченные ресурсы: запас (библиотеки) готовых решений; внутренние и внешние конфликты, от которых надо уходить, и которые надо разрешать и т. д. В ТРП проблемная область ПС является открытой. Она "расширяется", образно говоря, до проблемной области СС.

**Информационно-системная безопасность.** По мнению автора, ключевая проблема СС — это **информационно-системная безопасность (ИСБ)**. Основные ее положения сформулированы А. В. Чечкиным [4]. Они включают в себя положения традиционной информационной безопасности и отражают следующие аспекты.

- Информационная устойчивость СС: решение любой задачи ЖЦ СС должно быть обеспечено независимо от формы и полноты исходной информации путем ее защиты от помех и логического получения необходимой дополнительной информации. Обеспечение информационной устойчивости СС требует информационного, основанного на методологии информатики подхода ко всей проблемной области СС.
- Системная безопасность СС: решение любой задачи в жизненном цикле СС должно учитывать не только эту задачу, но и соответствующие последствия для проблемной области СС, требование бесконфликтности в течение всего ЖЦ СС, в том числе системную целостность СС. Обеспечение системной безопасности требует системного подхода к описанию и учету всех характеристик (атрибутов, значений и т. п.) проблемной области.

Основная цель предлагаемой ТРП состоит в следующем. Для обеспечения ИСБ ПС СС необходима их **интеллектуализация** на основе избыточной **моделирующей среды**, требующей специального формализма. Интеллектуальность понимается как способность успешно действовать в нештатных ситуациях, способность к выделению в проблемной области ПС СС узких мест, постановке и решению новых критически важных задач.

**Требования к технологии радикального программирования.** Приведем некоторые важные **требования к ТРП** с точки зрения "ПС как СС".

1. ТРП должна быть **универсальной** — она должна быть применима к проблемной области любого ПС.

2. ТРП должна основываться на едином специальном формализме ПС, обеспечивающем принципиальную возможность единой формализации (формального описания) проблемной области любого ПС.

3. В ТРП должны быть учтены в полном объеме и единстве преследуемые ПС **цели**, доступные **ресурсы** и возможные **конфликты**. Должно быть обеспечено прогнозирование, классификация, оценка и уход от конфликтов (разрешение конфликтов).

4. ТРП должна упорядочить, нормализовать и сделать прозрачной разработку и эксплуатацию ПС СС. На основе этого при приоритете **математического и программного обеспечения**, должна быть упорядочена разработка и эксплуатация **других видов обеспечения**, присущих сложным системам автоматизации: информационного, технического, методического, лингвистического, организационного, правового, эргономического, метрологического.

5. Должны быть учтены все значимые **составляющие** проблемной области ПС, все их значимые **свойства и связи**.

6. Должны существовать средства для **описания** проблемной области ПС, а также средства для **поиска, выделения** нужных описаний и их **активирования**.

7. Должна быть обеспечена оценка и, при необходимости, **обработка всех последствий** (возможно, отсроченная) изменений проблемной области ПС.

8. Должны быть в наличии средства для создания и упорядоченного развития специализированных **библиотек** большого объема, ориентированных на решение задач проблемной области ПС.

9. Должны быть предусмотрены средства, направленные на обеспечение процесса **интеллектуализации ПС**, в результате которого должно быть обеспечено: решение не только штатных, но и (некоторых) нештатных задач; самообучение ПС в процессе решения нештатных задач.

10. В случае необходимости, при решении задач должна запрашиваться и добываться дополнительная информация, инициироваться изменение, развитие ПС **извне**.

11. Должны быть в наличии средства для **визуализации** проблемной области ПС, для создания стабильного в своей основе и в то же время легко изменяемого и дружественного пользовательского интерфейса.

12. Должно поддерживаться четкое **разграничение** моделей и методов работы с ними, принадлежащих различным иерархическим уровням проблемной области ПС.

13. Средства ТРП не должны конфликтовать с другими инструментальными средствами, предназначенными для решения задач ЖЦ ПС. **Взаимодействие** с этим инструментарием, при необходимости, должно быть обеспечено (в пределах возможностей предоставляемых такими инструментами).

14. В рамках ТРП должна быть обеспечена перманентная **верификация** ПС в течение всего ЖЦ.

15. ТРП должна обеспечить создание и упорядоченное развитие полномасштабной интеллектуальной **моделирующей среды** проблемной области ПС.

16. ТРП должна быть направлена на поддержку постепенного **объединения** программного обеспечения таких моделирующих сред в программное обеспечение единой глобальной моделирующей среды "понимающих" друг друга и взаимодействующих друг с другом ПС. Выполнение последнего требования соответствовало бы объективному единству мира СС, погруженного в единую окружающую его естественную среду.

17. ТРП должна охватывать **все этапы, все задачи** ЖЦ ПС — от предпроектных исследований до эксплуатации и вывода ПС из эксплуатации.

**Результаты анализа математических и программных средств.** В целях разработки ТРП анализировались математические и программные средства СС. Анализ проводился с позиции необходимости обеспечения ИСБ ПС, интеллектуализации и охвата всех этапов ЖЦ ПС. Очевидны трудности такого анализа, связанные с разнообразием, многочисленностью и постоянным изменением таких средств. Проведенный анализ математических программных средств показал наличие у них многочисленных и разнообразных возможностей, и в то же время принципиальную **недостаточность** имеющихся средств для обеспечения ИСБ ПС, что подтверждается практикой.

**Концепция среды радикалов.** Анализ математического и программного обеспечения СС привел к выводу о целесообразности выбора в качестве основы формализма избыточного моделирования ПС СС **концепции среды радикалов** [5—7]. Главным понятием математической информатики, методология которой положена в основу этой концепции, является понятие **радикала**. И это еще одна причина, по которой предлагаемая новая технология программирования названа радикальной. Под радикалом понимается любая функциональная система, имеющая два доступных извне состояния (два множества состояний) — **активное и пассивное**. Активный радикал функционирует согласно своему предназначению, а пассивный радикал — нет. Он как бы выключен. Множество связанных между собой радикалов образует **среду радикалов**. Понятие радикала позволяет взглянуть на ПС СС и его окружение как на среду радикалов. Радикалами такой среды являются: составляющие ПС и его окружения; их связи; задачи, которые решаются на разных этапах ЖЦ ПС СС (включая нештатные задачи); методы решения таких задач; специалисты, работающие в проблемной области ПС СС и коллективы специалистов; нормативные документы и многое другое (с учетом возможной вариативности). По мнению автора и других сторонников концепции среды радикалов, она соответствует специфике избыточного моделирования проблемной области ПС СС. В основе этой концепции лежит понятие среды радикалов — избыточной модели проблемной области. Последняя представляется множеством связанных между собой составляющих (всех значимых составляющих), включая различные их вариации. Эволюция ИСБ ПС СС обеспечивается с помощью **интеллектуальной системы**, решающей на основе концепции среды радикалов как штатные, так и нештатные задачи. Эволюция ПС СС с обеспечением ИСБ (ПС и СС) осуществляется с помощью интеллектуальной системы, решающей на основе концепции среды радикалов как штатные, так и нештатные задачи.

**Схемы радикалов** [4, 8]. Описание среды радикалов основывается на специальном формализме схем радикалов — **языке RADICAL**. Схемы радикалов вводятся следующим образом. Фиксируется алфавит. Из символов алфавита строятся **имена радикалов**. Из имен радикалов и символов "→" строятся **цепочки радикалов**. Например, цепочка радикалов N0 → N1 → → abc → N3 → N4; (здесь ";" — завершающий символ). **Схемой радикалов** называется любое множество цепочек радикалов. Среда радикалов представляется с помощью схемы радикалов. Введены **стандартные схемы радикалов (стандартные радикалы) — уникумы** (u) и **контейнеры** (c). Уникумы соответствуют компонентам проблемной области, контейнеры — свойствам (связям) таких компонентов.

Приведем пример контейнера c_mb_run, который используется для описания возможности запуска программных модулей. В этом примере используются уникумы-модули u_A, u_B, u_C, u_D и u_E. Пусть модуль u_A может (при определенных условиях) запустить модули u_B, u_C, u_D и u_E. Модуль u_C может запустить модуль u_D, а модуль u_B — модуль u_E. Соответствующая схема радикалов в символьном представлении имеет следующий вид:

$$c\_mb\_run \rightarrow \{d[1] \rightarrow u\_A;\ d[2] \rightarrow u\_B;\ d[3] \rightarrow u\_C;\ d[4] \rightarrow u\_D;\ d[5] \rightarrow u\_E;\}$$

$$c\_mb\_run \rightarrow \{d[1] \rightarrow u\_C;\ d[2] \rightarrow u\_D;\}$$

$$c\_mb\_run \rightarrow \{d[1] \rightarrow u\_B;\ d[2] \rightarrow u\_E;\}.$$

Здесь d[1], ..., d[5] — служебные радикалы, идентифицирующие направления, доступные в контейнере c_mb_run.

Для схем радикалов, кроме рассмотренного символьного представления, удобно использовать также **геометрическое отображение (векторное представление)**, а также представление с помощью таблиц (таб-

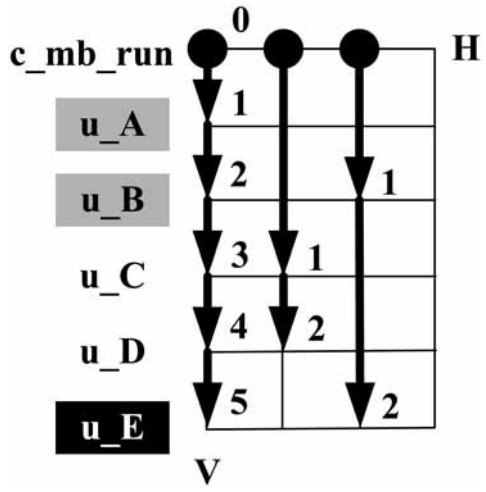



**Рис. 1. Геометрическое отображение контейнеров c_mb_run:** H и V — горизонтальная и вертикальная оси координат соответственно

| A | B | C | D | E | F |
|---|---|---|---|---|---|
| **-1** | **c** | **c_mb_run** | **0** | 0 | **0** |
| 1 | u | u_A | **1** | | |
| 2 | u | u_B | 2 | | **1** |
| 3 | u | u_C | 3 | 1 | |
| 4 | u | u_D | 4 | 2 | |
| **5** | **u** | **u_E** | **5** | | **2** |

**Рис. 2. Представление контейнеров c_mb_run с помощью таблиц:** поля A, D, E, F — целочисленные; поля B и C — строковые; первичный ключ составляет поле A

| A | B | C | D | F |
|---|---|---|---|---|
| **-1** | **c** | **c_mb_run** | **0** | **0** |
| 1 | u | u_A | **1** | |
| 2 | u | u_B | 2 | **1** |
| **5** | **u** | **u_E** | **5** | **2** |

**Рис. 3. Модули, запускающие модуль u_E**

лиц баз данных) (рис. 1 и 2). Отметим, что рассматриваемые представления не являются единственно возможными. Представления схем радикалов ориентированы на компьютерную реализацию, на использование цвета и движения, на быстрый переход от представления одного типа к представлению другого типа. Представления схем радикалов — основа пользовательского интерфейса среды радикалов. Рассмотренные представления демонстрируют возможность применения для схем радикалов средств теории графов, а также развитого инструментария баз данных.

Схемы радикалов удобны для манипулирования, для формирования "сечений" среды радикалов (для быстрого изменения точки зрения на среду радикалов), что исключительно важно для работы со сложной проблемной областью. Например, на рис. 3 представлен результат запроса обо всех модулях, запускающих модуль u_E. Видно, что таблица на рис. 3 получена из таблицы, представленной на рис. 2, путем удаления столбца E и строк A = 3, A = 4.

С помощью схем радикалов аналогичным образом представляются цели, ресурсы, конфликты, правила (ультраконтейнеры), запросы, задачи и методы их решения, активаторы (специализированные схемы радикалов, предназначенные для поиска ответов на запросы — к активаторам в конечном счете приводит использование методов решения задач), все значимые составляющие проблемной области (включая библиотечные составляющие), их свойства и связи. В схемах радикалов, как показывает опыт [9], удобно представлять ситуации, складывающиеся при программировании для СС. Представление осуществляется с учетом иерархичности проблемной области и степени "реализованности" создаваемого ПС (степени "проработки" проблемной области).

**Метод синтеза уникума.** Составляющие ПС СС могут испытывать в течение своего ЖЦ как штатные, так и нештатные воздействия. Некоторые из таких воздействий могут приводить к конфликтам, которых необходимо избегать. Конфликт понимается здесь весьма широко. Например, если при создании ПС выяснится, что для него необходимо разработать (еще один) новый модуль, то имеем дело с нештатным воздействием.

Это нештатное воздействие вызывает конфликт, который необходимо разрешить (путем разработки требуемого модуля). Если **при эксплуатации** ПС в составе СС происходит нечто непредвиденное, то это нештатное воздействие, которое, возможно, грозит конфликтом, подлежащим разрешению. Рассмотрим кратко метод, позволяющий решать **задачу синтеза целевого уникума** — важнейшую задачу, характерную для всех этапов ЖЦ ПС СС. Под целевым уникумом может пониматься некоторая подлежащая реализации составляющая проблемной области ПС, в том числе управляющее воздействие на ПС, уводящее СС от конфликта. Предлагаемый метод синтеза применим как в штатных, так и в нештатных ситуациях. Это **двунаправленный метод (ДНМ) ухода от конфликтов в среде радикалов — метод синтеза целевого уникума**. Отметим, что этот метод может применяться как в автоматизированном, так и в автоматическом режимах.

Итак, пусть необходимо построить целевой уникум uGoal, и требования к этому уникуму определяются целевыми контейнерами. Построение должно осуществляться с помощью **библиотеки радикалов**. Пусть в библиотеке найден библиотечный уникум, реализованный ранее и удовлетворяющий целевым контейнерам. Тогда поставленная задача решена. Если же такой библиотечный уникум не найден, то осуществляется попытка (попытки) реализации (построения) целевого уникума с помощью библиотечных радикалов, при помощи размещения (допустимым образом)

соответствующих уникумов в общих контейнерах. **Библиотека имеет иерархическое строение.** На **верхних уровнях** размещаются уникумы — **"обобщенные составляющие"**, представляющие классы составляющих "обобщенного ПС" (являющегося некоторым классом ПС). **На нижних уровнях** находятся уникумы, соответствующие **конкретным составляющим**, уже реализованным ранее. В попытках построить целевой уникум осуществляются многочисленные проходы "вниз" — от выбранных обобщенных составляющих к реализующим их конкретным составляющим, и проходы "вверх" — от конкретных составляющих к составляющим верхних уровней, контейнеры которых нуждаются теперь (после реализации конкретными составляющими) в уточнении. На каждом шаге, направленном вниз или вверх, будет осуществляться работа по верификации схем радикалов. Будем **пошагово** в автоматическом (автоматизированном) режиме **строить качественное верифицированное ПС**, а не только тестированный разработанный программный продукт или крупную его составляющую, когда трудно говорить о выявлении всех ошибок, а принципиальные изменения, на самом деле, уже невозможны. Такое пошаговое построение и верификация ПС — это путь к автоматизации программирования. Если работа ДНМ привела к выводу о нереализуемости целевого уникума, то это означает требование пополнения библиотеки радикалов новыми схемами, либо требование дополнительных ресурсов, либо требование и того и другого вместе.

Таким образом, **метод синтеза уникума** в среде радикалов, опирающийся на введенные средства, заключается в следующем. С помощью имеющихся ультраконтейнеров осуществляется обнаружение угрозы конфликта, его прогнозирование и классификация. Если конфликт **штатный**, то решение задачи — библиотечный уникум (вместе с характеризующими его контейнерами), уводящий среду радикалов от конфликта. Если конфликт **классифицирован как нештатный**, то в библиотеке радикалов может и не существовать уникума-решения задачи ухода от него. Тем не менее, поскольку точно это неизвестно, инициируется поиск такого решения в библиотеке радикалов. Если требуемый библиотечный уникум все-таки будет найден (библиотеки в ТРП избыточны), то задача ухода от конфликта решена. Если уникум не найден, то с помощью библиотеки радикалов осуществляется попытка синтеза верифицированного уникума, решающего задачу ухода от нештатного конфликта. В случае успеха появляется новая целостность — новый уникум, и построенная схема запоминается в качестве нового радикала — происходит самообучение среды радикалов.

Еще одна особенность описываемого метода синтеза уникума — это использование для решения задач схем, полученных в результате предварительного численного исследования среды радикалов на предмет ухода от гипотетических конфликтов, возникающих (относительно тех или иных подсистем контейнеров) в результате разнообразных **искусственно созданных угроз**, (направленных) **мутаций** схем радикалов. Такие исследования должны носить систематический и постоянный характер.

**Языковая среда радикалов.** Среда радикалов, включая библиотеки радикалов, имеет иерархическое строение. Доступно выделение уровней и манипулирование по определенным правилам (ультраконтейнерам) уникумами выделенного уровня, которые характеризуются определенными контейнерами (в зависимости от выбранной (предопределенной) точки зрения на среду радикалов, некоторые уникумы объявляются атомарными). В результате таких манипуляций части-уникумы погружаются в общие настраиваемые контейнеры, и, возможно, возникает искомая целостность — новый уникум, который может принадлежать уже другому иерархическому уровню. Уникумы этого уровня характеризуются уже другими контейнерами и ультраконтейнерами. Манипуляции ими осуществляются по другим правилам. Таким образом, уникумы среды радикалов оказываются связанными контейнерами, фиксирующими факты реализации одних уникумов с помощью других. Кроме того, в общем случае, для синтезированных уникумов в пределах заданных допусков имеет место вариантность их реализации, что также фиксируется с помощью контейнеров. Так, с помощью схем радикалов отображается объективно существующая целостность проблемной области ПС СС. Язык — это множество текстов, в нашем случае уникумов (схем) того или иного иерархического уровня среды радикалов. Уровней много и, таким образом, приходим к пониманию среды радикалов как множества единообразных языков, связанных между собой в смысле реализуемости уникумов одного уровня уникумами другого уровня (факты реализации фиксируются контейнерами реализации) в единую развивающуюся (расширяющуюся) языковую среду. Развитие это может происходить тремя путями: получение новых уникумов из уже имеющихся по правилам, присущим определенному иерархическому уровню; добавление извне новых уникумов к уже имеющимся уникумам определенного иерархического уровня; добавление извне новых иерархических уровней.

**Ключевые аспекты технологии радикального программирования.** Предлагаемая технология сосредоточена на ключевых для нее аспектах современной информатики в целом и программирования в частности. Это — **формализация, визуализация и интеллектуализация** проблемных областей, автоматизацию которых реализуют ПС СС. Без существенного продвижения в этих направлениях решение задач верификации и

поддержки должного уровня работоспособности таких ПС невозможно.

**Универсальность технологии радикального программирования.** В любой проблемной области при ее структурировании, как правило, приходится иметь дело со значимыми объектами (уникумами), их значимыми свойствами, связями (контейнерами) и правилами (ультраконтейнерами). Число уникумов, контейнеров и ультраконтейнеров для проблемной области ПС СС очень велико. Технология радикального программирования предоставляет средства для описания:

- составляющих проблемной области, принадлежащих разным иерархическим уровням;
- целей, которых необходимо достичь;
- возникающих задач и методов их решения;
- доступных ресурсов;
- недопустимых конфликтных ситуаций, которых необходимо избегать.

Для работы ПС в любой проблемной области требуются средства активации их описаний в целях решения поставленных задач/подзадач — необходимо формулировать запросы и логическим путем получать ответы на них. Технология радикального программирования имеет средства для этого. В любой проблемной области возникают как штатные, так и нештатные задачи, для решения которых используются библиотеки готовых решений. Средства ТРП позволяют с помощью библиотек радикалов решать оба вида задач — осуществлять поиск готовых решений, а также построение новых целевых объектов из уже имеющихся, соединяя их допустимым образом, осуществляя при этом верификацию получаемых решений и контролируя сложность создаваемой конструкции.

**Инструментальные программные средства для технологии радикального программирования.** Самый широкий спектр ПС и ПТС, как промышленного, так и офисного класса, может быть использован в качестве средств реализации среды радикалов и ТРП. Это, например, такие средства разработки приложений, как C++Builder и Delphi, графические библиотеки и системы геометрического моделирования, СУБД Microsoft SQL Server и Microsoft Access, средства логического программирования, инструментарий онтологий, разнообразные сетевые ПТС.

## Перспективы радикального программирования

Можно обоснованно предполагать, что предложенная ТРП, с учетом перспектив ее развития, лишена недостатков, присущих ТСМ, и удовлетворяет требованиям к ней, сформулированным выше. Применение формализма схем радикалов приносит, как показывает опыт, существенную пользу на практике [9]. Проблемная область ПС становится более обозримой, ее проще контролировать и легче принимать решения по самым разным вопросам. Далее кратко рассмотрим некоторые важные выгоды и изменения, ожидаемые в ходе развития ТРП.

**Средства технологии радикального программирования.** Развитие ТРП приведет к созданию ПС и ПТС нового класса, основанных на концепции среды радикалов. Для этих средств будут обеспечены приемлемые для практики:

- решение задач поддержки высокого уровня работоспособности верификации ПС СС;
- решение как штатных, так и (некоторых) нештатных задач.

Средства ТРП будут изначально тесно связаны между собой благодаря использованию общего формализма схем радикалов.

**Технология радикального программирования и программная инженерия.** Технология радикального программирования будет развиваться, основываясь в том числе на стандартах и рекомендациях программной инженерии [3, с. 8]:

- **по функциям**, которые программное обеспечение призвано выполнить по основному его назначению;
- **по обеспечению защиты** от реализации различных угроз собственно программного обеспечения, средств вычислительной техники и других ресурсов (включая информационные активы) автоматизированных систем, с которыми оно взаимодействует;
- **по обеспечению качества** всех перечисленных выше функций (в том числе высокой функциональности, надежности, удобства для пользователя, масштабируемости, мобильности и других показателей качества).

Рекомендации программной инженерии должны быть переработаны, расширены, формализованы и реализованы программно. Они должны стать неотъемлемой частью ПС ТРП.

**Жизненный цикл программного средства и его задачи.** С развитием ТРП, взгляд на ЖЦ ПС, его этапы и решаемые на этих этапах задачи существенно изменится. Широкое использование программно реализованных схем радикалов (схемы почти полностью заменят ТД на естественном языке, будет решена задача автоматизированного "извлечения" схем радикалов из ТД), использование численных оценок схем радикалов представят возможность оперативно оценивать ситуации изменяющейся проблемной области. Станут возможными: всесторонне обоснованный выбор первоочередных, критических направлений для дальнейшей проработки; обоснованное определение целесообразной глубины проработки на выбранных направлениях (вплоть до ввода в эксплуатацию), а также определение задач, решение которых может быть эффективно распараллелено.

