ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г.

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 004.4

В. А. Васенин, проф.,
д-р физ.-мат. наук, Институт проблем информационной безопасности,
МГУ имени М. В. Ломоносова,
e-mail: vasenin@msu.ru

Модернизация экономики и новые аспекты инженерии программ

Анализируется ключевая роль программной инженерии как области научно-технических знаний в модернизации экономики на современной, постиндустриальной стадии развития общества. Рассматриваются основные этапы становления этой области как результат эволюции вычислительной техники, методов и средств сопровождения ее программного обеспечения на всех этапах жизненного цикла. Представлены, в том числе с иллюстрацией на примерах, взгляды автора на программную инженерию с позиции ее состояния и как области знания, и как основного инструментария модернизации экономики путем автоматизации процессов во всех секторах национальной экономики.

Ключевые слова: *программная инженерия, автоматизация, программный продукт, жизненный цикл, модернизация экономики, вычислительные средства и системы*

От экономики индустриального к экономике информационного общества

Программная инженерия, как и любая другая область научно-технических знаний, представляет собой инструментарий, основной целью применения которого является производство продукта (услуги), обеспечивающего потребности мировой, или в масштабах отдельной страны, ее национальной экономики. Под национальной экономикой при этом понимается система, объединяющая существующие в стране сферы общественных отношений и реализующиеся в их рамках формы трудовой деятельности. Они включают материальное и нематериальное производство, а также сферу непроизводственной деятельности [1]. Кроме традиционных, сложившихся и преобладающих в индустриальном обществе отношений в сфере материального производства, в ходе которых создаются необходимые обществу средства производства и предметы потребления, экономика включает в себя производство продуктов (предметов) нематериального характера. К их числу относятся научные знания и произведения искусства, составляющие основу интеллектуальной сферы общественных отношений, а также услуги, оказываемые населению. Последние в большей степени характеризуют деятельность, направленную на обслуживание государственных институтов, включая армию, охрану общественного порядка и национальную безопасность, органы государственного управления, религиозные и общественные организации.

В широком понимании модернизация национальной экономики должна определять изменения во всех перечисленных выше сферах общественных отношений в стране. Модернизация в индустриальном обществе обеспечивала переход от преимущественно ручного, кустарного производства в доиндустриальном обществе[1] [2], в первую очередь средств производства и предметов потребления, а также наиболее востребованных им предметов (объектов) нематериального производства, к их массовому, индустриальному производству. Продукты такого способа производства и, соответственно, требования к ним в первоочередном порядке должны были предусматривать:

— их более высокие функциональные свойства (объем и точность выполняемых с их помощью функций);

[1] В нотации теории модернизации, в основе которой такие стадии развития общества как первобытная, доиндустриальная, индустриальная и постиндустриальная.

— требования к их эффективной отчуждаемости от создателя (производителя) и, как следствие, к возможности модифицировать их многочисленными пользователями (потребителями) самостоятельно;

— высокое качество выполняемых ими функций.

Следует заметить, что производства предметов (объектов) в сфере интеллектуальной деятельности такие изменения коснулись в значительно меньшей степени.

На рубеже XX и XXI веков в развитии мировой цивилизации наметилась новая тенденция, обусловленная процессом объективного развития общества. Отправным мотивом такого развития стало создание в 1970—1980-х гг. новых технологий, индустриально выпускаемых промышленностью на их базе средств вычислительной техники и телекоммуникаций, микро- и мини-ЭВМ. Следующим шагом (1980—2000 гг.) стало повышение производительности информационно-вычислительных систем, интеграция их ресурсов с помощью высокоскоростных коммуникаций для эффективного, надежного и защищенного разделения таких ресурсов между потенциальными пользователями. Эффективно используемые в массовом порядке вычислительные системы и сети передачи данных (сети связи) на основе пакетных коммуникаций кардинальным образом изменили формы (методы и средства) деятельности человека во всех перечисленных выше сферах отношений, характерных для индустриального общества. Автоматизация процессов, сопровождающих такую деятельность, с использованием вновь создаваемых информационно-вычислительных и телекоммуникационных средств и систем сформировала многоаспектный (многопрофильный) вектор, определяющий переход от индустриального общества к новому его состоянию, которое принято именовать **постиндустриальным** или **информационным** обществом [3]. Главными (определяющими) характеристиками этого вектора являются информационные технологии, ориентированные на создание, сопровождение и перманентную, отвечающую требованиям времени, модернизацию элементной базы средств вычислительной техники и автоматизированных систем на их основе, а также математического и программного обеспечения подобных средств и систем.