**Моделирование.** В ТСМ модель описанная и модель программно реализованная на основе этого опи-

сания — это разные вещи. Программная реализация модели может существенно отличаться от исходного описания модели. В предлагаемой ТРП модель и ее программная реализация — это одно и то же. Причем ТРП требует обязательного и единообразного описания целей, ресурсов и конфликтов, описания всех значимых составляющих проблемной области, их значимых свойств и связей. Предусмотрена возможность изменения, уточнения таких описаний с обязательной (возможно, отложенной) обработкой последствий таких действий. В ТРП можно будет без задержки переходить к экспериментам с программно реализованной моделью уже на самых ранних стадиях ЖЦ, осуществляя ее постепенное построение, усложнение и модификацию.

**Сохранность моделей.** Технология радикального программирования сделает принципиально невозможной "потерю" моделей в ЖЦ ПС. При работе с библиотекой радикалов, что в ТРП является обязательным, все выбранные модели (все уникумы, контейнеры и ультраконтейнеры), принадлежащие разным иерархическим уровням, будут при встраивании их в создаваемую модель сохранены автоматически.

Сами библиотеки будут разрабатываться, испытываться и развиваться на единой основе — концепции среды радикалов и формализме схем радикалов — самым тщательным образом и как часть единой библиотеки, статус которой будет очень высок. Так, со всей возможной полнотой будут зафиксированы в удобной для дальнейшего использования форме (и с учетом вариантности) модели, имеющие отношение к различным СС и ПС СС. Библиотеки радикалов аккумулируют в себе и сохранят знания, опыт многочисленных специалистов высокого уровня и самых разных специальностей, которые будут привлечены к работе на удобных условиях.

**Исследование проблемных областей.** Предлагаемая ТРП исходит из того, что мир СС (ПС СС) исследован недостаточно. Технология серых модулей "не настроена" на глубокие исследования. При ее применении ориентируются на быстрые результаты, к тому же, нет (не созданы) необходимые для таких исследований программные инструменты. Между тем имеют место: быстрое и часто слабо контролируемое (неконтролируемое) развитие СС, их стремительно растущая сложность, многочисленные и трудно решаемые (нерешаемые) задачи взаимодействия СС между собой и с окружающей природной средой, возрастающее значение и сложность ПС СС, проблема верификации которых не решена. Все это решительно требует всестороннего и систематического исследования проблемной области СС (ПС СС) принципиально новыми средствами, к которым относятся средства ТРП. Различные сечения проблемных областей, представленные схемами радикалов, характерные и критические ситуации, возникающие при моделировании, программировании — все это подлежит глубокому изучению. В результате исследований будут выявляться новые закономерности, проявятся новые целостности, их свойства и связи, будут сформулированы новые задачи, подлежащие решению. Можно предположить, что в проблемной области ПС СС будут выявлены обширные "белые пятна", детально исследовать которые по каким-либо причинам пока не удается. При создании и эксплуатации ПС СС их придется "обходить". Результаты исследований проблемных областей СС (ПС СС) должны служить материалом для осмысления, обобщения, выработки практических рекомендаций и создания необходимых средств решения задач в рамках ТРП.

**Схемы радикалов и программные коды.** Технология радикального программирования придерживается следующего принципа: схемы радикалов (в различных представлениях) — для человека, коды — для ЭВМ. Кодогенерация по схемам радикалов должна осуществляться автоматически с помощью специальных библиотек радикалов. Можно предположить, что, развиваясь, язык RADICAL станет "последним языком программирования" в смысле той языковой среды, с которой непосредственно будет работать человек, решая задачи ЖЦ ПС СС.

**Модифицируемость.** При ТСМ модифицируемость ПС затруднена, а модифицируемость ПС силами пользователей затруднена крайне. Предлагаемая ТРП рассматривает (серьезную) модификацию ПС как решение нештатной задачи. Для решения таких задач силами пользователей ПС должно быть интеллектуальным — должно уметь решать (некоторые) нештатные задачи. Для решения нештатных задач в ТРП существует специальный метод — метод синтеза уникума, опирающийся на использование пополняемых библиотек радикалов, которые применяются для построения новых уникумов. Модифицируя ПС средствами ТРП, можно быть уверенным, что все последствия внесенных изменений будут строго контролироваться. Специалист, осуществляя модификации, имеет доступ (в автоматизированном режиме) только к (некоторым) схемам радикалов. Код предназначен исключительно для ЭВМ и должен генерироваться по модифицированным схемам радикалов автоматически.

**Старые и новые программные средства.** На практике довольно длительное время будет иметь место сосуществование ПС ТСМ и новых ПС ТРП. Предлагаемая ТРП не конфликтна по отношению к другим технологиям и средствам. ПС ТСМ для совместной работы с ПС ТРП должны быть описаны с должной степенью подробности с помощью схем радикалов (и это вполне естественное требование). Здесь, однако, возможны следующие трудности: совершенно "законные" вопросы о ПС ТСМ могут не находить ответа (такова природа этой технологии); возможны конфликты, связанные с несоответствием заявленного ПС ТСМ действительности.

Конечно возникает желание (и необходимость) разобраться в некоторых ПС ТСМ "полностью" в целях более осмысленного его использования и возможной модификации. Видимо, для решения этой "сверхзадачи" следует, основываясь на специализированных библиотеках, пытаться осуществить следующие шаги. С помощью ПС ТРП проанализировать в автоматизированном режиме имеющийся код и документацию и извлечь из представленных источников схему радикалов, соответствующую ПС ТСМ и включающую все модели, ради которых писался код (т. е. восстановить модели, утраченные в условиях ТСМ). Представить схему так, как это принято в ТРП — она должна содержать схемы-цели, схемы-ресурсы и т.п. Выявить и исправить ошибки (в ТСМ, как известно, "любая программа имеет ошибки"). Оптимизировать полученную схему радикалов. Сгенерировать необходимые коды. Задача эта, как представляется, крайне сложная, в том числе в силу частого наличия в ПС ТСМ даже не темно-серых, но практически черных модулей.

**Тестовые испытания.** Традиционные тестовые испытания не только не исключаются, но являются необходимыми, однако все более настоятельно ставятся вопросы о создании строгих математических моделей, которые позволяли бы в автоматическом режиме проводить такие испытания и адекватно оценивать их результаты [3, с. 11]. Представляется, что ТРП и, в частности, метод синтеза уникума — это продвижение в этом направлении.

**Программное обеспечение информационной безопасности.** Недостатки и проблемы здесь аналогичны недостаткам и проблемам других ПС СС [3, с. 14]. В ТРП информационная безопасность рассматривается как часть ИСБ, и подход к созданию и сопровождению программного обеспечения ИСБ в целом аналогичен подходу, применяемому для любого ПС СС.

**Наука, искусство и производство программ.** Предлагаемую ТРП можно было бы охарактеризовать как переход от программирования как искусства к программированию как науке и производству. Здесь, однако, необходимо уточнение. Зададим следующий вопрос — а является ли искусством программирование при ТСМ? Вряд ли. Точнее, в отдельных случаях — да, но в целом — определенно нет. Скорее этот стиль можно охарактеризовать как прагматический, причем не в лучшем смысле этого слова, по причине традиционной для ТСМ сосредоточенности на локальных, сиюминутных задачах с особым вниманием к тонкостям оформления и прохождения документации. В итоге сам термин "программирование как искусство" приобретает иронический оттенок.

На первый взгляд ТРП, требуя от программиста целенаправленного конструирования программы из библиотечных радикалов с пошаговой верификацией, лишает его свободы творчества (на самом деле, свободы мнимой, характерной для ТСМ). Но это лишь на первый взгляд. На самом деле, ТРП вовсе не отбирает у программиста возможность творить. Если программисту не хватает библиотечных радикалов, он вполне может ввести в библиотеку удобные ему **виртуальные радикалы**, обладающие необходимыми ему свойствами (системами контейнеров), но (пока) не реализованные, и конструировать целевые уникумы с их помощью. При этом инструментальное ПС ТРП никогда "не забудет", что все осуществляемые таким образом построения — это пока лишь гипотеза, "дом, возводимый на песке", и что виртуальные радикалы необходимо будет, в конце концов, заменить реальными — реализованными (реализуемыми). Свобода творчества в ТРП вполне доступна (для ее обеспечения предоставляются самые широкие возможности), имеет характер временно отложенной ответственности, о которой "не удастся забыть", а ТРП с этой точки зрения — это переход от примитивного в своей основе прагматизма ТСМ к науке, настоящему, ответственному искусству программирования и высококультурному индустриальному производству ПС.

**Впереди программного средства.** Для успешного создания, эксплуатации и модификации ПС СС необходимо находиться "впереди" "проживающего" свой ЖЦ ПС. Необходимо постоянно готовиться и быть готовым к тому, что пока от ПС не требуется, но возможно потребуется в ближайшем или достаточно отдаленном будущем. Предлагаемая ТРП предусматривает средства для автоматизации такой работы. Это — формализм схем радикалов, библиотеки радикалов и метод синтеза целевых уникумов (метод решения штатных и нештатных задач). Средства ТРП предназначены не только для решения текущих задач. С их помощью необходимо постоянно "работать на будущее" — осуществлять, как говорят военные, "рекогносцировку местности" — самым тщательным образом изучать проблемную область, выявлять и анализировать тенденции, возможные варианты развития ситуации. Необходимо прогнозировать и экспериментировать. По результатам такой работы необходимо пополнять библиотеки радикалов, создавая, таким образом, "заделы" — основу для будущих возможных решений на "опасных" направлениях.

## Заключение

В данной статье кратко рассмотрена господствующая в настоящее время технология решения задач на всех этапах ЖЦ ПС, которую автор именует технологией серых модулей (ТСМ). Эта технология является самым существенным препятствием для дальнейшего развития программирования, для решения проблем сложности и верификации ПС, а также для развития информатики вообще. Учитывая все возрастающее значение программно-технических средств для сложных систем, можно сказать, что ТСМ оказывает силь-

нейшее отрицательное воздействие на состояние проблемной области СС в целом.

Как представляется автору, выход из создавшейся ситуации заключается в разработке и постепенном вводе в эксплуатацию новой, научно обоснованной технологии — технологии радикального программирования (ТРП), лишенной недостатков ТСМ и имеющей самые существенные практические преимущества.

Определяющим фактором в сложившейся ситуации является человеческий фактор. С одной стороны, ТРП, в своем развитии, безусловно, вызовет сопротивление ТСМ, причем как явное, так и скрытое. Главная причина в ставшей обычной для многих системе ролей и связей, возникшей и отшлифованной в условиях ТСМ. Как, например, отказаться от роли мастера и знатока, чуть ли не единственного реально доступного в конкретной ситуации специалиста, который может решать задачи, принадлежащие определенному классу? Или как допустить саму мысль об отчуждаемости ПС и возможности его модифицирования пользователем? Для действующей технологии все это совершенно неприемлемо. С другой стороны, решение проблем, накопившихся при использовании ТСМ, уже никак нельзя откладывать. Эти проблемы касаются очень многих, а с учетом стремительно возрастающего значения ПТС для СС, окружающих нас "со всех сторон", — всех. Позиции ТСМ находятся в остром противоречии со следующими актуальными требованиями: приемлемого для практики решения проблем сложности и верификации ПС, обеспечения беспрепятственно оперативного доступа к знаниям о проблемных областях (разумеется, согласно имеющимся правам доступа) и перехода к индустриальному производству ПС. Предлагаемая ТРП сконцентрирована на решении именно этих проблем.

Задача создания и развития ТРП масштабная, обещающая вобрать в себя и естественным образом объединить множество на первый взгляд разнородных направлений исследовательских и практических работ. Она потребует участия большого коллектива квалифицированных специалистов самых разных специальностей.

Технология радикального программирования выявляет цель и суть программирования (и моделирования) в самом широком их понимании — необходимость контролируемого и верифицируемого построения единой развивающейся избыточной моделирующей среды мира СС, обладающей интеллектуальными возможностями. Идеи новой технологии являются естественными для мира СС и сложных ПС. Эти идеи, а также актуальность и жизненная необходимость делают предлагаемую технологию привлекательной для молодых специалистов, а также для творческих специалистов всех возрастов. Таким образом, человеческий фактор, противодействующий принципиальным нововведениям в рамках ТСМ, может существенно способствовать прогрессивным изменениям, действуя "на стороне" ТРП.

В проекте, на основе среды радикалов можно обоснованно рассчитывать на объединение квалифицированных и творческих специалистов самых разных специальностей и на их плодотворную совместную работу. Для специалистов будет ценно ощущение принадлежности большому коллективу, в котором обеспечивается объединение их усилий при работе над важным проектом, служащим решению принципиальных проблем программирования и информатики. Благодаря концепции среды радикалов и формализму схем радикалов — единой основе технологии радикального программирования — проекты и работы будет удобно планировать, организовывать и контролировать, обеспечивая взаимопонимание и взаимодействие между специалистами, в том числе удаленными друг от друга. Привлечение специалистов может осуществляться в разных и удобных формах. С учетом актуальности, работы по новой технологии обещают стать перспективными как для руководителей, так и для исполнителей.

Развитие ТРП приведет к самому существенному повышению статуса математиков и программистов.

**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. **Boehm B.** A view of 20th and 21st Century Software Engineering. Los Angeles: University of Southern California, University Park Campus, 2006.

3. **Васенин В. А.** Модернизации экономики и новые аспекты инженерии программ // Программная инженерия. 2012. № 2. С. 2—17.

4. **Чечкин А. В., Пирогов М. В**. Метод интеллектуализации критических систем с использованием таблиц радикалов // Нейрокомпьютеры: разработка, применение. 2012. № 2. С. 3—12.

5. **Чечкин А. В.** Математическая информатика. М.: Наука, 1991. 412 с.

6. **Соболева Т. С., Чечкин А. В.** Дискретная математика. М.: Издательский центр Академия, 2006. 256 с.

7. **Чечкин А. В.** Обеспечение информационно-системной безопасности сложной системы на основе среды нейрорадикалов ее проблемной области // Нейрокомпьютеры: разработка, применение. 2008. № 7. С. 6—11.

8. **Чечкин А. В., Пирогов М. В.** Активаторы и регуляторы среды радикалов // Нейрокомпьютеры: разработка, применение. 2012. № 5. С. 3—8.

9. **Чечкин А. В. Евграфов А. Е., Рожков В. В., Лощенков В. И., Пирогов М. В.** Применение схем радикалов для описания проблемной области автоматизированного комплекса планирования и управления // Информационно-измерительные и управляющие системы. 2009. Т. 7. № 3. С. 5—11.

УДК 519.256

**А. А. Кузьмин**, студент, Московский физико-технический институт,
**В. В. Стрижов**, канд. физ.-мат. наук, науч. сотр.,
Вычислительный центр им. А. А. Дородницына РАН, г. Москва,
e-mail: strijov@ccas.ru

# Проверка адекватности тематических моделей коллекции документов[1]

*Рассматривается коллекция документов с экспертной тематической моделью. Для проверки адекватности экспертной модели предлагается построить алгоритмическую модель путем иерархической кластеризации коллекции текстов агломеративным и дивизимным способами. Определяется степень несоответствия экспертной модели и предлагаемой. Сравнивается качество моделей, полученных с помощью агломеративного и дивизимного алгоритмов. Визуализируются отличия полученной модели от экспертной.*

***Ключевые слова:*** *коллекция документов, тематические модели, иерархические модели, кластеризация*

## Введение

Перед программным комитетом конференции с большим числом участников встает задача проверки корректности построения иерархической модели тезисов конференции. В силу большого числа экспертов, субъективности экспертной кластеризации и отсутствия эталонной модели, оценить качество экспертной иерархической модели сложно. Поэтому предлагается построить иерархическую модель коллекции тезисов, основанную на их терминологическом сходстве, и сравнить результат с экспертной моделью. Для построения модели предлагается использовать метрический алгоритм кластеризации.

Существует два основных типа алгоритмов построения иерархической системы кластеров [1]. Дивизимные (нисходящие) алгоритмы на каждом шаге разбивают имеющиеся кластеры на более мелкие. Агломеративные (восходящие) алгоритмы на каждом шаге объединяют имеющиеся кластеры в более крупные.

В данной работе для построения тематических моделей используются и дивизимные и агломеративные алгоритмы. При построении иерархической модели проводится "жесткая" кластеризация, согласно которой объект принадлежит только к одному из кластеров, так как каждый документ может принадлежать только одной теме, что соответствует правилам проведения конференции. Требуется построить иерархическую модель, сохранив ее схожесть с экспертной, и выделить набор документов, которые попадают в кластеры, отличающиеся от экспертных. Функционал сходства моделей задан числом различий между построенной моделью и моделью, заданной экспертно.

Для построения модели предлагается использовать методы иерархической кластеризации. Для кластеризации множества документов вводится функция расстояния [2]. Каждому документу ставится в соответствие метрический вектор, содержащий информацию о словарном составе этого документа. Для того чтобы оставить в документе только те слова, которые несут информацию о его сходстве и отличии от других документов, предварительно проводится предобработка документов. Слова приводятся к начальной лексической форме [3], удаляются знаки препинания. Исключаются слова, встречающиеся малое число раз, а также слова, встречающиеся в большинстве документов. Для этого используется критерий TF—IDF

[1] Работа выполнена при поддержке Министерства образования и науки РФ в рамках Государственного контракта 07.524.11.4002.

(англ. TF — *Term Frequency*, IDF — *Inverse Document Frequency*) [4], а также словарь стоп-слов. Затем из оставшихся слов составляется словарь, а документы представляются в виде "мешков слов" [5].

После отсева неинформативных слов из словаря и документов, каждому документу ставится в соответствие булевый вектор. В работе [3] сравниваются способы построения вектора — описания документа.

Кластеризация текстов при построении каждого уровня иерархической модели может проводиться как с помощью метрических алгоритмов кластеризации, например, K-means [6], FOREL [7], C-means [8], STOLP [9], FRiS-STOLP [10], BoostML, DANN [11] и других [12, 13], так и с помощью вероятностных методов [14], например, с помощью вероятностного латентного семантического анализа (англ. PLSA — *probabilistic latent semantic analysis*) [5], или латентного размещения Дирихле (англ. LDA — *Latent Dirichlet Allocation*) [2]. В данной работе применяется метрический алгоритм кластеризации, подробно описанный в работе [3].

Результаты работы дивизимного и агломеративного алгоритмов иллюстрируются примером тематической иерархической кластеризации тезисов конференции "European Conference on Operational Research, EURO—2012". Предложенная модель сравнивается с экспертной.

## Постановка задачи

Пусть $W = \{w_1, ..., w_n\}$ — заданное множество слов (словарь), где $n$ — число слов в словаре. Документом $d$ из коллекции $D$ назовем неупорядоченное множество слов из $W$, $d = \{w_j\}$, где $j \in \{1, ..., n\}$.

Поставим в соответствие каждому документу $d$ его описание — булевый вектор $\mathbf{x}$ размера $n$ следующим образом. Если слово $w_j$ из словаря $W$ встретилось в документе $d_s$ хотя бы раз, то $x_{s,j} = 1$, иначе $x_{s,j} = 0$. Получим матрицу $\mathbf{X}$ "объект—признак", где каждая строка $\mathbf{x}_s = [x_{s,1}, ..., x_{s,n}]$ — признаковое описание документа $d_s$:

$$\mathbf{X} = \begin{pmatrix} x_{1,1} & \dots & x_{1,n} \\ \dots & \dots & \dots \\ x_{|D|,1} & \dots & x_{|D|,n} \end{pmatrix}. \qquad (1)$$

В качестве функции расстояния ρ между документами возьмем евклидову метрику:

$$\rho(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{j=1}^{n} \left(x_j - x'_j\right)^2}. \qquad (2)$$

Представим экспертную иерархическую модель в виде дерева (рис. 1). Глубину дерева обозначим $h$. Уровнем $l$ иерархии назовем множество всех узлов дерева, находящихся на глубине $l$. Каждый внутренний узел дерева обозначим $c_{i,l}$, где $l$ — уровень, к которому принадлежит данный узел, а $i$ — номер этого узла среди узлов на данной глубине. Документы $d$ являются листами этого дерева и имеют уровень $h$. Под кластером $c_{i,l}$ будем понимать множество дочерних элементов данного узла, $c_{i,l} = \{c_{i,l+1}\}$ для некоторых $i$. Будем говорить, что документ $d_s$ принадлежит кластеру $c_{i,l}$, если путь до данного документа от вершины проходит через узел $c_{i,l}$.

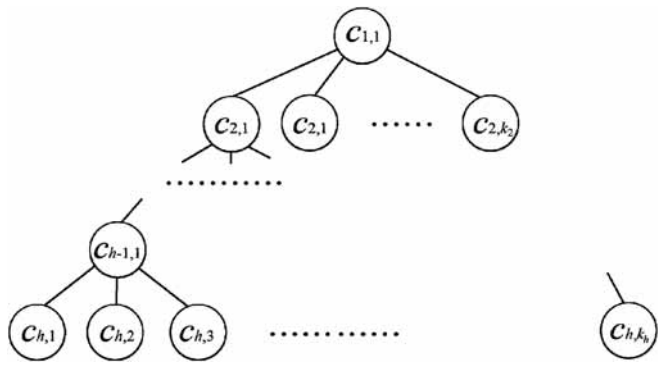



**Рис. 1. Иерархическое представление тематической модели**

Зададим путь от корня $c_{1,1}$ до документа $d_s$ следующим образом: $\mathbf{g}_h^s = \left(c^s_{i_1,1}, ..., c^s_{i_2,l}, ..., c^s_{i_h,h}\right)$, где $c^s_{i_l,l}$ — порядковый номер узла уровня $l$, через который проходит путь. Символом $\mathbf{g}_l^s$ обозначим путь, построенный по экспертной модели, а $\hat{\mathbf{g}}_l^s$ — путь, построенный по алгоритмической модели. Обозначим $\boldsymbol{\mu}\left(\mathbf{g}_l^s\right)$ — координаты центра кластера, соответствующего документу $s$ на уровне $l$ иерархической модели. Зададим функционал сходства экспертной и построенной иерархических моделей:

$$\sum_{s=1}^{|D|} \sum_{l=1}^{h} \left[\mathbf{g}_l^s \neq \hat{\mathbf{g}}_l^s\right] \rho\left(\boldsymbol{\mu}\left(\mathbf{g}_l^s\right), \boldsymbol{\mu}\left(\hat{\mathbf{g}}_l^s\right)\right) \to \min, \qquad (3)$$

где индикаторная функция $\left[\mathbf{g}_l^s \neq \hat{\mathbf{g}}_l^s\right]$ равна 1, если все элементы векторов $\mathbf{g}_l^s$ и $\hat{\mathbf{g}}_l^s$ совпадают. В противном случае функция равна 0. Оценивая степень схожести кластеров, основываясь на расстоянии ρ между их центрами $\boldsymbol{\mu}$, будем штрафовать алгоритмическую модель тем сильнее, чем больше расстояние между центром кластера $\boldsymbol{\mu}\left(\hat{\mathbf{g}}_l^s\right)$, к которому алгоритм отнес документ, и центром кластера $\boldsymbol{\mu}(\mathbf{g}_l^s)$, к которому эксперт отнес документ на каждом уровне $l$ иерархической модели. Задача построения иерархической модели сводится к задаче минимизации функционала (3).