Целью экономической политики любого государства является создание эффективной и конкурентоспособной экономики. Методы и реализующие их механизмы достижения этой цели — набор инструментальных средств, позволяющих создавать благоприятную среду для хозяйственной деятельности всех субъектов экономики, независимо от форм собственности. В информационном обществе, где эффективность деятельности во всех сферах общественных отношений определяется уровнем ее информатизации и, соответственно, автоматизации сопровождающих такую деятельность процессов, основным ресурсом модернизации экономики являются перечисленные выше технологии. Отметим тот факт, что в постиндустриальном обществе это касается не только информационных технологий, включая математическое и программное обеспечение автоматизации процессов создания средств и систем производства продуктов и услуг, используемых в массовом порядке. В значительной степени это относится и к средствам, системам автоматизации сложно организованных, науко- и ресурсоемких процессов в интеллектуальной сфере общественных отношений (в поисковых фундаментальных исследованиях и наукоемких производствах), в непроизводственной сфере (критических для государства секторах экономики). Более того, США и экономически развитые страны Европы модернизацию национальной информационно-телекоммуникационной инфраструктуры, ее критически важных для национальной безопасности сегментов и технологий расценивают как главный резерв обеспечения своего мирового военно-технического превосходства.

Вместе с отвечающими требованиям массового пользователя архитектурные решения средств вычислительной техники и коммуникаций, систем автоматизации процессов в различных областях деятельности человека, информационные технологии должны быть:

— менее ресурсозатратными на этапах их реализации и эксплуатации в составе целевых систем;

— более эргономичными, комфортными для пользователя — главного субъекта и объекта отношений во всех сферах жизнедеятельности общества, как следствие — эффективно отчуждаемыми от их производителя и легко модифицируемыми потребителем;

— функциональными — в должной степени удовлетворяющими назначению, ради которого они создаются;

— лучшего качества при выполнении заявленных функций;

— защищенными от потенциально возможных деструктивных воздействий как злоумышленных, так и непреднамеренных, менее опасны для людей и окружающей среды, с которыми они взаимодействуют.

Перечисленные выше общие требования к информационным технологиям, к средствам и системам автоматизации, поддерживающим различные секторы экономики, привели к необходимости существенного пересмотра подходов к их разработке и внедрению в практику, к сопровождению и перманентной модернизации. Результаты такого осмысления, критического анализа сложившейся еще в индустриальном обществе практики, ее изучения, создали предпосылки для выработки новых моделей и механизмов, методов и средств управления процессами создания и сопровождения информационных технологий, в первую очередь, их программного обеспечения, на всех этапах их жизненного цикла — от разработки до вывода из обращения. Систематизированные в ходе поисковых и прикладных исследований, такие знания применительно к программному продукту составляют основу научно-технической области знания и учебной дисциплины "Программная инженерия" [4, 5], а также тесно примыкающих к ней "Информационной безопасности" [6—8] и "Функциональной безопасности" [9]. При этом необходимо отметить тот факт, что все эти области знаний еще относительно молоды, так как оказались востребованы переходом к технологиям

постиндустриального общества. История их становления насчитывает менее 40 лет, наиболее активного развития — менее 20 лет. Однако беспрецедентно быстрое развитие информационных технологий в постиндустриальном обществе, порождаемые им новые вызовы, заставляют с других позиций взглянуть на эти области [10]. Принимая во внимание последнее обстоятельство, рассмотрим далее роль этих наук в модернизации экономики, обращая особое внимание на процессы создания и сопровождения математического и программного обеспечения средств вычислительной техники и информационных систем на их основе. Для краткости изложения будем условно именовать отмеченные выше три взаимодополняющие друг друга области научно-технических знаний "Программной инженерией в широком ее понимании" или более коротко — "Программной инженерией".