## Описание алгоритма

Опишем алгоритм тематической кластеризации для случая единственного уровня иерархии тематической модели. Алгоритм итеративно находит центры

кластеров $\boldsymbol{\mu}$, после чего относит документ $\mathbf{x}_s$ к тому или иному кластеру исходя из принципа "ближайшего соседа". Зафиксируем число кластеров $C$, равное числу тем в экспертной модели. Центр $\hat{\boldsymbol{\mu}}_y$ кластера с номером $y$ вычислим по следующей формуле:

$$\hat{\boldsymbol{\mu}}_y = \frac{\sum_{s=1}^{|D|} [y_s = y]\mathbf{x}_s}{\sum_{s=1}^{|D|} [y_s = y]}. \quad (4)$$

Здесь индикаторная функция $[y_s = y]$ равна 1, если условие равенства выполняется. В противном случае функция равна 0. Вычисленный центр кластера является "центром масс" множества документов, отнесенных к этому кластеру. В качестве начального приближения положений центров кластеров выберем центры экспертных кластеров $\hat{\boldsymbol{\mu}}_y = \boldsymbol{\mu}_y$, $y \in \{1, ..., C\}$. Отнесем каждый документ с номером $s$ к кластеру с номером $\hat{y}_s$:

$$\hat{y}_s = \arg\min_{y \in \{1,...,C\}} \rho(\mathbf{x}_s, \boldsymbol{\mu}_y),\ s = 1, ..., |D|. \quad (5)$$

Каждая итерация алгоритма кластеризации состоит из двух шагов. На первом шаге координаты центров кластеров $\hat{\boldsymbol{\mu}}_y$ пересчитываются по формуле (4). В качестве функции $\rho$ расстояния между документами используется евклидова метрика (2). На втором шаге для каждого описания документа $\mathbf{x}_s$ с номером $s$ решается задача (5), после чего документ $d_s$ с описанием $\mathbf{x}_s$ относится к кластеру с номером $\hat{y}_s$. Алгоритм останавливается, когда координаты $\boldsymbol{\mu}_y$ на очередной итерации не поменяются после пересчета (4) для всех $y \in \{1, ..., C\}$.

Для построения иерархической модели предлагается использовать два альтернативных алгоритма: дивизимный и агломеративный, а затем сравнить полученные результаты с помощью функционала (3).

Обозначим $\hat{\boldsymbol{\mu}}(c_{i,l})$ — вектор координат центра кластера, соответствующего узлу дерева $c_{i,l}$ (рис. 1). Через $\boldsymbol{\mu}(c_{i,l})$ будем обозначать вектор координат центра кластера $c_{i,l}$, полученный из экспертной модели. Значения $\hat{\boldsymbol{\mu}}(c_{i,l})$ можно найти, используя в формуле (4) в качестве множества $\{\mathbf{x}_s\}$, $s \in \{1, ..., |D|\}$ множество центров кластеров $\boldsymbol{\mu}(c_{i,l-1})$, $i \in \{1, ..., k_{l+1}\}$, а $y_s$ будет определяться по формуле (5), где множество $y$ будет множеством центров кластеров уровня $l$. На уровне $l = h$ объектами будут описания $\mathbf{x}_s$ документов $d_s$, чьи векторы координат есть строки матрицы $\mathbf{X}$ (1).

**Агломеративный алгоритм.** Алгоритм строит уровни иерархической модели, обходя дерево снизу вверх. На шаге $l$ центры кластеров $\{\hat{\boldsymbol{\mu}}_{h-l+1}\}$ уровня $h - l$ рассматриваются в качестве объектов. К ним применяется описанный выше алгоритм. В качестве начального приближения центров кластеров $\hat{\boldsymbol{\mu}}_y$ берутся координаты центров экспертных $\boldsymbol{\mu}_y$ кластеров уровня $h - l$. Таким образом, на шаге $l$ мы строим уровень иерархии $h - l$.

**Дивизимный алгоритм.** Алгоритм строит уровни иерархической модели, идя по дереву сверху вниз. На шаге $l$ для каждого кластера $c_{i,l}$, $i \in \{1...k_l\}$ применяется описанный выше алгоритм, использующий в качестве объектов множество $\{\mathbf{x}_s \mid y_s = i\}$, где $y_s$ находится из условия (5). В качестве начального приближения положения центров кластеров берутся центры экспертных кластеров уровня $l + 1$, лежащих внутри кластера $c_{i,l}$, $\hat{\boldsymbol{\mu}}_{m,\,h-l+1}\ \{m \mid c_{m,l} \in c_{i,l}\}$. В итоге каждый кластер $c_{i,l}$ делится на более мелкие кластеры:

$$c_{i,l} = \coprod_{m \in c_{i,l}} c_{m,l+1}.$$

На первом шаге в роли кластеров выступает единственный кластер $c_{1,1}$, состоящий из всех документов коллекции.

## Вычислительный эксперимент

Построим иерархические тематические модели для коллекции тезисов конференции "European Conference on Operational Research, EURO—2012" с помощью агломеративного и дивизимного алгоритмов. Данная коллекция из 1342 документов иерархически разбивается на области (*area*) и направления (*stream*), как показано на рис. 2. Сравним полученные модели с экспертной моделью. Более адекватной считается алгоритмическая модель, которая имеет меньшее значение функционала ошибки (3).

Найдем число несоответствий между моделями: число документов, которые относятся экспертной алгоритмической моделью к различным областям, число документов, которые относятся экспертной алгоритмической моделью к различным направлениям и число документов, которые относятся моделями к разным областям и разным направлениям. Полученные результаты для обоих алгоритмов приводятся в таблице.

**Значение функционала ошибки для разных способов построения иерархической модели**

| Вид алгоритма | Число несоответствий | | | Значение функционала (3) |
|---|---|---|---|---|
| | на уровне "Область" | на уровне "Направление" | на обоих уровнях | |
| Дивизимный | 554 | 555 | 550 | 1700 |
| Агломеративный | 342 | 208 | 182 | 500 |

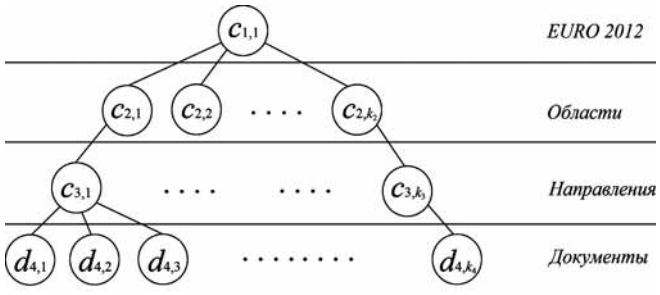



**Рис. 2. Пример иерархического представления тематической модели**

На рис. 3 (см. вторую сторону обложки) показано распределение документов по областям. Число столбцов гистограммы совпадает с числом кластеров уровня "область". Каждому документу присваивается кодовый цвет области, к которой он отнесен. Высота части столбца одного цвета показывает число документов. Рис. 3, *а* построен по экспертной кластеризации, поэтому каждый столбец содержит документы только одного цвета. Этот цвет одновременно является цветом области (кластера) с номером данного столбца. Результат работы дивизимного алгоритма изображен на рис. 3, *б*, а результат работы агломеративного алгоритма на рис. 3, *в*.

Номера областей (столбцов гистограмм) во всех трех графиках фиксированы. Алгоритические модели перераспределяют документы по темам, вследствие чего каждый столбец содержит документы различных кодовых цветов. Чем выше высота сегмента фиксированного цвета в столбце, тем больше документов из соответствующей области там содержится. На рис. 3, *в* видно, что в результате использования агломеративного алгоритма в каждый столбец гистограммы попадают документы из небольшого числа других столбцов. При кластеризации в столбцы попадают не отдельные документы из других столбцов, а целые кластеры уровня "Направление". Таким образом, изменение модели носит более систематический характер и отличия алгоритмической модели от экспертной легче интерпретировать. Вместе с тем при использовании агломеративного алгоритма появляются два кластера (6-й и 13-й), к которым вместо части документов из других областей были отнесены сразу целые направления, в результате чего число документов в этих областях стало отличаться от экспертного почти в два раза.

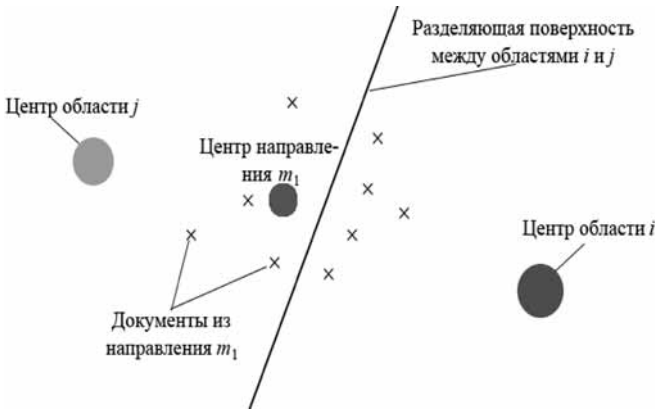



**Рис. 5. Направление, попавшее близко к разделяющей поверхности между областями**

Перераспределение документов по кластерам уровня "Направление" после применения агломеративного алгоритма показано на рис. 4. Для каждого документа координатой по оси абсцисс является номер направления, к которому отнес алгоритм данный документ, по оси ординат — его экспертное направление. Радиус круга определяется числом документов, попавших в эту точку. Наибольшему кругу соответствует число документов, равное 28. Заметим, что основное число документов попадает на диагональ. Таким образом, у большинства документов номер экспертного направления совпадает с алгоритмическим, что соответствует нашему требованию к схожести построенной метрической модели с экспертной.

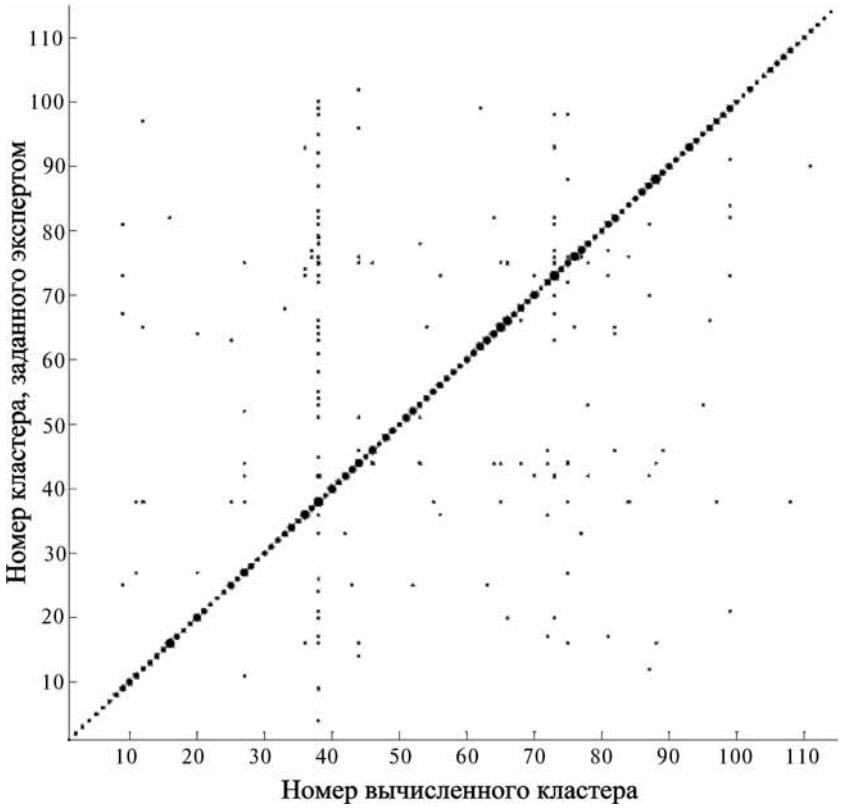



**Рис. 4. Перераспределение документов по направлениям для модели, построенной агломеративным алгоритмом**

При сравнении с экспертной моделью результата работы дивизимного алгоритма видно, что почти для всех документов, у которых не совпала область, не совпало и направление 0. Документ с номером *s*, попавший в *i*-ю область, уже не может попасть в направление из *j*-й области, даже если с метрической точки зрения центр этого направления ближе к документу *s*, чем все центры направлений из *i*-й области (см. рис. 2). Поэтому документы из тех направлений, центры которых попадают на разделяющую поверхность между различными областями, относятся к разным областям (рис. 5). Однако так

как направление может быть отнесено только к одной области, то на всех остальных документах, попавших в другую область, будет допущена ошибка.

Таким образом, результат получается лучше, если применять агломеративный алгоритм. В этом случае сначала учитывается терминологическое сходство документов при объединении их в направления. Затем каждое направление относится к области, учитывая уже усредненную информацию о терминологии всех документов, попавших в данное направление.

Создание иерархической тематической модели с использованием информации о соответствии документов как областям, так и направлениям, позволяет повысить адекватность модели (сходство алгоритмической модели с экспертной) на 15 % по сравнению с адекватностью "плоской" алгоритмической модели, построенной в работе [3] с использованием только информации о соответствии документов областям.

## Заключение

Исследовалась задача проверки адекватности тематической модели. Экспертная модель сравнивалась с моделью, построенной метрическими алгоритмами кластеризации. Были предложены различные метрические способы построения моделей, схожих с экспертной. Наилучшим из предложенных способов является использование агломеративного алгоритма построения иерархической тематической модели. На практике при построении экспертных тематических моделей результаты алгоритмической кластеризации используются для выявления возможных противоречий в действиях экспертов.

**Список литературы**


1. **Воронцов К. В., Потапенко А. А.** Робастные разреженные вероятностные тематические модели // Интеллектуализация обработки информации. Доклады 9-й международной конференции. 2012. С. 605—608.
2. **Blei D. M., Ng A. Y., Jordan M. I.** Latent dirichlet allocation // Journal of Machine Learning Research. 2003. Vol. 3. P. 993—1022.
3. **Кузьмин А. А., Адуенко А. А., Стрижов В. В.** Выбор признаков и оптимизация метрики при кластеризации коллекции документов // Известия Тульского государственного университета, Естественные науки. 2012. Вып. 3. С. 119—131.
4. **Blei D. M., Lafferty J. D.** Topic Models. Text Mining: Classifcation, Clustering, and Applications. Chapman & Hall. CRC Press, 2009.
5. **Hofmann T.** Probabilistic latent semantic indexing // Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval. New York: ACM, 1999. P. 50—57.
6. **Hartigan J. A., Wong M. A.** Algorithm as 136: A k-means clustering algorithm // Applied statistics. 1978. Vol. 28. P. 100—108.
7. **Загоруйко Н. Г., Елкина В. Н., Лбов Г. С.** Алгоритмы обнаружения эмпирических закономерностей. Новосибирск: Наука, 1985.
8. **Pal N. R., Bezdek J. C.** On cluster validity for the fuzzy c-means model // IEEE Transactions on Fuzzy Systems. 1995. Vol. 3. P. 370—379.
9. **Загоруйко Н. Г.** Прикладные методы анализа данных и знаний. Новосибирск: ИМ СО РАН, 1999. 270 с.
10. **Борисова И. А.** Использование fris-функции для построения решающего правила и выбора признаков (задача комбинированного типа dx) // Знания. Онтологии. Теории. Материалы Всероссийской Конференции. Новосибирск. 2007. Т. 1. С. 37—44.
11. **Tibshirani R., Hastie T.** Discriminative adaptive nearest neighbor classification // IEEE transactions on pattern analysis and machine intelligence. 1996. Vol. 18. N 6. P. 607—616.
12. **Peng J., Gunopulos D., Domenciconi C.** An adaptive metric machine for pattern classification // Advances in Neural Information Processing Systems 13. MIT Press, 2000. P. 458—464.
13. **Zagoruiko N. G.** Methods of recognition based on the function of rival similarity // Pattern recognition and image analysis. 2008. Vol. 18(1). P. 1—6.
14. **Daud A., Li J., Zhou L., Muhammad F.** Knowledge discovery through directed probabilistic topic models // Frontiers of computer science in China. 2010. Vol. 4(2). P. 280—301.


## *ИНФОРМАЦИЯ*

### 11th IEEE EAST—WEST DESIGN & TEST SYMPOSIUM (EWDTS 2013) состоится в г. Ростов-на-Дону, Россия, 27—30 сентября 2013 г.

Цель симпозиума — расширение международного сотрудничества и обмен опытом между ведущими учеными Западной и Восточной Европы, Северной Америки и других стран в области автоматизации проектирования, тестирования и верификации электронных компонентов и систем. Симпозиум проводится, как правило, в странах бассейнов Черного и Балтийского морей, Центральной Азии. Оргкомитет приглашает ученых, аспирантов и студентов принять участие в работе EWDTS'13.

Симпозиум будет проходить в Ростове-на-Дону — крупнейшем научном и образовательном центре Южного федерального округа России

**Важные даты**: **Срок подачи докладов**: 15 июля 2013 г.
**Итоги рецензирования**: 1 августа 2013 г.

**Детальная информация и регистрация докладов: http://www.ewdtest.com/conf**

**Адрес оргкомитета:** Проф. Владимир Хаханов, кафедра Автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники, пр. Ленина 14, Харьков, 61166, Украина.
Тел.: +380-57-702-13-26, E-mail: hahanov@kture.kharkov.ua

УДК 519.612.631

**К. Ф. Иванова,** канд. техн. наук, докторант, СПбГУ,
e-mail: klara.i2010@yandex.ru

# Точечно-интервальный метод оценки чувствительности численного решения уравнений эллиптического типа

*Предлагается оценка погрешности вектора неизвестных систем линейных алгебраических уравнений, полученных при численной аппроксимации уравнений Пуассона. Анализ чувствительности систем уравнений основывается на априорной оценке их реакции на вариабельность входных данных, выраженных интервальными величинами, соответствующими погрешностям краевых условий и функции источника. Предлагаемый метод определения чувствительности решения выявляет экстремальные отклонения компонент вектора неизвестных в наиболее уязвимых точках расчетной области. Результаты численных расчетов обусловливают априорную оценку погрешности решения систем уравнений, возникающих в строительной механике, приборостроении, распределении тепловых и электромагнитных полей и многих других задачах, и гарантируют надежность, необходимую для предотвращения опасных ситуаций при работе объектов, реализованных на расчетных программных модулях, построенных на идеологии точечно-интервального подхода.*

***Ключевые слова:*** *уравнения Пуассона, моделирование, аппроксимация, интервальные величины, чувствительность, оценка погрешности, экстремальные отклонения*

Необходимость в математическом описании процессов в виде уравнений в частных производных Лапласа и Пуассона очень часто возникает при исследовании физических и технических объектов, исследовании напряженно-деформированных состояний сооружений и конструкций, при моделировании стационарных электромагнитных и тепловых полей, а также в целом ряде других задач. Важность решения краевых задач для уравнений Пуассона определяется как при решении самостоятельных задач, так и в случае, когда они служат средством решения более общих краевых задач для уравнений эллиптического типа и различных нестационарных систем. Как правило, решение большинства этих задач не представляется возможным без применения численных методов. Основными из них являются метод конечных разностей (МКР) [1] и метод конечных элементов (МКЭ) [2, 3], развитый для решения задач более сложной структуры. Отправными для развития МКЭ стали фундаментальные результаты, связанные с исследованием сходимости и устойчивости конечно-разностных схем [1, 4]. Дискретизация дифференциальных уравнений, заменяющих задачу отыскания непрерывной функции на конечное число ее приближенных значений в отдельных точках-узлах, представляет собой систему алгебраических уравнений (СЛАУ) с разреженными матрицами.

Исследование, результаты которого представлены далее, не касается вопросов оценки точности аппроксимации, корректности, сходимости и погрешности применяемого численного метода расчета. Рассматривается задача оценки погрешности решения, непосредственно вызванная изменениями (вариациями) входных данных. Следует отметить, что общепринятая апостериорная оценка погрешности решения базируется во многом на оценках, идентичных тем, которые используются для оценки корректности метода,

а именно в своих оценках устойчивости апостериорная оценка использует усредненные оценки по всему ансамблю компонент вектора неизвестных. Так, при моделировании крупных объектов, основанных на алгоритмах расчета МКЭ, большие математические пакеты позволяют в широком диапазоне менять краевые и начальные условия системы, геометрические и физические входные параметры. Однако несмотря на все многообразие математических программ как отечественных (ПИОНЕР, ЛИРА, МИРАЖ, МОРЕ, ПАРСЕК, ПАРУС, НЕДРА и др.), так и зарубежных (GEOSLOPE, PLAXIS, ADVENTURE, GEOFEM, ANSYS, ADINA, ABAQUS, ALGOR, COSMOS, FINEL, NASTRAN, MARC), в настоящее время они не позволяют получить достаточно точные оценки погрешности отдельных компонент вектора решения, отличного от осредненных оценок по нормам и по числу обусловленности. Недооценка погрешности численного решения системы, основанная на норме, как мере ее чувствительности к возмущениям, не дает исчерпывающую информацию об отдельных точках или части области, где ошибка может быть достаточно велика. Это может привести к срыву и даже к авариям в процессе эксплуатации расчетного объекта.

Предлагаемый в работе подход определения погрешности решения, с одной стороны, опирается на "знаковую методику" [5, 6], а с другой стороны, позволяет решить так называемую "внешнюю" задачу интервальной системы линейных алгебраических уравнений (ИСЛАУ) [7, 8]. Разработанный алгоритм оценки максимальных отклонений решения под влиянием погрешностей входных воздействий является новым и ранее не применялся для интервальной оценки предельных значений объединенного решения ИСЛАУ, полученных для уравнений в частных производных. Этот алгоритм очень прост в программной реализации и позволяет гарантировать ошибку решения, ограниченную одновременно сверху и снизу [9].

Покомпонентная оценка погрешности вектора неизвестных определяет меру чувствительности алгебраической системы на изменения внешних воздействий и неизбежную для любой задачи неопределенность параметров краевых условий и флуктуаций источника поля или приложенных сил. Расчет погрешности численного решения уравнений Пуассона при дискретизации двумя параллельными методами предоставляет дополнительные свидетельства справедливости предлагаемой методики предельных интервальных оценок.

## Метод оценки погрешности решения интервальных линейных алгебраических уравнений

В общем виде решением интервальной системы линейных алгебраических уравнений

$$\mathbf{Ax} = \mathbf{b} \qquad (1)$$

является покоординатная оценка множества решений $\mathbf{x} = (x_i) \in \mathbf{IR}^n$, где $\mathbf{A} = (\mathbf{a}_{ij}) \in \mathbf{IR}^{n\times n}$ — интервальная матрица, $\mathbf{b} = (b_i) \in \mathbf{IR}^n$ — интервальный вектор правой части. При независимом варьировании элементов матрицы и правых частей системы (1) внешняя покоординатная оценка множества решений сводится к нахождению разброса решений содержащихся в ней всех точечных $(n \times n)$-систем с $A \in \mathbf{A}$ и $b \in \mathbf{b}$:

$$\Sigma = \{x \in \mathbf{R}^n | (\exists A \in \mathbf{A})(\exists b \in \mathbf{b})(Ax = b)\},$$

образованного решениями всех точечных систем $Ax = b$.

Задача (1) соответствует интервальной форме задачи о чувствительности для линейной системы, когда вариации входных данных и оценка решения задаются в интервальном виде. Как следует из многочисленных публикаций по интервальному анализу, решение этой задачи обладает вычислительной сложностью и является NP-трудной задачей во многих практически важных случаях [6, 10].

Обратимся к рассмотрению интервальной задачи (1), в которой интервальная система понимается как формальная запись двух точечных систем

$$Ax = \underline{b} \text{ и } Ax = \overline{b}, \qquad (2)$$

решение которых определяет предельные значения компонент вектора **x** как оболочки множества решений системы (2).

В системе (2) матрица $A = (a_{ij}) \in R^{n\times n}$, $i = \overline{1, n}$; $j = \overline{1, n}$ является точечной и квадратной, вектор $\mathbf{b} = (\boldsymbol{b}_i) \in \mathbf{IR}^n$ содержит вариации элементов в пределах заданных интервалов:

$$\mathbf{b} = [\underline{b}, \overline{b}] = \{\mathbf{b}; \underline{b} \leqslant b \leqslant \overline{b}\}, \ \underline{b} = b - \Delta, \ \overline{b} = b + \Delta, \quad (3)$$

с компонентами

$$b_i = [b_i - \Delta_i, \ b_i + \Delta_i], \quad i = \overline{1, n},$$

где $\underline{b}$, $\overline{b}$ — точечные векторы, полученные по знаковой методике, а b — вектор измерений (или вектор параметров), попадающий в ограниченный интервал $[b - \Delta, b + \Delta]$ [6]. В зависимости от знака относительных погрешностей $\varepsilon_i$, определяющих абсолютные отклонения $\Delta_i = \varepsilon_i b_i$ $(i = \overline{1, n})$, формируются компоненты вектора $\overline{b}$ или $\underline{b}$, доставляющие максимальное или минимальное значение расширенному определителю матрицы A. Решением сформированных точечных систем (2) с правой частью, представленной (3), является разброс полученных решений, из которых осуществляется выборка компонент вектора **x**, получивших наибольшие и наименьшие значения. Понятно, что компонента вектора относительной погрешности $\varepsilon_i$, доставляющая максимум вектору $\overline{b}$, будучи с противоположным знаком, формирует симметричную компоненту вектора $\underline{b}$, так что в конечном счете $\underline{b} = -\overline{b}$.

Процедура выбора знака компоненты $\varepsilon_i$ определяется из условий

$$\begin{cases} if\ \ b_i \mathrm{Ad}_{ij} > 0, & \varepsilon_i > 0, \\ if\ \ b_i \mathrm{Ad}_{ij} < 0, & \varepsilon_i < 0, \quad i, j = \overline{1, n}, \\ if\ \ b_i \mathrm{Ad}_{ij} = 0, & \text{не определено.} \end{cases} \tag{4}$$

$\mathrm{Ad}_{ij}$ — алгебраическое дополнение к элементу $a_{ij}$ матрицы A.

Таким образом, оценка снизу и сверху интервального решения проводится в два этапа. На первом этапе находится решение двух точечных систем $\underline{b}$, $\overline{b}$ интервального вектора **b**, сконструированных по правилам знаковой методики, основанной на анализе максимального линейного приращения детерминантов матриц системы (2). В соответствии с этим подходом знак для относительной погрешности ε при первых двух условиях в (4) очевиден. В третьем случае знак ε выбирается исходя из значения максимального (минимального) детерминанта при последовательной подстановке компонент $b_i - \varepsilon_i b_i$ и $b_i + \varepsilon_i b_i$ в расширенную матрицу A. На втором этапе проводится выборка предельных значений для компонент уже вычисленного вектора $\mathbf{x} = [\underline{x}, \overline{x}]$ и оценки $\min\{x_k \mid x \in \mathbf{x}\}$ снизу и $\max\{x_k \mid x \in \mathbf{x}\}$ сверху ($k = \overline{1, n}$) полученного интервального решения. Вычисление относительных погрешностей для каждой предельной компоненты $x_k$ осуществляется сравнением со сходной компонентой вектора неизвестных, полученного из решения невозмущенной системы Ax = b.

Как известно, неоднородная линейная система имеет решение при условии невырожденности или неособенности матрицы, т. е. при условии $|\mathrm{A}| \neq 0$. Только в этом случае решение может быть получено через обратную матрицу:

$$\mathrm{x} = \mathrm{A}^{-1}\mathrm{b}. \tag{5}$$

Это условие является необходимым и для интервальных матриц, когда каждая из точечных матриц, принадлежащих интервальной, должна оставаться невырожденной и удовлетворять выражению (5).

Предложенный подход покомпонентной оценки погрешности решения линейной системы в корне отличается от общепринятой усредненной оценки по числу обусловленности $\mu(\mathrm{A}) = \|\mathrm{A}\|\|\mathrm{A}^{-1}\|$ и нормам функций, входящих в выражение для относительной погрешности функций. В случае вариации компонент правой части выражения в отсутствие возмущенной матрицы системы относительная погрешность решения определяется известным выражением

$$\frac{\|\Delta \mathrm{x}\|}{\|\mathrm{x}\|} \leqslant \mu(\mathrm{A}) \frac{\|\Delta \mathrm{b}\|}{\|\mathrm{b}\|}, \tag{6}$$

где Δ характеризует разность между возмущенными и исходными матрицами и векторами [11, 12]. Выбор типа норм в выражении (6) в общем случае несущественен, так как во всех случаях они определяют модульные усредненные величины и характеризуют абсолютные погрешности отклонений. Значение μ(A) рассматривается как мера чувствительности системы к погрешностям данных и характеризует меру близости A к вырожденной матрице. Считается, что чем больше число обусловленности, тем сильнее влияние погрешности входных данных на конечный результат и матрица является плохо обусловленной при $\mu \gg 1$.

Существенно, что оценка погрешности по числу обусловленности является не только усредненной по всему ансамблю переменных, но и в большинстве случаев завышенной. В реальных ситуациях при моделировании физического процесса возникает серьезная необходимость в покомпонентной оценке погрешности вектора неизвестных, в том числе получаемого при дискретизации разностными операторами дифференциальных.

## Дискретизация двухмерного уравнения Пуассона

Простейшим уравнением в частных производных эллиптического типа являются уравнения Пуассона и Лапласа (при равной нулю правой части).

В случае двухмерного уравнения Пуассона существует зависимость искомого решения только от двух координат — $x$ и $y$. При этом считается, что решение уравнения Пуассона существует и удовлетворяет условиям однозначности, конечности и непрерывности [13]. Уравнение Пуассона в однородной изотропной среде с единичными коэффициентами перед вторыми производными имеет вид

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y), \tag{7}$$

в прямоугольной области, когда граница области совпадает с узлами сетки

$$\Pi_{a,b} \equiv [0 \times a] \times [0 \times b] \equiv$$
$$\equiv \{(x, y) \in \mathrm{R}^2 : 0 \leqslant x \leqslant a,\ 0 \leqslant y \leqslant b\},$$

$$u \in \mathrm{C}^1([0, a] \times [0, b]).$$

На отдельных сторонах $\Pi_{a,b}$ могут быть заданы условия разных видов — Дирихле или Неймана, обеспечивающие единственность решения.

Под условиями Дирихле и Неймана понимается задание функции $u$ или ее производной в каждой точке границы:

$$\begin{gathered} u(0, y) = \varphi 1(y),\ u(a, y) = \varphi 2(y), \\ u(x, 0) = \psi 1(x),\ u(x, b) = \psi 2(x) \text{ или} \\ \frac{\partial u(0, y)}{\partial y} = \varphi 3(y),\ \frac{\partial u(a, y)}{\partial y} = \varphi 4(y), \\ \frac{\partial u(x, 0)}{\partial x} = \psi 3(x),\ \frac{\partial u(x, b)}{\partial x} = \psi 4(x). \end{gathered} \tag{8}$$

В дальнейшем, если не оговорено противное, все фигурирующие функции по умолчанию предполагаются непрерывными и имеющими непрерывные производные соответствующих порядков. Численное решение краевых задач для эллиптических уравнений в простейших случаях, когда решение во всей рассматриваемой области меняется более или менее равномерно, а сама область не имеет узких "перешейков", осуществляется с помощью разностных схем.

Дискретизация дифференциального оператора (7) и граничных условий системы (8) с шагом $h$ и $l$ в направлениях $x$ и $y$ методом конечных разностей проводится с точностью аппроксимации $O(h^2 + l^2)$.

Для использования в дальнейшем идеологии знакового подхода оценки погрешности решения расчет ограничивается квадратной областью с $a = b$, шагами $l = h$ и числом узлов дискретизации $m$ в обоих направлениях.

В качестве краевых можно взять любые четыре условия из (8). В качестве шаблона принимается пятиточечная разностная схема с вертикальным перечислением узлов сетки, в которых последовательно вычисляются значения сеточной функции $u_h$, предложенная в работе [14].

Тогда дискретизация уравнения (7) для внутренних точек области приводит к системе уравнений

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} = h^2 f_{i,j}, \quad i, j = 2, \ldots, m-1, \tag{9}$$

где $f_{i,j}$ — значение функции $f(x, y)$ в точке с координатами $(x_i, y_j)$.

Дискретизация граничных условий Дирихле и Неймана (8) приводит к системе

$$\begin{cases} u_{1,j} = \varphi 1(y_j),\ u_{m,j} = \varphi 2(y_j),\ u_{i,1} = \varphi 3(x_i),\ u_{i,m} = \varphi 4(x_i), \\ (u_{2,j} - u_{1,j})/h = \psi 1(y_j),\ (u_{m,j} - u_{m-1,j})/h = \psi 2(y_j), \\ (u_{i,2} - u_{i,1})/h = \psi 3(x_i),\ (u_{i,m} - u_{i,m-1})/h = \psi 4(x_i). \end{cases} \tag{10}$$

Полученная алгебраическая система конечно-разностных уравнений (9)—(10) может быть записана аналогично системе (1)

$$\mathrm{A}\mathbf{u} = \mathbf{b} \tag{11}$$

с точечной матрицей $\mathrm{A} \in \mathrm{R}^{n \times n}$ и интервальным вектором $\mathbf{b} \in \mathbf{IR}^n$, размерность которых ($n$) определяется числом узлов дискретизации $m$ на регулярных сетках в направлении независимых координат уравнения, так что $n = m^2$. В силу этого для нахождения решения сеточной функции $\mathbf{u} = (\boldsymbol{u}_i) \in \mathbf{IR}^n$, $i = \overline{1, n}$ получается система алгебраических линейных уравнений большой размерности специального вида. В отличие от общего вида заполненных матриц, матрица конечно-разностных уравнений (11) является сильно разреженной и самосопряженной, в то время как обратная матрица является уже не разреженной, а заполненной. Это принципиально при выборе знака относительной погрешности $\varepsilon$ при определении $\mathbf{u}_h$ [15]. Однако следует отметить, что невырожденная вещественная симметричная матрица системы A, как показано в работе [16], является плохо обусловленной, с числом обусловленности, которое растет с уменьшением шага $h$, $\mu(\mathrm{A}) = \mathrm{tg}^{-2}\frac{\pi h}{2} \approx \frac{4}{\pi^2 h^2}$. При выборе шаге дискретизации для показательных примеров можно сопоставлять эту оценку с числом обусловленности $\mu(\mathrm{A})$ из неравенства (6).

## Примеры оценки погрешности численного решения уравнений Пуассона

Матрицы A, полученные дискретизацией дифференциального оператора конечно-разностным методом для граничных условий Дирихле и Неймана, символически представлены на рис. 1. Ненулевые коэффициенты для элементов матрицы обозначены символами "∗"и "✡". Увеличение размерности матрицы не изменяет ее блочного представления, и она остается пятидиагональной.

Рассчитанные сеточные предельные отклонения вектора неизвестных позволяют проанализировать, насколько чувствительна система к погрешностям правых частей уравнения (9) и граничных условий (10).

При использовании метода конечных элементов поиск функции из дифференциального уравнения с соответствующими граничными условиями заменяется поиском конечного числа ее приближенных значений в отдельных точках-узлах типовой подобласти, именуемой конечным элементом. Дискретная модель строится на множестве кусочно-непрерывных функций, определенных на конечном числе этих подобластей и в конечном виде представляет собой систему алгебраических уравнений $\mathrm{K}\mathbf{u} = \mathbf{Q}$. Глобальная функция жесткости K аналогична матрице A, а глобальный вектор сил $\mathbf{Q}$ соответствует вектору источника или нагрузки $\mathbf{b}$ (11). Расположение узлов в матрице, полученное, в частности, из триангуляции плоской ограниченной области, определяет совсем другую структуру элементов матрицы K по сравнению с матрицей A. При-

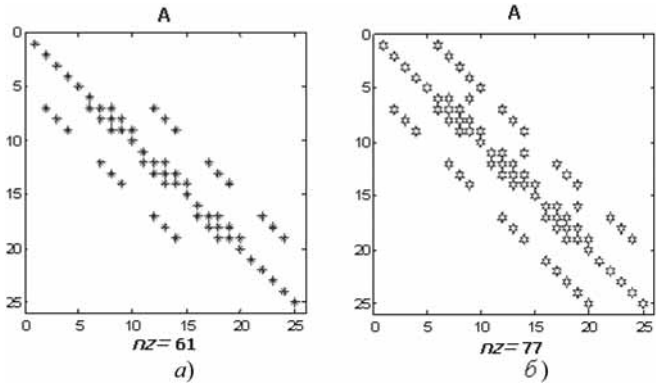
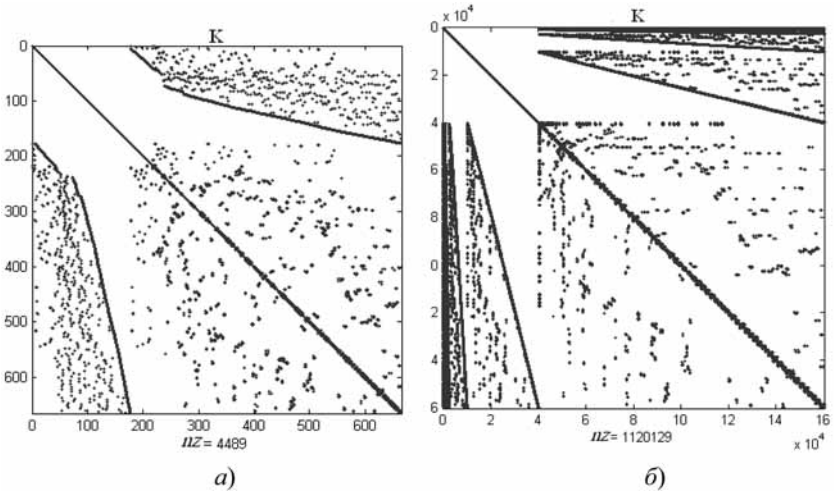



**Рис. 1. Матрицы, полученные дискретизацией дифференциального оператора конечно-разностным методом для граничных условий Дирихле (*а*) и Неймана (*б*):**
*nz* — число узлов с ненулевыми элементами матриц




*а)* *б)*

**Рис. 2. Матрицы, полученные для разного числа триангуляций области с разным числом ненулевых элементов:**
*а* — $nz$ = 4489; *б* — $nz$ = 1 120 129

меры таких матриц, полученных для разного числа триангуляций области и соответствующего числа узлов $nz$, представлены на рис. 2.

Применение к задаче (7—8) метода конечных элементов проводится с использованием высокоуровневой системы MATLAB с помощью приложения PDE Toolbox (Partial Differential Equation), обеспечивающего решение дифференциальных уравнений в частных производных методом конечных элементов в двухмерной постановке. Приложение включает: графический интерфейс; инструментальные средства для задания формы уравнений и граничных условий, средства для визуализации полученного решения и его анимации. Приложение PDE Toolbox использует проекционную формулировку метода конечных элементов и представляет собой набор специальных функций, написанных на языке MATLAB. При запуске рабочего окна разворачивается графический интерфейс, позволяющий экспортировать граничные условия и правую часть уравнения.

Необходимо отметить, что для оценки погрешности решения МКЭ возможности высокоуровневой системы PDE Toolbox позволили встроить формализованные поправки в граничные условия и в правую часть уравнения (7) в соответствии со знаковой методикой в интервальном варианте. Эта процедура является удобной формой исследования погрешности решения в случае применения предлагаемой методики для моделирования достаточно сложных физических задач в выпуклой многоугольной области в пространстве кусочно-линейных конечных элементов [2].

Определение интервальных отклонений решения под влиянием погрешности входных данных параллельно рассчитано двумя методами — МКР и МКЭ на сетках различной геометрии и дискретности. Оценки проводились для решения следующих задач:

- функция источника невозмущенная, граничные условия — интервальные величины;
- функция источника — интервальная величина, граничные условия постоянны;
- функция источника и граничные условия задаются интервально.

**Пример 1.** Расчет погрешности численного решения уравнения Пуассона МКР и МКЭ в целях сравнения полученных результатов для одинакового числа узлов сеточной области, равного $nz$ = 10 000, и разным количеством триангуляций сеточной области. Рассматриваются граничные условия Дирихле и Неймана.

Функция источника $f = 1$.

Граничные условия Дирихле

$$u|_{0,y} = 10\sin(2\pi y)(1 \pm \varepsilon\,\mathrm{sign}(\sin(2\pi y)));$$

$$u|_{a,y} = 10\sin(2\pi y)(1 \pm \varepsilon\,\mathrm{sign}(\sin(2\pi y)));$$

$$u|_{0,x} = 10\sin(2\pi x)(1 \pm \varepsilon\,\mathrm{sign}(\sin(2\pi x)));$$

$$u|_{b,x} = 10\sin(2\pi x)(1 \pm \varepsilon\,\mathrm{sign}(\sin(2\pi x)));$$

$$\varepsilon = 0{,}1.$$

Результаты расчета графически визуализированы на рисунках.

На рис. 3 (см. вторую сторону обложки), показано решение для поверхности $u(x, y)$ в объемном и контурном представлениях МКЭ с числом узлов аппроксимации $nz$ = 10 145.

На рис. 4 (см. третью сторону обложки) показано решение для поверхности $u(x, y)$ в объемном и контурном представлениях МКР с числом узлов дискретизации $nz$ = 10 000.

Визуальная картина для поверхности абсолютных отклонений $u(x, y)$ и соответствующие ей потенциальные уровни, полученные с начальной относительной погрешностью $\varepsilon$ = 10 %, представлены на рис. 5, см. третью сторону обложки.

На рис. 6 показаны два изображения поверхности абсолютных отклонений решения влево, полученные МКЭ для различного числа узлов сеточной области $np$ = 177 и $np$ = 40 257. Визуально рисунки 6, *а*, *б* и 5, *а* идентичны между собой, они характеризуют поверхности отклонений от невозмущенного решения, полученные МКР и МКЭ. На рис. 7, *а* (см. третью сторону обложки) показаны абсолютные отклонения решения в векторной форме. По модулю абсолютные максимальные погрешности для знакопеременного и постоянного значения $\varepsilon$ совпадают. Однако в первом случае возникает значительное увеличение числа компонент большей амплитуды по сравнению со второй вариацией. Именно в этом случае для отклонений возмущенного решения уместна оценка по норме: $\mathrm{norm}(u_{\max} - u) = 7{,}3095$; $\mathrm{norm}(u_c - u) = 3{,}7963$, где $u$ — невозмущенное решение; $u_{\max}$ — возмущенное решение при знакопеременном $\tilde{\varepsilon}$; $u_c$ — возмущенное решение, полученное при постоянном значении $\varepsilon$; число точек дискретизации в квадратной области $N$ = 121. При замене в примере 1 граничных условий Дирихле на условия Ней-

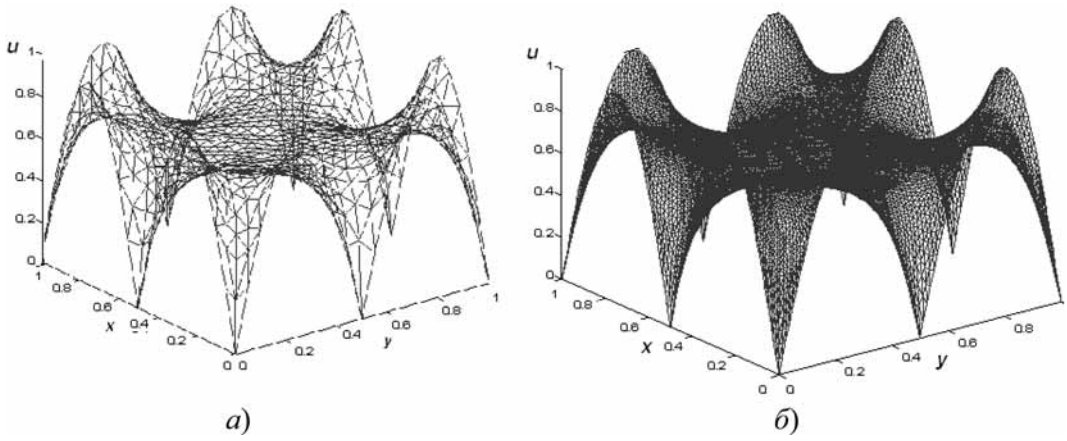



**Рис. 6. Абсолютные отклонения от невозмущенного решения, полученные по знаковой методике МКЭ для разного числа узлов:** *а — пр* = 177; *б — пр* = 40 257

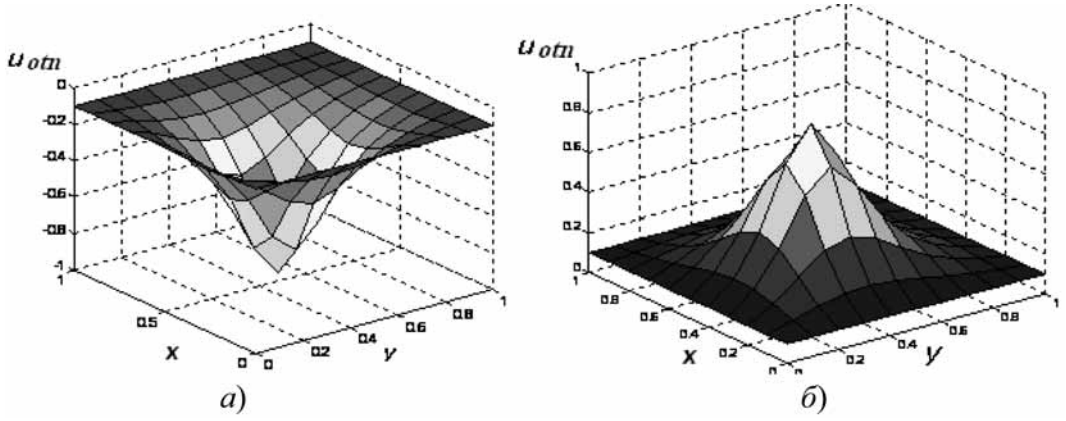



**Рис. 9. Предельные относительные интервальные отклонения решения влево (*а*) и вправо (*б*) от невозмущенной поверхности *u*(*x*, *y*)**

мана абсолютные отклонения компонент вектора неизвестных будут иметь вид, представленный на рис. 7, *б* (см. третью сторону обложки).