Особенности экономики и информатизации в современной России

Особенности Российской Федерации накладывают определенные ограничения на процессы модернизации ее экономики и, как следствие, на темпы изменения национальной телекоммуникационной среды и информационно-вычислительной инфраструктуры, на базе которых происходит совершенствование средств и систем автоматизации в отдельных ее секторах. Не вдаваясь в детали, отметим, что специфику России на настоящем этапе развития страны определяют следующие, влияющие на темпы модернизации ее экономики факторы.

- Россия находится в стадии перехода от централизованного, директивно-планового способа хозяйствования к рыночным механизмам управления экономикой. В условиях многоукладности, территориальной удаленности ее системообразующих объектов, а также целого ряда других факторов этот процесс еще не завершен. Это обстоятельство объективно затрудняет подготовку должным образом обоснованных среднесрочных и тем более долгосрочных планов модернизации экономики и, в частности, национальной информационно-телекоммуникационной инфраструктуры.
- По причинам субъективного и объективного характера Россия отстает от ряда развитых в экономическом и технологическом отношении стран в области создания и массового производства собственной элементной базы для современных средств вычислительной техники, телекоммуникаций и автоматизированных систем, а также уступает этим странам на направлении разработки низкоуровнего системного и среднего слоя (*middleware*) программного обеспечения. Это обстоятельство и, как следствие, отсутствие соответствующих кадров специалистов ограничивают возможности проведения востребованных экономикой поисковых исследований и широкомасштабных прикладных работ с применением положений "Программной инженерии" на указанных направлениях.
- В силу отмеченных выше особенностей разработка программного обеспечения в России сложилась ранее и преобладает в настоящее время на основе подходов "программирование как искусство", которые характерны для индустриального общества. Эти подходы основаны на методах и средствах, которые являются результатом интеллектуальной деятельности программиста, его эвристических представлений об области назначения создаваемого им продукта, требованиях к нему, алгоритмическом обеспечении и средствах представления программы в понятных для ЭВМ кодах. Для программных средств, предназначенных для ограниченного круга пользователей, такой подход к их разработке оправдан. Однако, как уже отмечалось ранее, в условиях информационного общества, при разработке продуктов массового использования или сложно организованных в архитектурно-технологическом плане критически важных программных комплексов, такой подход становится неэффективным.
- В России на настоящее время нет (не осталось) необходимого числа специалистов, обладающих опытом проведения крупных проектов, включающих поисковые исследования и прикладные работы, направленных на создание и внедрение в практику программного обеспечения средств вычислительной техники и телекоммуникаций, автоматизированных систем, которые предназначены для массового использования. Нет подобного масштаба проектов, которые, наряду с государством, финансируются ведущими бизнес-структурами. Особенностью таких проектов является то, что они, как правило, не дают большой прибыли в относительно короткое время после их выполнения. В условиях отсутствия нормативно-правовых регуляторов, позволяющих стимулировать участие бизнес-структур в подобных проектах, они, как правило, уклоняются от участия в их финансировании. В результате многие национально значимые и даже критически важные для модернизации экономики страны проекты не могут быть выполнены в рамках объективно ограниченных возможностей их финансирования со стороны государства.
- В условиях ограниченного бюджетного финансирования, в целях уменьшения рисков и сокращения сроков получения конечных (практических) результатов при отборе проектов, предпочтение отдается тем, которые основаны на типовых, уже апробированных инженерных решениях, в ущерб другим, имеющим более хорошие инновационные перспективы, требующим значительной исследовательской составляющей. Как следствие, содержание требований к средствам и системам автоматизации деятельности в различных секторах экономики, которые излагаются в тендерной документации, ориентировано на опытно-конструкторские работы с использованием широко представленных на рынке информационных технологий аппаратных средств, как правило, зарубежного производства и поддерживающего их проприетарного (закрытого — без предоставления пользователю открытых исходных кодов) программного обеспечения.