В данном случае наблюдается значительное превышение покомпонентных отклонений при задании $\tilde{\varepsilon}$ по сравнению с ε, и отношение максимальных отклонений составляет $k = 1{,}37$ при нормах отклонений, равных 6,9872 и 3,3936 соответственно. Полагая значение правой части $f = 10$, т. е. в 10 больше первоначально заданной, получаем для отношения максимальных отклонений $k = 1{,}59$. Этот факт означает, что при увеличении значения абсолютной величины функции правой части, при прочих равных условиях, возрастает превышение максимального отношения $k$.

Оценки покомпонентных отклонений относительных погрешностей вектора решения могут стать значительными и даже бесконечно большими вследствие нулевых значений невозмущенного решения в отдельных узлах области. При этом, однако, они не могут считаться в данной постановке задачи репрезентативными.

**Пример 2.** Оценивается влияние на погрешность решения флуктуации интервальной функции правой части $f(x, y)$, зависящей от обеих координат $x$ и $y$: $f = 10(\sin(3/2\pi y) + \sin(3/2\pi x))(1 \pm \varepsilon\,\mathrm{sign}(10(\sin(3/2\pi y) + \sin(3/2\pi x)))$. На границах квадрата краевые условия Дирихле задаются постоянными:

$$u|_{0,y} = 1;\ u|_{a,y} = 1;\ u|_{0,x} = 1;\ u|_{b,x} = 1.$$

Общее число сеточных узлов матрицы $nz = 14\,641$; $\varepsilon = 0{,}1$.

На рис. 8, *а* (см. третью сторону обложки) показано невозмущенное решение $u(x, y)$ в объемном представлении, полученное без погрешности входных данных. Поверхность $u(x, y)$ ограничена снизу плоскостью $u = 0{,}2$. Это позволяет с конечной точностью оценить предельные относительные погрешности решения при флуктуациях правой части уравнения.

Максимальные отклонения относительной погрешности решения влево и вправо от среднего значения $u(x, y)$ достигают значения 0,8 при знакопеременном $\tilde{\varepsilon}$ (рис. 9). Отметим, что при постоянных относительных погрешностях ε они остаются на уровне заданного значения, равного 0,1 (рис. 8, *б*, см. третью сторону обложки). Поверхности на рис. 9 отражают интервальные относительные отклонения решения, которые отсчитываются от уровня $u = \pm 0{,}1$. На рис. 10, *а* (см. четвертую сторону обложки) показаны абсолютные погрешности в векторном представлении реше-

ния, откуда получена оценка для максимальных интервальных отклонений, равная $|u_{\max} - u| = 0{,}18$, которая почти вдвое превосходит погрешность решения при постоянном $\varepsilon = 0{,}1$.

На рис. 10, *б* (см. четвертую сторону обложки) показаны расчеты относительных значений погрешности в векторной форме решения. Их графическое представление подтверждает характер погрешностей, полученных для абсолютных и относительных отклонений в виде поверхностей в объемном представлении. На графиках показаны одновременно все сравнительные отклонения, рассчитанные при постоянных и переменных ε. В то же время оценки, проводимые по нормам и числу обусловленности, вообще не отвечают реальным значениям. Так, формула, определяемая неравенством (9), при возмущенной правой части, дает оценку, равную 144,57, что является завышением на несколько порядков по сравнению с полученным предельным значением $u_{\text{отн}} = 0{,}8$. Это превышение возникает, прежде всего, за счет очень большого числа обусловленности μ(A).

Рост погрешности относительных и абсолютных погрешностей решения будет увеличиваться с ростом размерности системы, что имеет место и для роста погрешности за счет округлений при счете. Однако, как показывает пример 1, в пределах доступных для решения в MATLAB систем уравнений больших размерностей этот рост оказывается не столь значительным и может быть спрогнозирован.

**Пример 3.** Рассматривается совместное влияние на погрешность решения интервальных вариаций погрешности правой части и краевых условий.

$$f(x, y) = 10\left((-1)^{(m+1)y} + (-1)^{(m+1)x}\right); \qquad (12)$$

$$u|_{0,y} = \cos(\pi/3y);\ u|_{a,y} = \cos(\pi/3y);$$
$$u|_{0,x} = \cos(\pi/3x);\ u|_{x,b} = \cos(\pi/3x);$$
$$\varepsilon = 0{,}1.$$

В качестве источника выбрана функция, которая в заданной области имеет переменное по знаку значение от точки к точке. На границах области заданы гармонические функции. Такая ситуация может существовать при возникновении флуктуации электрического потенциала. На рис. 11 показано поле источника правой части уравнения (12). На рис. 12 представлена поверхность невозмущенного решения $u(x, y)$. Воздействие на погрешность решения переменного поля источника представлено на рис. 13—15.

На рис. 13 (см. четвертую сторону обложки) представлены отклонения поверхностей влево и вправо от невозмущенной поверхности для постоянного и переменного значений ε. Нормы отклонений при обоих значениях ε составляют 1,2614 и 0,9456 соответственно, что объясняется значительным превышением погрешности практически каждой компоненты для возмущенного решения $\tilde{u}$ по сравнению с $u_c$.

Относительные поверхности отклонения решения визуализированы на рис. 14. Как видно из отклонений поверхностей, их значение в отдельных точках области возрастает почти в 2 раза по сравнению с заданным значением погрешности входных данных.

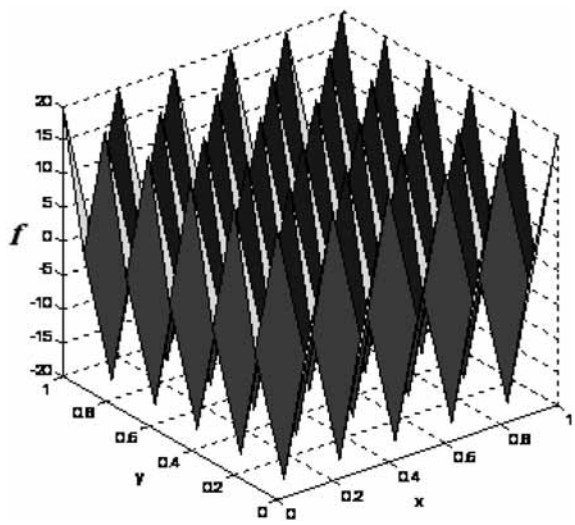



**Рис. 11. Объемное поле $f(x, y)$ правой части уравнения (12)**

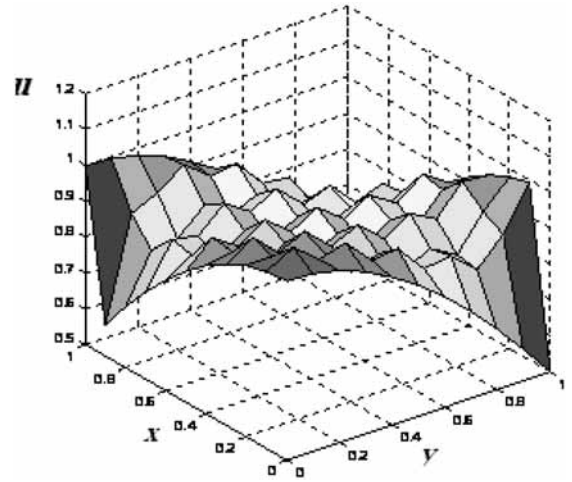
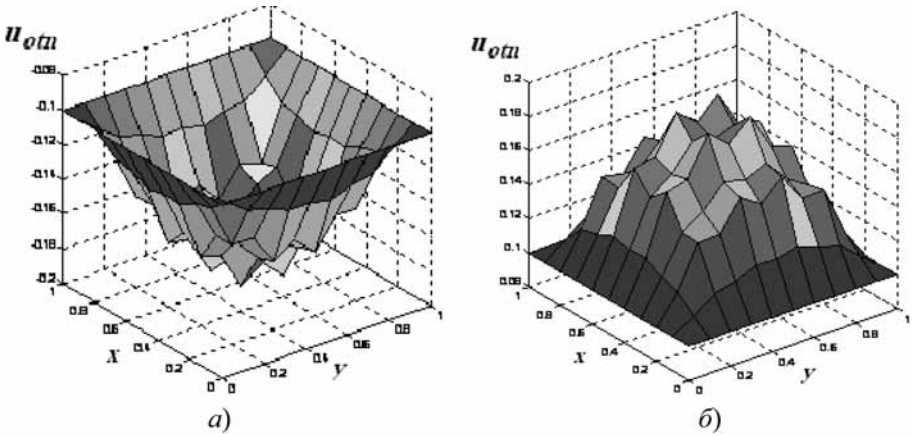



**Рис. 12. Поверхность невозмущенного решения**

Отклонения влево и вправо от невозмущенной поверхности при переменном и постоянном ε в векторном варианте показаны на рис. 15 (см. четвертую сторону обложки). Графики наглядно подтверждают их аналогию с объемным решением, представленным на рис. 13—14. На рис. 15, *а* изображены абсолютные отклонения решения, обозначенные цифрой *1* — при постоянных ε и цифрой *2* — при знакопеременных $\tilde{\varepsilon}$. На рис. 15, *б* теми же цифрами отмечены относительные отклонения решения в процентах.

Рассмотренные примеры убедительно показывают, что предельные интервальные отклонения численного решения уравнения Пуассона, при возмущении и вариациях входных данных, в той или иной степени зависят от функции граничных условий, правых частей и размерности системы. Независимо от выбора метода расчета, погрешности, выявленные с помощью предложенной




*а)* *б)*

**Рис. 14. Относительные интервальные отклонения поверхностей от невозмущенного решения при знакопеременном ε:**
*а* — влево; *б* — вправо

оценочной методики, оказываются во многих случаях значительно больше тех значений, которые получаются без учета знакового подхода к заданию погрешности входных данных. Интервальный характер начальных погрешностей позволяет оценить симметричные отклонения решения влево и вправо, полезные для пространственного анализа решения задачи.

Погрешность систем очень больших размерностей должна быть исследована с помощью усовершенствованных программных пакетов. Современная устойчивость решения к вариациям внешних воздействий, не проверенная на предельные отклонения, может быть объяснена тем, что в реальных условиях далеко не все компоненты внешних параметров имеют ощутимые погрешности, влияющие на конечный результат, что снижает риск большого сбоя. Однако при возникновении таких погрешностей, что в принципе неизбежно при решении любой сложной задачи, пренебрежение заявленными оценками может привести к опасным ситуациям при реализации таких моделей на практике.

## Выводы

Проведен анализ оценки погрешностей численных решений уравнений Пуассона, полученных МКЭ и МКР с использованием знаковой методики и интервального представления граничных условий и функции источника.

На простых примерах наглядно продемонстрировано, что относительная погрешность решения может возрастать в несколько раз и в значительной степени зависит от вида функции правых частей и граничных условий.

Показано, что предложенная методика оценки предельных погрешностей, получаемых при моделировании физических процессов уравнениями в частных производных, позволяет априорно определить погрешность решения не по оценкам усредненного критерия всего ансамбля компонент вектора решения, а по отклонениям значений функции в отдельных узлах области. Неучтенные погрешности отдельных компонент вектора неизвестных могут стать источниками опасных ситуаций и даже аварий при практической реализации объекта.

Использование системы MATLAB позволяет быстро и эффективно на разработанных алгоритмах реализовать программы, совместимые с языком и интерпретаторами среды MATLAB. В их число входит составление функций и *m*-файлов для численного решения уравнений, использование приложения PDE Toolbox для решения задач МКЭ и графической визуализации решений задач, что определяет дополнительную наглядность проводимого анализа [17, 18].

**Список литературы**


1. **Самарский А. А., Андреев Б. В.** Разностные методы для эллиптических уравнений. М.: Наука, 1976. 352 с.
2. **Ильин В. П.** Методы конечных разностей и конечных объемов для эллиптических уравнений. Новосибирск: Изд-во Ин-та математики, 2000. 345 с.
3. **Зенкевич О., Морган К.** Конечные элементы и аппроксимация. М.: Мир, 1986. 318 с.
4. **Тихонов А. Н., Арсенин В. Я.** Методы решения некорректных задач. Издание третье. М.: Наука, 1986. 287 с.
5. **Петров Ю. П.** Как обеспечить надежность решения систем уравнений. СПб.: БХВ-СПб, 2009. 172 с.
6. **Иванова К. Ф.** Знаковый подход к оценке решения интервальных линейных систем // Информационные технологии. 2012. № 9. С. 46—52.
7. **Шарый С. П.** Конечномерный интервальный анализ. Новосибирск: XYZ, 2007. 700 с.
8. **Rohn J.** A Handbook of Results on Interval Linear Problems. URL: http://uivtx.cs.cas.cz/~rohn/publist/!handbook.pdf
9. **Шарый С. П.** Алгебраический подход во "внешней задаче" для интервальных линейных систем // Фундаментальная и прикладная математика. 2002. Т. 8. № 2. С. 567—610.
10. **Шокин Ю. И.** Интервальные вычисления и перспективы их развития в контексте кодо-логической эволюции // Наукові праці Донецького національного технічного університету. Серія "Проблеми моделювання та автоматизації проектування" (МАП-2010). 2010. Вип. 8 (168). URL: http://masters.donntu.edu.ua/2012/fknt/kulibaba/library/7.htm
11. **Ильин В. А., Позняк Э. Г.** Линейная алгебра. М.: Наука, 1978. 302 с.
12. **Форсайт Дж., Молер К.** Численное решение систем линейных алгебраических уравнений (пер. с англ.). М.: Мир, 1969. 168 с.
13. **Михлин С. Г.** Курс математической физики. М.: Наука, 1968. 568 с.
14. **Рындин Е. А.** Методы решения задач математической физики. Электронный учебник. URL: http://www.twirpx.com/file/ 79613/
15. **Иванова К. Ф.** Оценка погрешности численного решения уравнений Пуассона под воздействием флуктуаций входных параметров в среде Matlab. СПб.: ПИЯФ РАН, 2010. 34 с.
16. **Молчанова Л. А.** Разностные методы решения дифференциальных уравнений. Дальневосточный государственный университет. Уч. пособие. Электронная книга. URL: http://group22x.narod.ru/calculus/LECTION.PDF
17. **Иванова К. Ф.** Свидетельство РФ о государственной регистрации программы для ЭВМ № 2011611641 "Программный комплекс оценки экстремальных значений погрешности выходных характеристик решения стационарных задач строительной механики, тепловых и электромагнитных процессов" (ПКОПП). 2011.
18. **Иванова К. Ф.** Свидетельство РФ о государственной регистрации программы для ЭВМ № 201261281. "Программа оценки решения моделей физических и технических задач на основе расчета интервальных линейных систем" (POPRIZ). 2012.

УДК 004.056.53

**А. В. Шумилин**, вед. науч. сотр., ОАО "НПО РусБИТех",
e-mail: shums@inbox.ru

# Подход к оцениванию влияния средств разграничения доступа к данным на производительность реляционных СУБД

*Рассмотрен подход к оцениванию влияния средств разграничения доступа к данным на производительность реляционных СУБД на примере мандатного разграничения доступа в СУБД отечественной операционной системы специального назначения Astra Linux Special Edition.*

***Ключевые слова:*** *системы управления базами данных, защита информации*

Один из принципов разработки защищенных программных комплексов заключается в том, что защищенность информации, обрабатываемой в них, должна быть обеспечена без снижения их функциональности и производительности [1]. В то же время необходимо понимать, что реализация той или иной модели разграничения доступа в защищенном программном комплексе подразумевает сбор и обработку информации, которая необходима для принятия решения о предоставлении доступа. Последнее обстоятельство привносит дополнительные временные потери на различных этапах обработки данных.

При создании защищенных программных комплексов, в частности СУБД, необходимо уделять внимание минимизации временных потерь, вносимых механизмами разграничения доступа (МРД) в штатном режиме работы. Для достижения указанной цели в первую очередь следует проанализировать влияние на производительность СУБД используемых в ней МРД. Подобная информация позволяет определить особенности проектирования и эксплуатации БД, а также дать рекомендации по совершенствованию используемых СУБД и МРД.

Для определения влияния МРД на производительность СУБД необходимо провести анализ ее архитектуры с учетом особенностей реализации и функционирования МРД в процессе обработки данных.

Архитектура современных СУБД подробно рассмотрена в работах [2, 3]. Большинство используемых в настоящее время СУБД построено по двухзвенной или многозвенной клиент-серверной архитектуре, одним из преимуществ которой является централизация управления доступом. На рисунке с небольшими изменениями отображена рассмотренная в работе [4] схема обработки SQL-запроса.

Как видно на схеме, сервер СУБД содержит:

— тракт обработки запросов пользователя, включающий синтаксический анализатор (1), компилятор (2), планировщик (3) и исполнитель запросов (4) (цифрами обозначены этапы обработки запроса);

— механизмы хранения пользовательских данных и информации об их структуре (метаданные);

— менеджеры ресурсов и хранения данных, отвечающие за распределение буферов хранения, вычислительных устройств и других, необходимых для обработки ресурсов сервера.

Рассмотренная схема сильно упрощена, так как не включает механизмы управления транзакциями и блокировками, а также не содержит средств обеспечения целостности данных и отказоустойчивости. Перечисленные механизмы являются неотъемлемыми элементами современных СУБД, обеспечиваются системными средствами ядра СУБД и, как правило, не содержат МРД. Детальная информация об архитектуре конкретной СУБД может быть получена из анализа ее исходного кода и предоставленной разработчиком документации. В то же время сведения об особенностях реализации и функционирования МРД могут отсутствовать или быть неполными, так как подобная информация, как правило, носит конфиденциальный характер. Например, особенности встраивания и функционирования МРД в реляционных

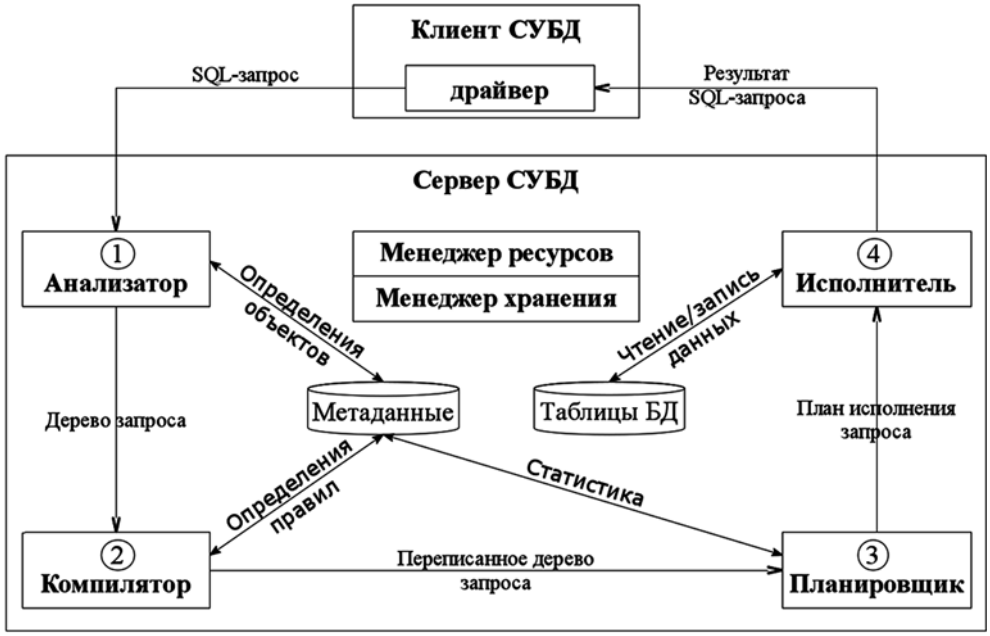



**Схема обработки SQL-запроса**

СУБД рассмотрены в работах [1, 5] для вариантов реализации механизмов мандатного разграничения доступа в СУБД Oracle [1] и в СУБД ОС СН Astra Linux Special Edition [5].

Анализ представленной архитектуры СУБД и особенностей встраивания МРД [5] показывает, что поскольку правила разграничения доступа (ПРД) могут применяться как к объектам в целом, так и при доступе к отдельным записям, все МРД можно разделить на МРД уровня объектов и МРД уровня данных. При этом вносимые этими механизмами потери могут зависеть от запроса пользователя, состава и количества запрашиваемых операций, атрибутов безопасности пользователя и объектов, а также от механизмов доступа и числа вовлеченных в обработку записей. Следует учитывать, что влияние МРД на производительность СУБД определяется не только потерями времени на их применение (действия по сбору и обработке информации, требуемой для принятия решения о предоставлении доступа), но и потерями, связанными с встраиванием (наличием) этих механизмов.

Рассмотрим подробнее характер временных потерь МРД в процессе обработки запроса в СУБД, использующей мандатное разграничение доступа, на примере защищенной СУБД, которая входит в состав операционной системы специального назначения Astra Linux Special Edition [6].

Работа пользователя с СУБД начинается с *установки соединения и создания сеанса пользователя*. При этом проводится сопоставление пользователя с сеансом обработки данных СУБД, что включает в себя инициализацию контекста безопасности пользователя (атрибутов безопасности), который в дальнейшем будет использован МРД. Потери при установке соединения зависят от состава контекста безопасности и способа его получения. Так как работа пользователя с СУБД проводится, как правило, в рамках сессии, основное влияние МРД на производительность проявляется при обработке запросов.

Архитектура реляционной СУБД подразумевает следующие этапы обработки запроса [2—4, 7, 8]:

1. ***Разбор запроса.*** После получения запроса от пользователя осуществляются его проверка, анализ и преобразование во внутреннее представление. Результатом этого этапа является дерево запроса, определяемое синтаксисом языка запросов SQL и содержащее список объектов БД (отношения, атрибуты, операторы, вызовы хранимых процедур и т. п.) с информацией о типах доступа для каждого из объектов и составе требуемого результата.

2. ***Преобразование запроса.*** Проанализированный запрос "переписывается" в соответствии с заданными правилами. Одним из примеров является механизм получения представлений. В этом случае исходный запрос преобразуется в запрос к отношениям, указанным в теле представления. Могут также применяться и дополнительные правила изменения запроса [7].