Несмотря на перечисленные выше особенности и объективные причины, ограничивающие возможности модернизации экономики России за счет информатиза-

ции различных ее секторов и автоматизации поддерживающих их производственно-технологических процессов, важность движения в этом направлении осознается на разных уровнях государственного управления. Такое понимание выражается в виде финансирования на тендерной основе проектов в рамках крупномасштабных национальных программ. К их числу относятся программы Минобрнауки РФ, Минсвязи РФ, ведомственные программы Минобороны РФ, МВД, ФСТЭК России, отдельных Госкорпораций.

Средства государственного бюджета, выделяемые на выполнение перечисленных проектов в рамках этих программ, значительны. Однако, как показывает практика их реализации, конечные результаты выполнения проектов оказываются неудовлетворительными. Полученные в их рамках средства вычислительной техники и телекоммуникаций, системы автоматизации, математическое и программное обеспечение не удовлетворяют даже требованиям, предъявляемым на начальных этапах формирования этих крупномасштабных программ. Основной недостаток заключается в том, что созданные в их рамках математическое и алгоритмическое обеспечение и программные средства оказываются недостаточно функциональными, неотчуждаемыми от разработчиков и низкоэффективными. Без активного участия разработчиков такие средства не поддаются:

— модификации под перманентно изменяющиеся в силу объективных причин требования к ним со стороны пользователей;

— интеграции в составе территориально распределенных автоматизированных комплексов аналогичного назначения;

— верификации на предмет выполнения требований, изначально предъявляемых к их функциональным характеристикам и к качеству выполнения функций, включая требования по обеспечению безопасности и надежности.

Использование созданных в рамках выполнения проектов государственных программ аппаратно-программных средств и систем на практике создает дополнительные трудности, затраты на преодоление которых соизмеримы, а иногда и превосходят те, которые ушли на их разработку. К числу таких относятся, например, средства и системы, созданные в течение 2002—2010 гг. в рамках проектов ФЦП "Электронная Россия". Они не только наследуют все перечисленные выше недостатки, но и не поддерживают на современном уровне многие функции, которые перечислены в целевых установках на выполнение этой программы.

Выходом из сложившегося положения является переход к формированию программ национального масштаба и заданий на основе принципов, которые:

— ориентируют разработчиков на создание и внедрение в практику реальных секторов экономики опережающих традиционные по современным меркам инновационно-перспективных методов и средств, систем автоматизации;

— изначально предполагают использование положений "Программной инженерии" на всех этапах жизненного цикла программного обеспечения таких систем, которые, кроме перечисленных выше, учитывают экономические факторы [4], психологические аспекты, связанные с коллективом исполнителей проектов [11], и т. п.

Для того чтобы лучше понять мотивы эволюции и суть изменений требований к процессам, сопровождающим разработку, эксплуатацию и модернизацию программного обеспечения, сделаем краткий экскурс в историю развития вычислительной техники.

Эволюция методов и средств разработки и сопровождения программ

Разработка программ и их сопровождение как отдельное направление деятельности человека появилось еще в индустриальном обществе в конце 1940-х — начале 1950-х гг. вместе с появлением первых вычислительных машин — ЭВМ. К их числу относятся ENIAC, Uniac, IBM 701 (США), EDSAC (Англия), БЭСМ-1, Стрела, М-2 (СССР), которые с полным правом можно отнести к электронным, в отличие от их предшественников, использующих электромеханические элементы. Не останавливаясь на детальном описании эволюции этой деятельности — от первых ЭВМ до современных вычислительных комплексов высокой (терафлопной = 10^{12} операций в секунду) и сверхвысокой (петафлопной = 10^{15} операций в секунду) производительности по меркам настоящего времени и экзафлопной (10^{18} операций в секунду) производительности в будущем, так как эта эволюция достаточно подробно изложена в целом ряде учебных и научных изданий (например в работах [12—14]), кратко остановимся на общих особенностях, характеристиках методов и средств, использующихся на разных ее этапах.

Первым этапом в процессе эволюции ЭВМ и программирования, как направления деятельности человека, связанной с разработкой и сопровождением программ, **принято считать 1950-е гг**. Заметим, что термин программирование здесь и далее будет рассматриваться более широко, чем кодирование — описание алгоритма выполнения задачи (вычислений) на языке, понятном ЭВМ. В 1950-е гг. число ЭВМ в мире измерялось десятками. Это были, как правило, ламповые (построенные на радиолампах) ЭВМ с различным (оригинальным) набором функциональных элементов, схем и способов их объединения. Эти ЭВМ обеспечивали быстродействие до 10 тыс. операций в секунду. Серьезного влияния на экономику на этом этапе ЭВМ не оказывали.