3. ***Генерация планов исполнения и выбор наиболее оптимального.*** Исходя из полученного на предыдущих этапах описания требуемого результата, планировщик СУБД строит планы его достижения, учитывая затребованные операции, объекты, их атрибуты и доступные для них механизмы доступа. Построенные планы рассматриваются оптимизатором с учетом статистики по числу участвующих в обработке записей, наличия необходимых индексов и особенностей указанных механизмов доступа, после чего выбирается наименее затратный по времени и ресурсам план ис-

полнения запроса. Как правило, именно в этот момент осуществляется применение МРД пользователя к участвующим в запросе объектам БД. Примером подобных МРД (уровня объектов) является штатный механизм дискреционного разграничения доступа СУБД. На этом этапе МРД вносят задержку, связанную с определением необходимости их применения к каждому из объектов и непосредственно применению механизмов разграничения доступа. В этом случае влияние МРД зависит от числа объектов в запросе и не зависит от числа обрабатываемых записей.

4. ***Исполнение выбранного плана запроса.*** На этом этапе осуществляется непосредственное обращение к необходимым страницам индексов и записям участвующих в запросе отношений, выполняются операции соединения, группировки и сортировки, а также выдача полученного результата. При этом могут применяться некоторые виды МРД (уровня данных). Примером подобных МРД является механизм мандатного разграничения доступа в случае хранения метки непосредственно в записях [1, 5]. Такие механизмы встраиваются, как правило, в менеджер хранения данных, и применяются в момент обращения к конкретной записи. Разграничение осуществляется независимо от МРД уровня объектов.

На каждом из этапов могут присутствовать потери времени на регистрацию событий доступа. Состав и детализация событий зависит от используемых МРД и настроек подсистемы регистрации событий [9]. Объем потерь может быть достаточно большим, поскольку сопряжен с получением детальной информации о событии и с фиксацией этой информации в журнале регистраций событий. В связи с этим обстоятельством не рекомендуется осуществлять фиксацию всех успешных попыток доступа, ограничившись фиксацией отказов доступа. В каждом конкретном случае политика регистраций событий должна определяться в соответствии с требованиями, предъявляемыми к защищенной СУБД в составе конкретной автоматизированной системы (АС).

Существуют накладные расходы, связанные с передачей части контекста безопасности пользователя по сети и с чтением с диска атрибутов безопасности объекта и данных. Передача контекста и атрибутов безопасности по сети не зависит от работы с СУБД, так как в случае использования защищенной АС это присуще любому сетевому обмену и реализовано на уровне операционной системы. Потери времени на чтение атрибутов безопасности (метки) данных с диска уже учтены в потерях на уровне данных.

Аналогично запросам пользователя на модификацию данных, запросы модификации метаданных (схемы БД) также подвергаются разграничению доступа. При этом работа МРД осуществляется аналогично, с использованием, например, для мандатного разграничения доступа приведения операций CREATE и ADD к доступу на запись, а ALTER и DROP — к последовательному выполнению доступа на чтение и запись. Далее будут рассматриваться только запросы пользователя на модификацию данных. Предложенный способ определения влияния МРД на производительность СУБД может быть применен и для запросов на модификацию метаданных. Поскольку метаданные также подлежат защите, при работе с ними применяются те же МРД как непосредственно при модификации метаданных, так и при анализе и разборе запроса к данным БД.

Поскольку СУБД представляют собой алгоритмически сложные программные системы, характер влияния МРД на производительность не всегда очевиден. При решении задачи определения влияния МРД на производительность СУБД исследуемую систему можно представить в виде "черного ящика". На его вход подаются воздействия в виде специально подготовленных SQL-запросов, а в качестве результата рассматривается время их исполнения. Точность определения в таком случае зависит от совершенства предполагаемой модели исследуемой системы и качества инструментальных средств.

При анализе влияния МРД на производительность работы реляционной СУБД, использующей мандатное разграничение доступа, на примере выбранной СУБД, можно опираться на следующие предположения по возможным видам влияния.

- Потери создания контекста пользователя — величина, не зависящая от запроса пользователя.
- Потери на встраивание МРД на уровне объектов — величина, пропорциональная числу объектов запроса. К объектам запроса могут относиться и используемые в запросе функции преобразования, сортировки и доступа.
- Потери на применение МРД на уровне объектов — величина, зависящая от реализации функции принятия решения о доступе. Для дискреционного разграничения доступа величина зависит от числа записей в списке контроля доступа, для мандатного — от вида SQL-запроса, так как разные виды SQL-запросов используют разный набор типов доступа, приводимых к операциям чтения и записи [8]. Вклад потерь этого рода зависит от действительного количества применений механизма разграничения доступа.
- Потери на встраивание МРД на уровне данных — величина, зависящая от механизмов хранения и доступа к записям и, в худшем случае (при неиндексированном доступе), пропорциональная числу защищенных записей, содержащихся в объектах, входящих в запрос. Потери этого рода характерны для мандатного разграничения доступа к записям.
- Потери на применение МРД на уровне данных — для мандатного разграничения доступа величина постоянная в расчете на одну запись, так как при принятии решений о доступе используется метка записи и текущий контекст сеанса. Вклад потерь этого рода зависит от действительного количества защищенных записей. Строки индексных и временных отношений, как правило, метками не защищаются.
- Потери на встраивание механизма регистрации событий — величина, зависящая от реализации меха-

низма хранения регистрируемой информации и детализации информации о событии. В общем случае на один объект доступа может существовать как минимум одно событие об успешном доступе и несколько событий отказа в доступе. При этом в журнал регистрации событий может попасть только одно из перечисленных событий. Потери этого рода не зависят от установленной политики регистрации событий.

- Потери на регистрацию событий доступа — величина потерь зависит от числа регистрируемых событий доступа к объектам, что в свою очередь определяется политикой регистрации событий сеанса.

Перечисленные виды потерь могут быть описаны рядом отдельных параметров влияния МРД на производительность. При этом состав и значения параметров могут различаться для разных типов доступа к данным (SQL-запросов select, insert, update и delete), так как логика работы и вовлекаемые МРД при их обработке отличаются. Должны учитываться и параметры потерь при доступе к метаданным, и при выполнении операций объединений и соединений объектов, участвующих в запросе. Полученная таблица параметров влияния МРД на производительность может быть использована в дальнейшем для оценки потерь при исполнении сложных запросов.

Основная задача, возникающая при определении значений параметров временных потерь, заключается в точном определении времени отдельных этапов исполнения запроса. В зависимости от условий проведения анализа, таких как наличие исходных кодов СУБД, встроенных средств профилирования и отладки, возможностей исполнения хранимых процедур и программирования на стороне сервера, можно определить следующие способы получения значений интересующих параметров.

- ***СУБД доступна в исходных кодах.*** В случае наличия исходного кода СУБД и возможности его модификации, измерения могут быть проведены внедрением в код инструментальных средств учета времени исполнения требуемых участков кода. При этом потери времени на работу МРД могут быть учтены с высокой степенью точности. Данный вариант представляется наиболее предпочтительным. К сожалению, не всегда существует возможность получить или модифицировать исходный код СУБД.

- ***СУБД поддерживает средства профилирования.*** Некоторые СУБД обеспечивают возможность разработки подгружаемых на стороне сервера модулей, которым могут быть доступны контрольные точки [7], по которым можно судить о ходе исполнения запроса, в том числе и для измерения времени отдельных этапов. Этот подход также может обеспечить высокую точность определения значений параметров модели в случае достаточности контрольных точек и предоставляемой ими информации.

- ***СУБД поддерживает исполнение хранимых процедур.*** Если СУБД предоставляет возможность создания хранимых процедур для обработки данных, измерения могут быть проведены с помощью языковых средств самой СУБД. Это дает возможность избежать влияния межпроцессного клиент-серверного взаимодействия, которое может осуществляться, в том числе и по сети передачи данных, потери времени в которой сложно учесть. Поскольку большинство СУБД поддерживает исполнение хранимых процедур, рассматриваемый вариант является наиболее применимым. Для корректного определения потерь также необходима функция получения временных меток достаточной точности, которая, как правило, присутствует в наборе функций СУБД, либо может быть реализована с помощью вызовов внешних подгружаемых функций. Преимуществом является также то, что в большинстве случаев процедуры и триггеры хранятся в компилированном виде. Это снижает влияние самих инструментальных средств измерения.

- ***Существует возможность только исполнения запросов к СУБД.*** При невозможности использования одного из уже рассмотренных вариантов, значения параметров могут быть определены только по времени исполнения тех или иных запросов к СУБД. В этом случае также следует обеспечить минимальное влияние посторонних факторов. Для исключения сетевого взаимодействия запросы предпочтительно выполнять непосредственно на сервере. Кроме того, желательно исполнять запросы с помощью предоставляемых СУБД программных вызовов для исключения влияния промежуточного ПО (драйверов баз данных). При этом точность определения может оказаться недостаточно высокой в связи с необходимостью учета большого числа факторов, поскольку полученные временные значения исполнения запросов будут включать в себя работу МРД на всех этапах его обработки. В этом случае значения отдельных параметров могут быть получены лишь косвенно.

Рассмотрим подробнее способ получения значений параметров временных потерь, вызванных встраиванием и функционированием МРД, основанный на использовании языковых средств СУБД.

При определении значений с помощью исполнения хранимых процедур на сервере необходимо учитывать следующие обстоятельства.

- Функция получения текущего значения временной метки должна обеспечивать необходимую точность (порядка микросекунд). Полученная разница замеров должна составлять несколько порядков единиц измерения, в противном случае необходимо обеспечить выполнение этого условия множественным повторением интересующей операции.

- Снятие временных меток должно осуществляться непосредственно до и после выполнения операции для исключения влияния остального кода.

- В зависимости от исследуемой операции, модель полученного значения может содержать несколько параметров. При этом значения самих параметров могут быть получены косвенным путем. Например, при замере операции обновления данных будет получено время полного исполнения запроса, включая его

разбор, оптимизацию, работы механизмов МРД, поиск и обновление данных.

- Точность определения потерь на уровне данных может быть повышена в случае поддержки языковыми средствами СУБД курсоров при доступе к данным, так как при этом полученное значение не включает влияние МРД на этапе разбора запроса.
- Вызов отдельных измерительных функций должен выполняться от имени конкретного пользователя для корректной работы МРД. Имперсонация в ходе исполнения запроса или после установки сессии может быть невозможной или влиять на точность измерений.

Предлагаемый подход к определению значений параметров представляет собой серию последовательных экспериментов, проводимых на эталонной БД. Полученные результаты измерений составляют систему уравнений, содержащих наборы параметров для каждой из проводимых операций, что позволяет выявить степень влияния того или иного параметра. Например, может быть использована следующая последовательность экспериментов.

1. Создание объекта измерения — таблицы. Для определения значений базовых параметров таблица должна быть минимальной (содержать один столбец) и по возможности должны быть отключены все МРД (пользователь является владельцем таблицы, списки доступа пустые, метки не назначены, использование меток столбцов отключено, строки таблицы метками не защищены, отключены все механизмы регистрации событий).

2. Измерение времени исполнения каждого из запросов select, insert, update, delete к созданной пустой таблице. Полученное время будет содержать потери на разбор запроса, минимальное определение прав доступа (владение), и потери на встраивание МРД, поскольку реальной обработки данных проведено не будет (за исключением операции insert).

3. Измерение времени исполнения запроса для одной записи. В дополнении к предыдущему пункту, потери будут включать обработку одной строки данных.

4. Для операции select возможно определение времени получения одной записи при проходе по результирующей выборке. Указанное время может характеризовать потери за счет встраивания МРД на уровне данных.

5. Измерение времени исполнения всех операций на различных объемах данных. Это позволяет определить зависимость временных потерь от числа записей. Зависимости определяются используемыми в данной конкретной СУБД способами доступа к данным.

Таким образом, могут быть получены значения временных потерь при доступе к одной таблице. Последовательно усложняя запрос добавлением других объектов или операций с ними, можно получать значения временных потерь для остальных интересующих параметров. Следует помнить, что при выполнении сложных запросов полученное значение временных потерь отражает совокупное влияние нескольких из рассмотренных видов потерь.

Описанный подход к оцениванию производительности реляционных систем управления базами данных в условиях реализации в них механизмов разграничения доступа позволяет получить представление о степени замедления выполнения запросов к БД. Это может помочь разработчикам минимизировать потери производительности при построении защищенных систем.

В случае достаточно точного определения параметров влияния МРД, возможно построение системы автоматизированной оценки такого влияния при выполнении заданного SQL-запроса по полученной из реальной БД или по заданной пользователем информации о метаданных и числе записей в объектах.

Подход может быть применен для оценки влияния не только рассмотренных МРД, а любых механизмов, расширяющих функциональность СУБД. При проведении исследований необходимо уделять внимание минимизации постороннего влияния путем соответствующей настройки исследуемых механизмов и инструментальных средств.

**Список литературы**


1. **Смирнов С. Н.** Безопасность систем баз данных. М.: Гелиос АРВ, 2007.
2. **Гарсия-Молина Г., Ульман Д., Уидом Д.** Системы баз данных. Полный курс: Пер. с англ. М.: Вильямс, 2003. 1088 с.
3. **Sumathi S., Esakkirajan S.** Fundamentals of Relational Database Management Systems. Springer, 2007. 776 p.
4. **Lane T.** A Tour of PostgreSQL Internals — Great Bridge. LLC, 2000. 25 p. URL: http://www.postgresql.org/files/developer/tour.pdf
5. **Шумилин А. В.** Перспективный подход к обеспечению защиты информации от несанкционированного доступа в СУБД // Программная инженерия. 2012. № 1. С. 35—40.
6. **Astra** Linux Special Edition. НПО РусБиТех. URL: http://astra-linux.com/
7. **PostgreSQL** 9.1.3 Documentation. The PostgreSQL Global Development Group, 2012. 2475 p.
8. **Date C. J.** An Introduction to Database Systems 8e. INSTRUCTOR, 2003. 401 p.
9. **Гостехкомиссия России.** Руководящий документ. Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации. М.: Военное издательство, 1992.

УДК 004.023

**П. В. Матренин**, магистрант, e-mail: pavel.matrenin@gmail.com,
**В. Г. Секаев**, канд. техн. наук, доц., Новосибирский государственный технический университет

# Оптимизация адаптивного алгоритма муравьиной колонии на примере задачи календарного планирования

*Рассматривается решение задачи календарного планирования методом колонии муравьев. Описываются приемы, позволившие значительно повысить качество получаемых решений и скорость работы программной реализации алгоритма планирования. Основное отличие предложенного в настоящей статье подхода от рассматриваемого ранее — в динамической эволюционной адаптации алгоритма под условия задачи. Представлено также описание программной реализации и рекомендации по оптимизации подобных программ.*

***Ключевые слова:*** *задача календарного планирования, метод колонии муравьев, многостадийные системы, генетический алгоритм, параллельные вычисления*

## Введение

Во всех сферах человеческой деятельности для достижения желаемого результата, как правило, составляются расписания и планы. Сложность задач планирования вместе с постоянным совершенствованием средств автоматизации подобной деятельности привели к развитию интереса к теории синтеза расписаний и календарному планированию. Задачи календарного планирования отражают процесс распределения во времени ограниченного числа ресурсов, выделяемых для выполнения проекта, который включает заданный перечень взаимосвязанных работ.

## Формализованная постановка задачи календарного планирования

Большинство задач по составлению календарных планов (КП) связаны с понятием многостадийных обслуживающих систем [1]. К их числу относят системы, в которых процесс обслуживания требований состоит их нескольких стадий. Несмотря на разнообразие производственных систем, формализованное описание задачи календарного планирования (*job-shop problem*), представленное в работе [1], может считаться базовым для большого класса многостадийных систем. Рассмотрим его основные положения.

1. Существуют конечное множество $N = \{1, 2, ..., n\}$ требований (работ, заказов) и конечное множество $M = \{1, 2, ..., m\}$ приборов (станков, исполнителей, рабочих станций и т. д.).

2. Процесс обслуживания требования $i$ включает $r_i$ стадий. При этом каждому требованию $i$ и каждой стадии $q$ ($1 \leqslant q \leqslant r_i$) его обслуживания ставится в соответствие некоторое подмножество приборов $M_q^i$ из множества $M$. Предполагается, что каждый прибор одновременно может обслуживать не более одного требования. В таких системах с последовательными приборами для каждого требования $i$ задается своя,

характерная для этого требования последовательность $\mathbf{L}^i$ его обслуживания приборами:

$$\mathbf{L}^i = \left\{ l_1^i,\ l_2^i,\ ...,\ l_{ri}^i \right\}.$$

3. Требование $i$ сначала обслуживается прибором $l_1^i$, затем прибором $l_2^i$ и т. д. Последовательности обслуживания могут быть различными для разных требований и могут содержать повторения приборов. Если требование $i$ на стадии $q$ должно быть обслужено прибором $l$, то предполагается заданной длительность $t_{liq}$ его обслуживания этим прибором. Процесс функционирования системы может быть описан путем задания расписания (календарного плана, плана-графика), т. е. некоторой совокупности указаний относительно того, какие именно требования какими именно приборами обслуживаются в каждый момент времени.

При принятых выше предположениях расписание можно рассматривать как вектор $\{s_1(t), s_2(t), ..., s_m(t)\}$, компоненты которого являются кусочно-постоянными непрерывными слева функциями, каждая из них задана на интервале $0 \leqslant t < \infty$ и принимает значения $0,1, ..., n$:

$$\mathbf{s} = \{s_1(t), s_2(t), ..., s_m(t)\}.$$

Если $s_l(t') = i$, $l \in M$, $i \in N$, то в момент времени $t'$ прибор $l$ обслуживает требование $i$. При задании расписания должны соблюдаться все условия и ограничения, вытекающие из постановки рассматриваемой задачи, т. е. расписание должно быть допустимым.

Если существует несколько допустимых расписаний, необходимо выбрать лучше из них, т. е. задать некоторый критерий выбора (критерий качества). В классической теории расписаний за такой критерий берется время завершения выполнения всех требований, т. е. время завершения последнего требования. Каждое допустимое расписание **s** однозначно определяет вектор $\mathbf{T}(\mathbf{s}) = \{T_1(s),\ T_2(s),\ ...,\ T_n(s)\}$ моментов времени завершения обслуживания требований.

Задается некоторая действительная неубывающая по каждой из переменных функция $F(x) = F(x_1, x_2, ..., x_n)$. Тогда качество расписания **s** оценивается значением этой функции при $x = \mathbf{T}(\mathbf{s})$: $F(x) = \max\{x_i\},\ i = 1, n$. В этом случае:

$$F(\mathbf{T}(\mathbf{s})) = T_{\max}(\mathbf{s}), \text{ где } T_{\max}(\mathbf{s}) = \max\{T_i(\mathbf{s})\},\ i = 1, n. \quad (1)$$

Из приведенной выше постановки задачи сразу заметны основные сложности ее решения: дискретность; многовариантность; многофакторность; отсутствие возможности построения целевой функции в виде алгебраического выражения, так как целевая функция рассчитывается только алгоритмически. Для реальных производственных задач точное оптимальное решение может быть найдено только теоретически (без учета фактора времени выполнения расчетов). Получение такого строгого решения на практике невозможно.

## Метод колонии муравьев при решении задач КП

Рассмотрим описание алгоритма муравьиной колонии для решения поставленной задачи и механизма адаптивного подбора значений его параметров.

**Описание метода.** Муравьи решают задачи поиска путей с помощью химической регуляции [2, 3]. Каждый муравей оставляет за собой на земле дорожку особых веществ — феромонов. Другой муравей, почуяв след на земле, устремляется по нему. Чем больше по одному пути прошло муравьев — тем заметнее для них след, а чем более заметен след — тем большее желание пойти в ту же сторону возникает у муравьев. Поскольку муравьи, нашедшие самый короткий путь к "кормушке", тратят меньше времени на путь туда и обратно, их след быстро становится самым заметным. Он привлекает большее число муравьев, таким образом, процесс поиска более короткого пути быстро завершается. Остальные пути — менее используемые — постепенно пропадают. Можно сформулировать основные принципы взаимодействия муравьев: случайность; многократность; положительная обратная связь.

Так как каждый муравей выполняет примитивные действия, то и алгоритм получается очень простым и сводится к многократному обходу некоторого графа, дуги которого имеют не только вес, но и дополнительную динамически меняющуюся количественную характеристику, называемую количеством феромона или просто феромоном.

Для решения задачи КП методом муравьиной колонии (МК) необходимо:

— представить задачу в виде ориентированного графа;

— определить эвристику поведения муравьев при построении решения;

— настроить параметры алгоритма.

Итерационный алгоритм МК включает построение решения всеми муравьями, улучшение решения методом локального поиска, обновление феромона. Построение решения начинается с пустого частичного решения, которое расширяется путем добавления к нему новой допустимой компоненты решения.

На основании алгоритма и формул, предложенных в работе [2], авторами были записаны представленные далее расчетные соотношения, которые используются при адаптации метода к задачам КП. Выбор компо-

ненты решения осуществляется по правилам вероятностного выбора на каждом шаге построения решения в соответствии с формулой

$$P_k = \frac{(f_k)^{\alpha}}{\sum_i (f_i)^{\alpha}}. \quad (2)$$

Коэффициент $\alpha$ определяет степень влияния количества феромона на $k$-й дуге $f_k$ на вероятность того, что муравей выберет эту дугу. В знаменателе — сумма по всем доступным из узла дугам.

Обновление феромона необходимо для его увеличения на лучшем (коротком) пути и для уменьшения его количества на путях, соответствующих плохим решениям. Используется также испарение феромона для того, чтобы избежать слишком быстрой сходимости алгоритма.

Если $F$ — значение целевой функции на маршруте, то количество феромона, нанесенного муравьем на все дуги маршрута $\Delta f$ можно определить как

$$\Delta f = \left(\frac{F}{\gamma}\right)^{\beta}. \quad (3)$$

Здесь $\beta$ и $\gamma$ — коэффициенты интенсивности выделения феромона. Коэффициент $\beta$ введен с тем, чтобы сделать зависимость нанесения феромона на граф более гибкой (не обязательно линейной).

После каждой итерации количество феромона на дугах графа уменьшается (испаряется) от значения $f$ до $f'$. Испаряемость феромона характеризуется коэффициентом $\rho$. При этом считается, что на дугах всегда должно оставаться некоторое минимальное не нулевое количество феромона ($f_{\min}$). В противном случае вероятность выбора дуги может оказаться нулевой и дуга будет "игнорироваться" муравьями. Максимальное значение ($f_{\max}$) также ограничено, что предотвращает возможную сходимость алгоритма к одному, далекому от оптимального, решению. Коэффициент принимает значения от 0 (нет испарения) до 1 (испаряется до минимального уровня). Таким образом, после испарения количество феромона на каждой дуге графа равно:

$$f' = \begin{cases} f(1-\rho), & f_{\min} < f(1-\rho) < f_{\max}, \\ f_{\min}, f(1-\rho) \leqslant f_{\min}, \\ f_{\max}, f(1-\rho) \geqslant f_{\max}. \end{cases} \quad (4)$$

В ходе экспериментальных исследований было выявлено улучшение результатов при усилении значимости текущего наилучшего решения. Для этого на все дуги пути, соответствующего лучшему результату, на каждой итерации добавляется некоторое количество феромона, которое определяется коэффициентом $\lambda$ и количеством феромона на предыдущей итерации ($f_{\text{best}}$). Итоговое количество феромона после усиления равно $f'_{\text{best}}$:

$$f'_{\text{best}} = \begin{cases} f_{\text{best}}\lambda, f_{\text{best}}\lambda < f_{\max}, \\ f_{\max}, f_{\text{best}}\lambda \geqslant f_{\max}. \end{cases} \quad (5)$$

Таким образом, ограничение на максимальное количество феромона учитывается и здесь.