Разработка программы, решающей прикладную задачу, представляла собой творческий процесс и требовала от программиста хорошего знания архитектуры и технических возможностей ЭВМ. Причина в том, что программист, составивший программу, должен был сам садиться за пульт управления ЭВМ и проводить вычисления. Отладка программы при такой организации вычислительного процесса была очень трудоемкой и занимала более половины времени жизни самой программы. Класс программиста в значительной степени определялся умением в режиме

реального времени, выделенного ему на ЭВМ, находить ошибки и исправлять их. Потребностей в разработке каких-либо требований к программам, процедурам их разработки и выполнения не возникало. Однако уже на этом этапе появилась потребность в средствах автоматизации процедуры разработки программ, в первую очередь — в более эффективных языках программирования, в упрощающих отладку программ средствах и, как следствие, увеличивающих время их полезного использования.

Вторым этапом в развитии ЭВМ условно принято считать **1960-е гг**. Уже к середине 1960-х гг. в мире появились десятки тысяч транзисторных ЭВМ на твердых схемах с быстродействием до сотен тысяч операций в секунду. В их числе и мини- и микро-ЭВМ с меньшей функциональностью и производительностью, рассчитанные на возможность их эффективной адаптации и использования в отдельных секторах экономики. Были расширены их возможности по вводу-выводу, увеличены объемы и оперативной, и внешней памяти. Отметим в их числе и машины советского производства Мир, Наири, ЭВМ серии Минск (Минск-2, 22, 22-М, 32), БЭСМ-6. В этот период уже стали формироваться потребности в использовании ЭВМ для автоматизации процессов в области государственного управления (статистики), материального производства. Однако их основной сферой применения оставались исследовательский сектор и наукоемкое производство.

Улучшилось программное обеспечение этих машин. Появились трансляторы с алгоритмических языков, широкий, по тем временам, набор библиотек, более развитые средства отладки и управления режимом выполнения задачи. Программисты, как правило, уже не допускались в машинные залы. Появились первые признаки, демонстрирующие элементы отчуждения программы от ее разработчика. Однако по прежнему программирование, особенно для мини- и микро-ЭВМ, представляло собой сложный процесс, требующий от математика-программиста и креативного мышления, и глубокого знания возможностей аппаратуры.

Третье поколение ЭВМ и вычислительных комплексов на их базе, которое принято связывать с **1970-ми гг.**, создавалось уже на интегральных схемах и обеспечивало быстродействие до десятков миллионов операций в секунду. Число таких установок к концу 1970-х гг. измерялось сотнями тысяч. Многие функции, ранее выполнявшиеся программным путем, теперь были возложены на аппаратуру. Получили развитие системы прерываний, механизмы расширения и разделения памяти, ее защиты. Характерной особенностью этого периода стало появление семейств универсальных ЭВМ для массового применения, имеющих схожие архитектурные решения. К их числу относятся, например, семейства IBM, ЕС ЭВМ. Такие решения уже отвечали потребностям автоматизации процессов не только в научных исследованиях и технических разработках, но и в различных секторах экономики, включая непроизводственную сферу. Вычислительная техника становилась реальной производственной силой.

Разработка программного обеспечения для ЭВМ третьего поколения, его эксплуатация и модернизация в силу массового характера их использования стали представлять собой отдельные процессы в плане используемых для их реализации методов и средств, в которых участвуют различные и немалые научные и инженерно-технические коллективы. Появилась необходимость изучения, систематизации и унификации деятельности этих коллективов. Результатом такой унификации стали основанные на опыте разработки и внедрения в практику, эксплуатации по назначению и поэтапной (перманентной) модификации программных средств нормативы и рекомендации, унифицирующие и регламентирующие деятельность математиков-программистов, инженеров-администраторов, сопровождающих программное обеспечение и модифицирующих его по мере необходимости. Как ответ на такие потребности в СССР в эти годы по аналогии со стандартами конструкторской документации начала разрабатываться Единая Система Программной Документации (ЕСДП).

Для **четвертого поколения** вычислительных систем **(1980-е гг.)** характерно то, что они строились на основе больших интегральных схем. Их быстродействие доходило до сотен миллионов операций в секунду. Стали активно внедряться многопроцессорные вычислительные комплексы с общей памятью и общим полем внешних устройств. Элементная база позволяла коллективно использовать разные ЭВМ, объединенные сетями передачи данных. К высокопроизводительным вычислительным установкам этого поколения относятся суперкомпьютеры серии i PSC (*Intel Personal Super Computer*), Cray Y-MP, а также системы Эльбрус-1 и Эльбрус-2 советского производства. Подобные комплексы стали широко использоваться не только для автоматизации процессов в наукоемких или требующих больших вычислительных ресурсов секторах экономики, но и в других сферах деятельности человека.

Для программного обеспечения ЭВМ четвертого поколения характерно использование языков высокого уровня, в том числе предметно-ориентированных. Продолжилось развитие средств автоматизации процедур разработки и отладки программ, механизмов разделения ресурсов в режиме их коллективного использования. В целом программные средства этого времени значительно усложнились. Издержки от ошибок в программном обеспечении, особенно для систем, поддерживающих критически важные объекты, стали очень высоки. Использование вычислительных комплексов в режиме удаленного доступа повысило вероятность деструктивных воздействий как злоумышленных, так и непреднамеренных. Как следствие перечисленных выше причин, затраты на производство программного обеспечения таких комплексов уже намного превышали стоимость оборудования. Требования к программным продуктам стали еще более жесткими. Работы в области его стандартизации продолжились. В США, например, эту деятельность возглавили Институт инженеров по электротехнике

и электроники — IEEE и Институт программной инженерии — SEI.

Отличительной особенностью **вычислительных комплексов начиная с 1990-х гг.** [14] является массовый параллелизм. Прогресс микроэлектронной техники, появление элементной базы на основе сверхбольших по меркам того времени интегральных схем, доступных широкому кругу разработчиков, определили тенденции развития вычислительной техники на очередном, пятом (в принятом нами делении) этапе. В отличие от преобладающих на рынке суперкомпьютерных систем до середины 1980-х гг. многокомпонентных, специализированных векторно-конвейерных процессоров, на рынке информационных технологий в достаточном количестве появились серийно выпускаемые процессоры, такие как Pentium III (Intel), Alpha (DEC), Ultra Sparc (Sun), Power PC (IBM) и др. Как следствие, произошел переход к выпуску на их базе высокопроизводительных систем. Эту тенденцию подкрепила принятая в США в 1995 г. программа ускоренной стратегической компьютерной инициативы — ASCI, определившая на десятилетие вперед общие черты промышленных стандартов, которым должны были соответствовать суперкомпьютерные комплексы. Вслед за ASCI в рамках программы "Интернет следующего поколения" (NGI) в США [15] через отдельные проекты, такие как Internet-2, NREN (NASA), ES Net (DoE) и другие, также поддержанные государством, стали создаваться высокоскоростные сети связи национального масштаба. Активно в эти процессы включилась и Россия [16]. Одним из первых проектов создания высокопроизводительной национальной сети с прямым выходом в Internet-2 стал российско-американский проект Mirnet [17].

Одновременное с развитием вычислительных систем и беспрецедентно быстрое **в 1990-х гг.** развитие технологий пакетных коммуникаций и метасети Интернет на их основе, высокие скорости передачи данных, которые становились сопоставимы со скоростями передачи по внутренним каналам связи массово-параллельных вычислительных установок, открыли **перспективы создания высокопроизводительных кластерных систем**. Как результат активных действий на этом направлении появились проекты создания распределенных информационно-вычислительных метакластерных комплексов на основе технологий Grid. Следуя опыту США, были приняты аналогичные программы и предприняты действия в Канаде, Европе, Японии и ряде других стран.