Представим поиск решения задачи КП следующим образом. Пусть $W$ — суммарное число этапов всех работ из множества $N$.

Для того чтобы полностью задать расписание, достаточно указать, какую работу загружать на нужный прибор на каждом $i$-м шаге, $i = 1, 2, ..., W$. Тогда у графа будет $W + 1$ вершин, причем первая вершина соединена только со второй, вторая — с первой и третьей, третья — со второй и четвертой и т.д. (граф является ориентированным). Вершина с номером $W + 1$ соединена только с вершиной $W$. Дуги, соединяющие вершины, соответствуют работам.

Проходя по графу, муравей запоминает свой путь [2], в данном случае — последовательность работ. Как только работа $i$ попала в эту последовательность столько раз, сколько у нее этапов ($r_i$), муравей начинает игнорировать соответствующие ей дуги до конца пути.

**Адаптивный подбор параметров алгоритма.** Как отмечается в работах [2, 3], качество получаемых при применении муравьиного алгоритма решений сильно зависит от используемых в нем коэффициентов (параметров). В приведенном выше алгоритме такими коэффициентами являются $\alpha$, $\beta$, $\gamma$, $\rho$, $\lambda$ (формулы 2—5). Поскольку каждый из коэффициентов может принимать бесконечное число значений, встает вопрос о выборе коэффициентов, позволяющих получить решение, наиболее близкое к оптимальному. Подбор коэффициентов вручную неэффективен в силу большого диапазона их значений и отсутствия методик по их выбору. В данной работе предлагается проводить выбор коэффициентов при помощи их эволюционного отбора. Наиболее распространенным методом реализации такого отбора является генетический алгоритм (ГА). Подбор настроечных параметров осуществляется по изложенной далее схеме [4, 5].

1. **Генерация случайного начального состояния.** Первое поколение создается из произвольно выбранных решений (хромосом), где в качестве генов используются параметры $\alpha$, $\beta$, $\gamma$, $\rho$, $\lambda$.

2. **Вычисление коэффициента выживаемости (*fitness*).** Каждому решению (хромосоме) ставится в соответствие некое численное значение, зависящее от его близости к значению наилучшей функции.

3. **Воспроизводство.** Хромосомы, имеющие большую выживаемость (*fitness*), попадают к потомкам с большей вероятностью, осуществляется одноточечный (многоточечный) оператор кроссовера и оператор мутации.

4. **Следующее поколение.** Если новое поколение содержит решение достаточно близкое к значению наилучшей $F(x)$, то задача решена. В противоположном случае оно проходит через тот же процесс.

Работа генетического алгоритма представляет итерационный процесс до выполнения критерия останова, например, заданного числа поколений. В данном случае рассматривается задача непрерывной оптимизации [6]:

$$\min F(x),\ D = \{x_1, x_2, x_3, x_4, x_5 | x_i \text{ на } [a_i, b_i]\},$$

где $F(x)$ — минимизируемая целевая функция (в нашей работе это функция, рассчитываемая по формуле (1)), $D$ — область поиска, $a_i$ и $b_i$ — минимально и максимально возможные значения $x_i$ соответственно. Под решением указанной задачи будем понимать вектор $\mathbf{x} = \{x_1, x_2, x_3, x_4, x_5\}$. Оптимальным решением задачи будем считать вектор $\mathbf{x}^*$ с конкретными значениями $\alpha, \beta, \gamma, \rho, \lambda$, при котором целевая функция принимает наименьшее значение [6]. Результаты реализации адаптивных свойств алгоритма МК приведены далее.

## Программная реализация

Для исследования метода МК и вносимых модификаций было разработано и реализовано программное приложение с учетом сформулированных авторами особенностей данного типа задач [7]. Основные функции приложения: редактирование файлов с исходными данными (технологическими маршрутами и длительностями операций); поиск наилучшего решения задачи планирования методами МК и адаптивным методом МК; вывод результатов в различных формах; автоматизированное формирование отчета в MS Word.

Все представленные в настоящей статье данные о результатах, временных затратах на вычисления, выводы об эффективности применения тех или иных приемов получены в упомянутом приложении, которое прошло регистрацию и сертификацию в Федеральной службе по интеллектуальной собственности РФ [8].

## Повышение быстродействия программной реализации алгоритма

При описанном выше подходе резко возрастает время поиска решения, так как много раз запускается решение задачи методом МК. Существенно повысить скорость поиска можно с помощью распараллеливания расчетов, разделив популяцию на части и распределив вычислительную нагрузку по работе с этими частями между процессорами.

Мутация и вычисление функции приспособленности особей легко поддаются распараллеливанию, так как происходят для каждой особи независимо. При этом общие для всех особей данные используются только для чтения, поэтому не возникают сложности, связанные с необходимостью синхронизации этих этапов и потери времени на блокировки процессов при ожидании освобождения ресурсов.

Этап скрещивания сложнее распараллелить, так как в ходе этого этапа происходит взаимодействие между особями из разных частей популяции. Однако в распараллеливании нет необходимости, так как эта стадия занимает ничтожно малое время по сравнению с остальными расчетами. В данной работе для реализации параллельных вычислений использовалась API поточной обработки от Microsoft.

Можно выделить и другие приемы, более подробно описанные авторами в работах [6, 7, 9]. Среди них — выбор структуры графа и механизма нанесения на него феромона, оптимизация алгоритма расчета значений целевой функции, выбор средств разработки.

Особенно эффективным оказался следующий прием. Наносить феромон на матрицу, представляющую граф, не в конце каждой итерации (после обхода графа всеми муравьями), а создать копию матрицы и после обхода каждым муравьем увеличивать значение феромона в этой копии, а после окончания этапа обхода выполнять обратную замену.

Использование этого способа (обозначим два графа феромона как *fGraph* и *fTmpGraph*) позволяет намного повысить скорость расчетов, применив следующий прием. В пределах одной итерации число феромона на каждой дуге остается неизменным, поэтому значение параметра $(f_k)^\alpha$ из формулы (2) тоже не меняются в пределах одной итерации. В этом случае не возникает необходимость рассчитывать их для каждого муравья заново. Тогда на этапе инициализации дуги *fTmpGraph* получают начальное значение феромона, а дуги *fGraph* начальное значение в степени $\alpha$. На каждой итерации муравьи, обходя граф, ориентируются на *fGraph* (вычислять $(f_k)^\alpha$ уже не требуется — в этом и заключается суть повышения скорости работы), а феромон откладывают на *fTmpGraph*. После окончания обхода графа всеми муравьями и испарения феромона с графа *fTmpGraph*, каждая *k*-я дуга *fGraph* получает значение феромона, равное $fGraph_k = (fTmpGraph_k)^\alpha$.

## Экспериментальные результаты

Программная реализация метода проверялась на всемирно известных модельных задачах [10—12] и производственной задаче, представленной в работе [13]. Ниже

*Таблица 1*

**Примеры лучших наборов коэффициентов, полученных при использовании ГА**

| Задача | $I_z$ | $L_{\min}$ | $L_{avg}$ | $I_{mk}$ | $C_{ant}$ | α | β | ρ | γ | λ |
|---|---|---|---|---|---|---|---|---|---|---|
| abz6 | 40 | 948 | 977,333 | 1000 | 100 | 0,63 | 2,0 | 0,696 | 28 | 1,3 |
| abz6 | 40 | 945 | 982,524 | 1000 | 100 | 1,1 | 1,0 | 0,499 | 900 | 1,3 |
| ft10 | 40 | 950 | 995,866 | 1000 | 100 | 0,63 | 1,2 | 0,7 | 1000 | 1,1 |
| ft10 | 20 | 951 | 1006,1 | 1000 | 50 | 0,3781 | 1,711 | 0,97 | 1108,7 | 2,277 |
| la17 | 20 | 784 | 805,4 | 1000 | 100 | 0,392 | 2,7701 | 0,2998 | 425,5387 | 1,9615 |
| la17 | 20 | 785 | 798,25 | 1000 | 100 | 0,0341 | 1,7968 | 0,5461 | 949,1535 | 4,5589 |
| la15 | 20 | 1207 | 1215,45 | 1000 | 100 | 0,531 | 1,72 | 0,663 | 1032 | 1,2 |
| 3plity | 10 | 657,55 | 662,075 | 30 | 10 | 0,2782 | 0,4251 | 0,4919 | 296,61 | 2,5373 |
| 3plity | 10 | 657,55 | 662,3275 | 30 | 10 | 0,1754 | 0,5705 | 0,3836 | 326,05 | 2,5083 |
| la01 | 10 | 666 | 670,2 | 200 | 20 | 0,2262 | 0,8665 | 0,6883 | 903,3459 | 2,0363 |
| la01 | 10 | 666 | 669,4 | 200 | 20 | 0,3542 | 0,6527 | 0,8001 | 120,8012 | 2,1305 |
| la21 | 10 | 1107 | 1150,9 | 2000 | 100 | 0,7478 | 1,1134 | 0,3488 | 790,57 | 2,2236 |
| pr_6x10 | 20 | 107 | 111,45 | 30 | 10 | 0,6218 | 2,7953 | 0,6995 | 335,6241 | 1,2376 |
| pr_6x10 | 20 | 107 | 111,5 | 30 | 10 | 0,6204 | 0,2863 | 0,0775 | 137,6017 | 1,479 |

для примера приводятся данные по нескольким из тестовых задач. В табл. 1 приведены значения коэффициентов и параметров МК, которые были отобраны ГА как наилучшие для некоторых задач.

В табл. 1 и в следующих двух введены следующие обозначения:

$I_z$ — число запусков, по которым проводилось усреднение (с одинаковыми коэффициентами);

$L_{\min}$ — лучшее полученное решение;

$L_{avg}$ — усредненное по $I_z$ запускам решение;

$I_{mk}$ — число итераций МК;

$C_{ant}$ — число муравьев;

α — степень значимости феромона при выборе дуги графа;

β — нелинейный коэффициент нанесения феромона;

ρ — коэффициент испаряемости феромона;

γ — линейный коэффициент нанесения феромона;

λ — коэффициент учета лучшего текущего решения.

Из полученных данных следует, что лучшие найденные наборы коэффициентов различаются даже при решении одной и той же задачи, поиск взаимосвязи коэффициентов и их корреляции является одним из направлений дальнейшего развития работы.

Эксперименты показали существенное улучшение результатов по сравнению с полученными ранее (без эволюционного подбора коэффициентов), что отражено в табл. 2. Выделены столбцы с лучшими результатами до и после применения адаптации.

В табл. 2, кроме уже описанных, использованы следующие обозначения:

$L_{m1}$ — лучшее решение, которое было зафиксировано до использования ГА (коэффициенты подбирались оператором вручную, "на глаз");

$L_{a1}$ — среднее значение решений, которые были зафиксированы до использования ГА;

$L_{gm}$ — лучшее решение, которое было получено с использованием ГА;

$L_{ga}$ — среднее значение решений, которые были получены с использованием коэффициентов, найденных ГА;

$L_{m2}$ — лучшее решение, которое было получено с использованием коэффициентов, найденных усреднением по другим решенным задачам;

$L_{a2}$ — среднее значение решений, которые были получены с использованием коэффициентов, найденных усреднением по другим решенным задачам (кроме коэффициента γ);

Здесь под усредненными коэффициентами понимается набор коэффициентов, найденных как среднее арифметическое среди лучших найденных коэффициентов по задачам подобной размерности.

Для оценки повышения качества расписаний, составленных с адаптацией параметров метода, использовались квазиоптимальные решения тестовых задач [11]. Результаты приведены в табл. 3, выделены лучшие результаты решения, полученные авторами, и лучшие результаты, указанные в работе [14].

*Таблица 2*

**Сравнение результатов**

| Задача | $L_{m1}$ | $L_{a1}$ | $L_{gm}$ | $L_{ga}$ | $L_{m2}$ | $L_{a2}$ | $I_{mk}$ | $C_{ant}$ |
|---|---|---|---|---|---|---|---|---|
| abz6 | **980** | 1005,3 | **945** | 977,33 | 948 | 985,26 | 1000 | 100 |
| ft06 | **55** | 55,16 | **55** | 55 | 55 | 55,16 | 30 | 10 |
| ft10 | **1017** | 1038,8 | **950** | 995,87 | 975 | 1013,84 | 1000 | 100 |
| la01 | **666** | 673,08 | **666** | 669,4 | 666 | 673,08 | 30 | 10 |
| la10 | **958** | 958 | **958** | 958 | 958 | 958 | 1000 | 100 |
| la15 | **1211** | 1220,6 | **1207** | 1215,45 | 1207 | 1220,62 | 1000 | 100 |
| la17 | **796** | 809,32 | **784** | 798,25 | 787 | 809,32 | 1000 | 100 |
| la21 | **1121** | 1168,1 | **1107** | 1150,9 | 1118 | 1154,06 | 1000 | 100 |
| 3plity | **657,55** | 664,782 | **657,55** | 662,08 | 657,55 | 664,782 | 30 | 10 |
| pr_6x10 | **109** | 113,22 | **107** | 111,45 | 108 | 112,12 | 30 | 10 |

*Таблица 3*

**Сравнение результатов**

| Задача | $L_{m1}$ | $L_{a1}$ | $L_{gm}$ | $L_{ga}$ | $I_{mk}$ | $C_{ant}$ | ***Best*** |
|---|---|---|---|---|---|---|---|
| abz6 | 980 | 1005,3 | **945** | 977,33 | 1000 | 100 | **943** |
| ft06 | 55 | 55,16 | **55** | 55 | 30 | 10 | **55** |
| ft10 | 1017 | 1038,8 | **950** | 995,87 | 1000 | 100 | **930** |
| la01 | 666 | 673,08 | **666** | 669,4 | 30 | 10 | **666** |
| la10 | 958 | 958 | **958** | 958 | 1000 | 100 | **958** |
| la15 | 1211 | 1220,6 | **1207** | 1215,45 | 1000 | 100 | **1207** |
| la17 | 796 | 809,32 | **784** | 798,25 | 1000 | 100 | **784** |
| la21 | 1121 | 1168,1 | **1107** | 1150,9 | 1000 | 100 | **1048** |
| 3plity | 657,55 | 664,782 | **657,55** | 662,08 | 30 | 10 | — |
| pr_6x10 | 109 | 113,22 | **107** | 111,45 | 30 | 10 | — |

Примечание: "—" — по этой задаче нет данных.

В табл. 3, кроме ранее указанных, используется обозначение *Best* — условно оптимальные решения [14].

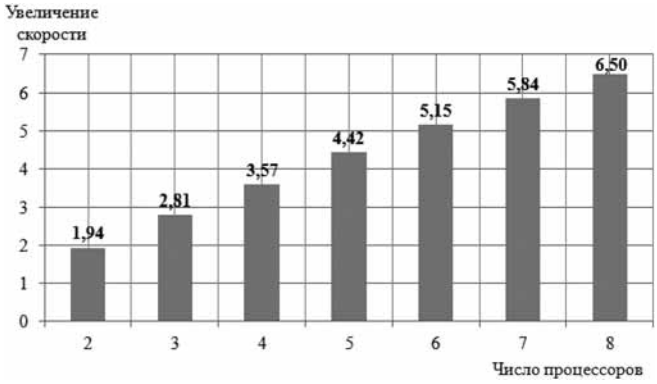
По задаче обработки станочных плит (3plity) приводится результат 657,55 [13], что также совпадает с решением, полученным в данной работе.

Эксперименты показали существенное улучшение результатов (до 12 %) по сравнению с найденными ранее (без адаптации коэффициентов).

Решения, полученные предложенным адаптивным алгоритмом по многим задачам из [10—12], оказались не хуже известных квазиоптимальных решений [14] по этим задачам. Отклонение от известных квазиоптимальных решений не превышает 6 % (в работе основателей муравьиного алгоритма [2] сказано об отклонении их результатов в задачах *job-shop* на 10 %). При этом адаптивный метод позволяет гарантированно получать решения указанного качества на каждом запуске при достаточном количестве итераций.

На рисунке приведены экспериментальные данные о росте скорости параллельных расчетов по сравнению с последовательной работой, в зависимости от числа используемых процессоров (ядер).

На рисунке видно, что предложенный подход позволяет существенно снизить время выполнения расчетов даже на двухъядерном процессоре.




**Эффект от распараллеливания расчетов**

Эффект от распараллеливания тем больше, чем больше объем расчетов в решении задачи методом МК, т. е. чем больше размерность задачи, число муравьев и число итераций. Действительно, согласно закону Амдала, параллельные вычисления тем эффективнее, чем больше доля расчетов, выполняемых параллельно, в рассматриваемых тестовых примерах доля таких расчетов составила 3...5 %.

Выигрыш от описанного выше приема использования двух графов тем больше, чем больше число муравьев и размер графа. В тестовых задачах календарного планирования с десятью требованиями по пять этапов время расчетов сократилось примерно в 4 раза, в задачах с десятью требованиями по десять этапов — в 5 раз, в задачах с 50 требованиями по десять этапов — в 7 раз.

## Заключение

Метод муравьиной колонии позволяет успешно решать задачи календарного планирования.

Использование ГА для поиска параметров метода упрощает исследование, позволяет сделать метод МК адаптивным к решаемой задаче и улучшить получаемые планы.

Наборы наилучших параметров алгоритма различаются от задачи к задаче, кроме того, зависят от модификаций алгоритма.

Адаптивный метод легко поддается распараллеливанию, что повышает производительность на многоядерных процессорах, существуют и другие эффективные способы повышения быстродействия.

**Список литературы**


1. **Танаев В. С., Сотсков Ю. И., Струсевич В. А.** Теория расписаний. Многостадийные системы. М.: Наука, 1989. 328 с.

2. **Dorigo M., Maniezzo V., Colorni A.** The Ant System: Optimization by a colony of cooperating agents // IEEE Transactions on Systems, Man, and Cybernetics — Part B. 1996. V. 26. № 1. P. 1—25.

3. **Штовба С. Д.** Муравьиные алгоритмы. // Exponenta Pro. Математика в приложениях. 2003. № 4. С. 70—75.

4. **Батищев Д. И., Власов С. Е., Булгаков И. В.** Решение задачи "слепого" упорядочивания при помощи генетических алгоритмов // Обозрение прикладной и промышленной математики. 1996. Т. 3. Вып. 5. С. 725—734.

5. **Норенков И. П.** Генетические методы структурного синтеза проектных решений // Информационные технологии. 1998. № 1. С. 9—13.

6. **Матренин П. В.** Адаптивный алгоритм муравьиной колонии при построении и оптимизации расписаний. // В мире научных открытий. 2012. № 8(32). С. 108—126.

7. **Матренин П. В., Секаев В. Г.** Программирование оптимизационных задач управления, взаимодействие с пользователем при длительных расчетах // Журнал научных публикаций аспирантов и докторантов. 2012. № 1. С. 67—69.

8. **Свидетельство № 2012614741.** Адаптивная система календарного планирования методом муравьиной колонии / П. В. Матренин, В. Г. Секаев. НГТУ — 2012612776; заяв. 11.04.12; опуб. 28.05.12.

9. **Матренин П. В., Секаев В. Г.** Повышение быстродействия приложений, использующих алгоритм муравьиной колонии, путем оптимизации их программного кода // Новый Университет. 2012. № 3(9). С. 74—80.

10. **Adams J., Balas E., Zawack D.** The shifting bottleneck procedure for job shop scheduling // Management Science. 1991. № 34. P. 391—401.

11. **Fisher H., Thompson G.** Probabilistic learning combination of local job-shop scheduling rules, in Industrial Scheduling. N. J.: Prentice-Hall, Englewood Cliffs, 1963.

12. **Lawrence S.** Supplement to "resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques". Tech. rep., GSIA, Carnegie Mellon University, October, 1984.

13. **Секаев В. Г.** Использование алгоритмов комбинирования эвристик при построении оптимальных расписаний // Информационные технологии. 2009. № 10. С. 61—64.

14. **Pezzella F., Merelli E.** A tabu search method guided by shifting bottleneck for the job shop scheduling problem // European Journal of Operational Research. 2000. № 120. P. 297—310.

УДК 004.02

**Д. В. Бутенко**, канд. техн. наук., **А. С. Ананьев**, аспирант, e-mail: ananievandr@yandex.ru,
**Л. Н. Бутенко**, д-р хим. наук, Волгоградский государственный технический университет

# Концептуальный подход к проведению предпроектных исследований информационных систем

*Описан алгоритм проведения предпроектных исследований и моделирования информационных систем. Реализация данного алгоритма требуется для решения актуальной научной задачи повышения результативности предпроектной работы по созданию информационных и технических систем. Определен перечень базовых процедур, необходимых для проведения предпроектной работы на стадии концептуального проектирования информационных систем. Предлагаемая методика позволит повысить результативность процесса моделирования и предпроектной работы при создании информационных систем за счет реализации системного подхода к их анализу и синтезу.*

***Ключевые слова:*** *концептуальное проектирование, системный анализ, инновационные технологии проектирования, информационные технологии*

Процессы генерации новых знаний становятся доминирующими в различных сферах деятельности человека, а умение конвертировать их в открытия, изобретения, новые технологии и реализующие их аппаратно-программные средства, информационные системы, в интеллектуальную собственность — базовым квалификационным требованием к выпускникам учебных заведений. Проблема новизны технических решений в сфере создания новых информационных продуктов и технологий приобретает особую остроту в условиях конкуренции, обеспечения интеллектуального превосходства в процессе перехода общества на постиндустриальную стадию развития, где доминируют информационные технологии и инновации [1]. Информационные и когнитивные технологии являются смысловой доминантой предстоящей NBIC-конвергенции [2]. Сейчас интеллектуальная составляющая в научной и инженерной подготовке требует большего внимания, а системная реализация интеллектуальных технологий должна опираться на новизну создаваемых решений. В настоящее время превосходство государственной политики не может быть обеспечено за счет импорта качественных технологий. Оно реализуется тогда, когда в основе новых технологий лежат собственные оригинальные идеи. Только в этом случае страна сможет стать лидером в экспорте технологий, чем обеспечит себе преимущество в экономике знаний.

Под интеллектуальной технологией будем понимать совокупность методов и средств генерации идей, построение на их основе инновационных, востребованных практикой решений, исследование и проектирование на основе таких решений инновационных продуктов, которые будут обладать высоким уровнем охраноспособности. Реализация таких продуктов должна обеспечить их конкурентоспособность при использовании на практике.

Начальные этапы любого проекта по созданию новых технологий объектов техники, такие как создание и анализ макетной версии, разработка основных положений проекта, относящиеся к поиску и обоснованию тех или иных вариантов решения поставленных задач, в настоящее время называются концептуальным проектированием систем [1]. Для стадий концептуального проектирования характерна низкая структурированность предметной области, многоаспектность характеризующих ее процессов, отсутствие достаточной количественной информации об их динамике, нечеткость, противоречивость, изменчивость процессов во времени, что приводит к большой неопределенности при разработке проектных решений.

Целью результатов исследований, представленных в данной работе, является определение перечня базо-

вых процедур (положений), которых рекомендуется придерживаться при проведении предпроектных исследований информационных систем, анализе их специфики и выборе инструментария для осуществления действий на стадии концептуального проектирования информационных систем.

Существующие в настоящее время методы предпроектных исследований достаточно активно обобщаются [3], однако большинство из них на практике не формализованы. Ярким примером, подтверждающим такие выводы, является ГОСТ 34.601—90. К недостаткам таких методов относится то обстоятельство, что они рассматривают только конкретные единичные макеты создаваемых систем и, как правило, направлены на поиск решения текущих задач и устранение единичных недостатков конкретного макета. Они опираются на субъективное определение целей и постановку задач. Как следствие, отсутствует полноценный анализ тенденций развития отдельных классов систем, которые с позиции современных требований науки и техники отражаются в целом ряде недостатков используемых для их создания макетов и прототипов. С учетом изложенного выше, авторами был сделан вывод о необходимости интеграции достоинств рассматриваемых ранее и найденных в открытых источниках подходов для повышения эффективности процедур предпроектных исследований применительно к предметной области проектирования информационных систем.

Технические и информационные системы относятся к классу искусственно созданных объектов, которым свойственно выполнение одной главной и большого числа дополнительных функций, а также многоэлементность и иерархичность. Для определения архитектуры и требований к условиям функционирования таких систем существует множество подходов, схем и методов, часть которых описана в книге А. И. Половинкина "Основы инженерного творчества" [3]. Применение системного анализа при создании сложных программных продуктов дает возможность выделить перечень и указать целесообразную последовательность выполнения взаимосвязанных задач. Такой подход позволяет не упустить из рассмотрения важные стороны и связи изучаемого объекта систематизации. За основу предлагаемой методики предпроектных исследований было принято концептуальное описание процесса проектирования технических систем, сделанное А. И. Половинкиным [3]. В качестве основных элементов в этой работе рассматриваются:

- главная полезная функция;
- функциональная структура;
- принцип действия;
- техническое решение;
- параметрическое решение.

Для концептуального описания процесса проектирования сложных информационных систем следует дать необходимые уточнения.

*Информационная система* — программно-аппаратный комплекс, предназначенный для автоматизации целенаправленной деятельности его конечных пользователей, обеспечивающий, в соответствии с заложенной в него логикой обработки, возможность получения и систематизации информации, ее хранения и выдачи по запросам пользователей.

*Главная полезная функция* информационной системы — это удовлетворение потребностей ее пользователей путем выдачи им релевантной информации по поступающим запросам.

*Функциональная структура* — совокупность описаний требований к функциональным возможностям системы и процедур выделения реализующих эти функции подсистем.

*Принцип действия* — это последовательность выполнения действий, базирующихся на преобразованиях потоков данных, которые обеспечивают выполнение требуемых функций информационной системы.

*Техническое решение* — это описание информационной системы на том или ином уровне его формализации (от вербального до строго математического) с представлением функциональной структуры и алгоритмов функционирования.

*Параметрическое решение* — это полное описание информационной системы с указанием технологических и конструктивных параметров.

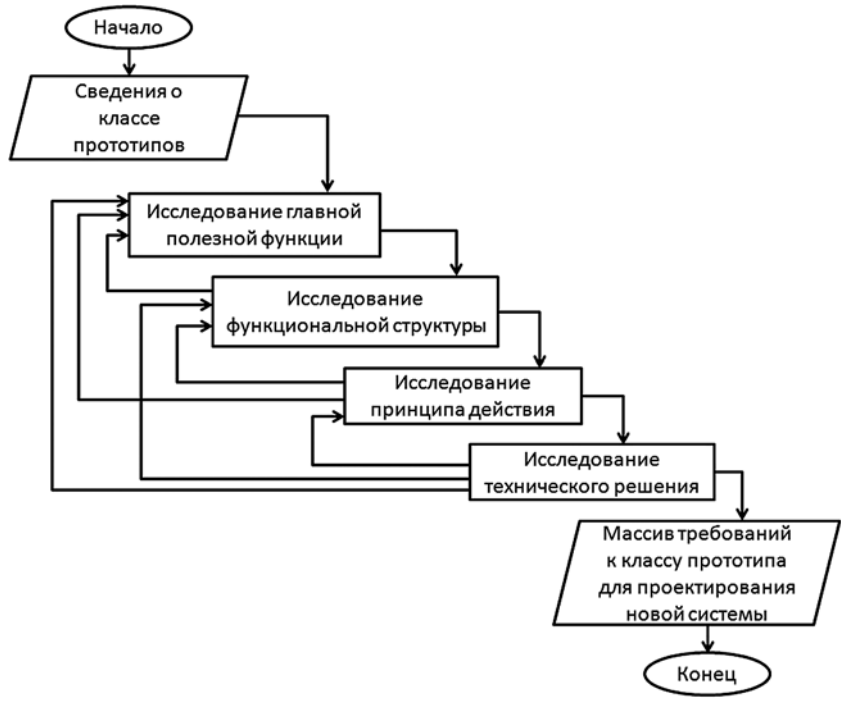



**Рис. 1. Обобщенный алгоритм проведения предпроектных исследований**

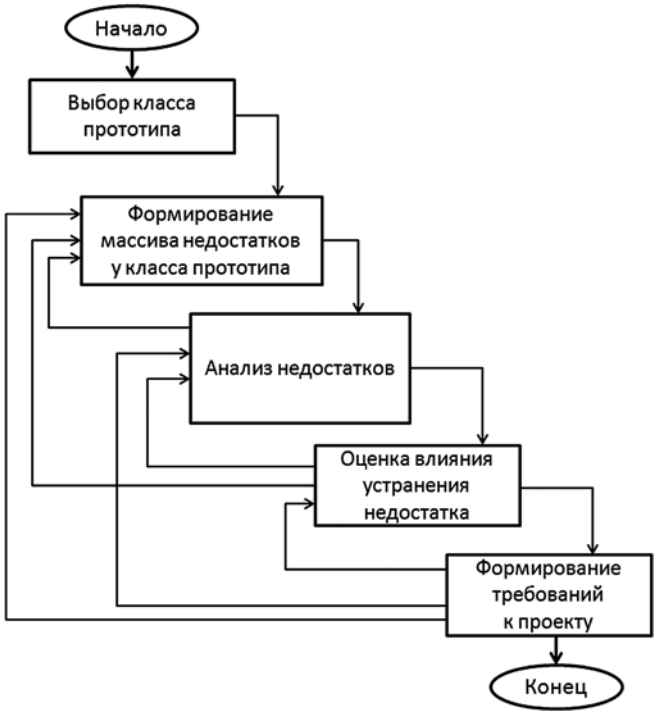



**Рис. 2. Описание стадии анализа и исследования прототипов**

На основе описанных выше определений и их уточнений разработано общее (концептуальное) описание процесса проведения предпроектных исследований, которое представляет собой следующую "рекуррентную" последовательность действий (рис. 1):

- сбор сведений о классе объекта исследования [4];
- исследование главной полезной функции системы;
- исследование функциональной структуры;
- исследование принципа действия;
- исследование технического решения;
- формирование перечня требований к классу объекта исследования для проектирования новой системы.

На основании представленного общего описания процесса проведения предпроектных исследований авторами разработано описание действий на стадии анализа и исследования прототипов, которое изображено на рис. 2.

На основе представленных решений можно составить кортеж уровней проектирования: $K = \langle T, E, R, S \rangle$, к которым они применимы. Здесь $T$ — концепт для главной функции; $E$ — концепт для функциональной структуры; $R$ — концепт для принципа действия; $S$ — концепт для технического решения. Таким образом, для определения тенденции развития стадий предпроектного исследования предложенные решения можно использовать для каждого изучаемого объекта.

Общая последовательность действий при проектировании информационной системы в соответствии с выделенными и представленными выше стадиями была апробирована на примере проектирования и разработки ряда компьютерных комплексов и систем, по которым получен ряд свидетельств о регистрации программ в Роспатенте (№ 2010617474, № 2010612774, № 2010612500, № 2011615174, № 2011619360, № 2011612133 и др.). К их числу относятся программа психосемантического анализа и визуализации звука, которая соответствует ФЗ № 436 "О защите детей от информации, причиняющей вред их здоровью и развитию" № 2012616458 [5].

Представленные концептуальные положения методики предпроектных исследований, по мнению авторского коллектива, отвечают современным тенденциям в образовании, обеспечивая современный уровень подготовки специалистов. Предлагаемая методика позволяет повысить результативность предпроектных исследований при создании информационных систем за счет реализации системного подхода к их анализу и синтезу.

**Список литературы**

1. **Бутенко Д. В., Попов К. В., Ананьев А. С.** Методика концептуального проектирования программных информационных систем // Программные продукты и системы. 2012. № 2. С. 101—104.

2. **Бутенко Д. В., Бутенко Л. Н.** Теория развития систем. Задачи концептуального проектирования и их взаимосвязь с закономерностями развития систем // Качество. Инновации. Образование. 2004. № 1. С. 38—41.

3. **Половинкин А. И.** Основы инженерного творчества: Учеб. пособие для студентов втузов. М.: Машиностроение, 1988. 368 с.

4. **Ананьев А. С., Бутенко Д. В., Попов К. В.** Интеллектуальные технологии проектирования информационных систем. Методика проектирования программных продуктов в условиях наличия прототипа // Инженерный вестник Дона: электронный журнал. 2012. № 2. URL: http://ivdon.ru/magazine/archive/n2y2012/815/

5. **Бутенко Д. В., Ананьев А. С., Попов К. В.** Концептуальное проектирование информационных систем. Программная среда психосемантической идентификации звука // Известия ВолгГТУ. Серия "Актуальные проблемы управления, вычислительной техники и информатики в технических системах". Вып. 14: межвуз. сб. науч. ст. 2012. № 10(97). С. 151—155.

УДК 004.41

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

# Программная инженерия: требования к профессиональной компетенции кадров

*В настоящей публикации в кратком изложении со ссылкой на положения изданных автором учебников и монографий по Программной инженерии рассматриваются общие вопросы, возникающие в процессе сопровождения сложных программных продуктов на основных этапах их жизненного цикла. Анализируются требования к компетенции участвующих в этой деятельности кадров специалистов. Для каждого требования представлены учебники или монографии, опубликованные автором, которые описывают Программную инженерию.*

***Ключевые слова:*** *магистры, программные продукты, профессиональная компетенция, технология, проектирование, производство, испытания, эксплуатация*

***Программная инженерия*** — это область компьютерной науки и технологии, одной из главных задач которой является исследование методов и средств построения программных систем, настолько больших и сложных, что для этого требуется участие слаженных ***команд разработчиков различных специальностей и квалификаций***. Обычно такие системы существуют и применяются долгие годы, развиваясь от версии к версии, претерпевая на своем жизненном пути большое число изменений. Улучшаются существующие функции, добавляются новые или удаляются устаревшие возможности. Программные системы адаптируются для работы в новой среде, устраняются дефекты и ошибки. ***Суть методологии программной инженерии*** состоит в применении систематизированного, научного и предсказуемого процесса проектирования, производства и сопровождения программных комплексов реального времени. ***Управление промышленным проектированием и производством*** сложных программных продуктов — это особый вид рациональной, коллективной деятельности, включающий постановку задач, исследования и подготовку проектных решений, планирование процесса производства, организацию работ и контроль за ходом выполнения работ и использованием ресурсов. Задачи целевого управления такими работами — сводить воедино усилия руководителей, исполнителей — специалистов разной квалификации, подрядчиков и субподрядчиков, добиваясь, чтобы они ***выступали как целевая команда***, а не как разрозненная группа, реализующая свои цели. Для используемых средств производства характерна взаимосвязь реализуемых с их помощью технологий на отдельных этапах производственного процесса, в результате выполнения которых используемые на предприятии объекты и процессы превращаются в готовую продукцию. В итоге должна обеспечиваться концептуальная целостность создания технических систем и высокое качество решения главных задач заказчиков при сбалансированном использовании ресурсов.

Массовое создание сложных программных средств промышленными методами и большими коллективами специалистов вызвало необходимость их четкой организации, планирования работ с учетом необходи-

мых для этого ресурсов, с привязкой к этапам и срокам реализации. Для решения этих задач в программной инженерии формируется новое направление исследований — ***экономика жизненного цикла программных комплексов***. Это направление включается в исследования экономики промышленного производства средств и систем вычислительной техники. Объективно исследования на этом направлении осложняются трудностью измерения экономических характеристик производства программного продукта. Широкий спектр количественных и качественных показателей, которые с различных сторон характеризуют содержание подобных продуктов, и невысокая достоверность оценки их значений определяют значительную дисперсию при попытках описать и измерить экономические свойства создаваемых или используемых крупных комплексов программ. Вследствие роста сфер применения и повышения ответственности выполняемых ими функций, резко возросла необходимость ***гарантировать высокое качество программных продуктов***, а также регламентирования и корректного формирования требований к характеристикам реальных комплексов программ и их достоверного определения. В результате специалисты в области теории и методов, определяющих качество продукции, вынуждены осваивать сферу развития и применения нового, несвойственного для их основной деятельности продукта — отдельных программных средств и систем в целом, а также подходов к оценке их качества при использовании. Сложность анализируемых объектов и уверенность ряда программистов в собственной "непогрешимости" зачастую приводят к тому, что реальные характеристики качества функционирования программных продуктов остаются неизвестными не только для заказчиков и пользователей, но также для самих разработчиков. Отсутствие четкого декларирования в простых документах атрибутов и значений, характеризующих характеристики качества программных комплексов, вызывает конфликты между заказчиками-пользователями и разработчиками-поставщиками вследствие разной трактовки одних и тех же характеристик.

***Руководство организации-заказчика*** должно быть уверено, что их требования полностью понятны исполнителю-поставщику и могут быть выполнены. Для этого заказчик во взаимодействии с исполнителем должен установить общесистемные процедуры для ***управления требованиями и документами***, которые необходимы для эффективного функционирования системы проектирования и производства. Такая система должна обеспечивать уверенность заказчика в том, что документы анализируются, при необходимости уточняются и снова утверждаются. ***Управление специалистами*** необходимо для того, чтобы подтверждать заказчику должный уровень их компетенции для осуществления своей деятельности, который гарантируют соответствующее образование, знания и практические навыки проектирования и производства продукции.

Методы программной инженерии поддерживают и конкретизируют ***технологический процесс***, а также обеспечивают перманентный контроль качества программного продукта на всех стадиях его производства. Для каждого проекта создания программного продукта, выполняющего ответственные функции, должны разрабатываться и применяться система качества, планы и программа испытаний, в рамках которых реализуется специальная методология и используются определенные инструментальные средства разработки и испытаний. Такая методология и инструментарий должны обеспечивать заданные на стадии проектирования ***качество, надежность и безопасность функционирования программных продуктов***. Эти методы и процессы позволяют разработчикам и заказчикам программных продуктов более корректно взаимодействовать при определении и реализации требований контрактов и технических заданий.

Для эффективного процесса изготовления определенного программного продукта и его использования в конкретных условиях эксплуатирующего предприятия, необходимо ***создание соответствующей технологии его проектирования и производства***. Технология производства включает методы, используемые для этого оборудования, инструментальные средства и систему взаимосвязанных способов изготовления продукции и выполнения установленного вида работ. Технология должна включать весь перечень последовательных операций по превращению исходного замысла проекта, требований заказчика и потребителей в готовый продукт, с указанием методов, типа и характеристик инструментальных средств, которыми специалисты пользуются на каждом этапе производства.

***Организация технологии проектирования и производства*** — система мер, направленных на рационализацию взаимодействия в пространстве и времени материальных компонентов и специалистов, занятых в процессе производства. Дифференциация коллективного труда предполагает разделение производственного процесса на отдельные части (процессы, операции) и их закрепление за соответствующими подразделениями и специалистами предприятия. В практической деятельности по организации производства приоритет должен отдаваться тому методу, который обеспечит ***наилучшие экономические и социальные характеристики производственного процесса***.

Основные концепции программной инженерии сконцентрированы в ***целостном комплексе система-***

***тизированных международных стандартов***, охватывающих и регламентирующих практически все процессы жизненного цикла сложных программных средств. Несколько десятков стандартов этого комплекса допускают целенаправленный отбор процессов. С использованием положений этих стандартов формируют проблемно-ориентированные ***профили стандартов***, предназначенные для определенных типов проектов и/или предприятий-заказчиков. Практическое применение профилей стандартов, сосредоточивших мировой опыт создания различных типов крупных комплексов программ, способствует значительному ***повышению производительности труда*** специалистов и ***качества*** создаваемых ими программных продуктов. Эти стандарты определяют эффективную модифицируемость, мобильность и возможность повторного применения программных компонентов и комплексов, их расширяемость и переносимость на различные аппаратные и операционные платформы. Отмеченные факторы отражаются на росте экономической эффективности технологий и процессов создания различных программных средств и систем. Для регламентирования процессов жизненного цикла такие профили стандартов должны ***адаптироваться*** и конкретизироваться применительно к определенным классам и функциям проектов, процессов и компонентов программных комплексов. При этом должна сохраняться концептуальная целостность применяемой совокупности стандартов. Должно обеспечиваться их положительное влияние на процессы и результаты, на качество, надежность и безопасность программных продуктов при ***реальных ограничениях*** на ***использование ресурсов***, доступных для их сопровождения в жизненном цикле.

***Методология программной инженерии*** и международные стандарты рекомендуют современные методы и средства проектирования и производства, внедрения и применения программных продуктов в составе сложных систем. Организованные коллективы специалистов, применяющих традиционные, регламентированные процессы программирования, верификации, тестирования и сопровождения программных комплексов и их компонентов, обеспечивают заказчикам ***стабильные, предсказуемые результаты***, а именно программные продукты с требуемым качеством обработки технической информации. Основной интеллектуальный труд специалистов вкладывается в разработку функциональных алгоритмов и текстов программ, а также в их интеграцию, испытания, документирование на всех этапах проектирования и производства сложных программных продуктов. Большая сложность информации в таких системах определяет высокие ***затраты квалифицированного интеллектуального труда*** специалистов на комплексы программ для ***материализации информации в программном продукте***.

Методология программной инженерии и стандарты ***регламентируют современные процессы управления проектами*** сложных систем и программных продуктов. Они обеспечивают организацию, освоение и применение апробированных годами, высококачественных процессов проектирования, программирования, верификации, тестирования и сопровождения программных комплексов и их компонентов. Тем самым, эти проекты и процессы позволяют получать стабильные, предсказуемые результаты и программные продукты требуемого качества. Многообразие классов и видов сложных комплексов программ, обусловленное различными функциями и сферами применения систем, определяет формальные трудности, связанные с методами и процедурами доказательства ***соответствия создаваемых и поставляемых программных продуктов*** условиям контрактов, требованиям заказчиков и потребителей. По мере расширения применения и увеличения сложности систем, выделились области, в которых дефекты, недостаточное качество комплексов программ или данных могут наносить значительный ущерб, намного превышающий положительный эффект от их использования. В таких ***критических системах*** (например, управления атомными электростанциями, крупными банками или системами вооружения) ***недопустимы проявления высоких рисков нарушений принятых режимов функционирования*** программных продуктов при любых искажениях исходных данных, сбоях, частичных отказах аппаратуры, ошибках пользователей и других нештатных ситуациях. Подобные риски комплексов программ могут определять безопасность функционирования объектов, предприятий и даже страны. Вследствие этого резко ***повысилась ответственность специалистов за качество*** результатов их труда и создаваемых программных продуктов. Это требует непрерывного совершенствования, обучения и повышения квалификации заказчиков, разработчиков и пользователей в области программной инженерии, освоения ими современных методов, процессов и международных стандартов, а также ***высокой корпоративной культуры коллективов специалистов***, обеспечивающих жизненный цикл критических программных продуктов.

В соответствии с требованиями федерального образовательного стандарта, областью профессиональной деятельности магистров по направлению подготовки ***231000 Программная инженерия*** является ***индустриальное производство программного обеспечения для информационно-вычислительных систем различного назначения***. Объектами профессиональной деятельности специалистов являются: программные проекты; программные продукты; методы и инструментальные

средства разработки программных продуктов; коллективы специалистов, участвующие в процессах жизненного цикла сложных комплексов программ. Они должны быть компетентными в основных видах профессиональной деятельности.

В образовательном стандарте каждому виду деятельности сопутствуют очень ***краткие описаниями требований к профессиональной компетенции*** магистров, которые недостаточны для создания на конкретных предприятиях и в вузах проблемно-ориентированных программ и планов обучения на степень магистра. Эти описания рекомендуются при обучении и аттестации на степень магистра программной инженерии. Однако состав компетенций содержит крупные недостатки и ***не отражает требования к современному индустриальному производству сложных программных продуктов***. Вызывает удивление структура и распределения ресурсов на профессиональные компетенции — непосредственному проектированию и производству программных продуктов посвящено только 5—7 профессиональных компетенций из 17, но они не конкретны и не отражают все разнообразие индустриальных процессов и компетенций, необходимых магистрам для создания современных программных продуктов высокого качества.

Для обучения и повышения квалификации и компетенции специалистов до уровня магистров автором подготовлен комплекс учебников и монографий, содержащих полные электронные копии (они в свободном доступе полных текстов представлены по адресу http://istina.imec/msu.ru.profile/V. V. Lipaev).

- **Липаев В. В.** Проектирование и производство сложных заказных программных продуктов. М.: СИНТЕГ, 2011. 408 с.
- **Липаев В. В.** Экономика производства программных продуктов. Издание второе. М.: СИНТЕГ, 2011. 358 с.
- **Липаев В. В.** Человеческие факторы в программной инженерии: рекомендации и требования к профессиональной квалификации специалистов. Учебник. М.: СИНТЕГ, 2009. 328 с.
- **Липаев В. В.** Тестирование компонентов и комплексов программ. Учебник. М.: СИНТЕГ, 2010. 400 с.
- **Липаев В. В.** Сопровождение и управление конфигурацией сложных программных средств. М.: СИНТЕГ, 2006. 372 с.
- **Липаев В. В.** Сертификация программных средств. Учебник. М.: СИНТЕГ, 2010. 348 с.

## *ИНФОРМАЦИЯ*

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

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

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

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

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

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

# CONTENTS



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

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

Сдано в набор 04.02.2013 г. Подписано в печать 20.03.2013 г. Формат 60×88 1/8. Заказ PI413
Цена свободная.

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