Одна из тенденций, начало которой было положено еще в середине 1990-х гг., заключалась в создании распределенных на сетевой среде информационно-вычислительных комплексов, объединяющих высокопроизводительные ресурсы, которые сопровождаются различными организациями. Целью такого объединения, как правило, являлось решение одной или класса задач, представляющих интерес для этих организаций. К их числу относились сложные наукоемкие задачи вычислительного характера, управления данными, автоматизации технологических процессов. Основой для начала активных работ на этом направлении стали высокопроизводительные параллельные вычислительные системы и высокоскоростная телекоммуникационная инфраструктура, которая окончательно сложилась в США, Канаде, Европе и Японии. Поддержка исследованиям и выполнению практически значимых проектов в этой области на уровне государства в 2004 г. была продемонстрирована в США, где была объявлена президентская стратегическая программа в области Grid (*Strategic Grid Computing Initiative*). Вслед за США аналогичные действия были предприняты и в других странах.

Переход к широкому применению многопроцессорных вычислительных комплексов, в том числе территориально распределенных и не всегда однородных по используемым платформам, в еще большей степени усложнил создание для них эффективного программного обеспечения. И без того сложная технология параллельного программирования поставила перед системными программистами ряд новых задач. Одна из них связана, как правило, с относительно медленным ростом производительности для решения практически значимых, настоятельно требующих распараллеливания классов приложений.

Вторая задача обусловлена сложностью переноса программ с одних установок и комплексов на другие, тем более на распределенные кластерные системы. В какой-то степени эта задача решается с использованием разработанных для этих целей стандартизованных средств распараллеливания, таких как PVM (параллельная виртуальная машина), MPI (интерфейс передачи сообщений), Open MP (интерфейс прикладных программ) для систем с распределенной памятью. Однако эти средства очень низкоуровневые и, в отличие от традиционных императивных языков высокого уровня, требуют от разработчиков прикладных программ глубоких программистских знаний. По этой причине многие из программ, в том числе критически важных для национальной экономики больших программных комплексов, написанных ранее на императивных языках, оказываются "неподъемными" для их модификации в целях использования на параллельных комплексах.

Наметившийся **в 2000-х гг. переход** к более эффективным, в том числе путем специализации, **высокопроизводительным вычислительным системам смешанной архитектуры**, включающим и массово-параллельную составляющую, и средства ускорения вычислений за счет внедрения проблемно-ориентированных на более мелкие, но часто встречающиеся по ходу решения задачи устройства с программируемой логикой (ПЛИС/FPGA) и видеоускорителей, в еще большей степени усложнили процессы разработки программного обеспечения.

Описанные ранее этапы эволюции средств вычислительной техники и систем на их основе привели к трудностям, которые возникают в процессе разработки, сопровождения и модернизации (модификации) программного обеспечения. Увеличение сложности программных комплексов с позиций выполняемых ими функций, объемов кода, в плане архитектурных и

технологических решений, привели к необходимости привлечения достаточно больших коллективов специалистов к разработке таких продуктов. Практика их создания показала, что это не только большие коллективы под административно-техническим управлением, как правило, крупных коммерческих (бизнес) фирм, подобных IBM, Microsoft, Sun и др., которые инициируют и финансируют эти разработки. Созданные такими фирмами продукты, как правило, предлагаются на рынок информационных технологий и используются потребителями без предоставления им открытых исходных кодов (являются проприетарными), авторское право на которые принадлежит фирмам-разработчикам. С середины 1990-х гг. в разработку программного обеспечения все более активно стали включаться отдельные программисты и коллективы, которые формировались на безвозмездной (по крайней мере на начальном этапе разработки) основе, выполняющие работу с использованием технологий интерактивного взаимодействия в Интернет. Подобные способы создания программного продукта в режиме "Открытого проекта" (*Open Source*) имеют свои преимущества, так как позволяют аккумулировать идеи и решения большого числа заинтересованных разработчиков. Однако им присущи и недостатки, обусловленные отсутствием централизованного администрирования, четких требований к целевому продукту и, как следствие, к контролю за промежуточными и конечными результатами. Созданные в ходе таких проектов программные продукты, как правило, передаются разработчику с открытыми исходными кодами. Последнее обстоятельство упрощает их модификацию с учетом условий эксплуатации, однако "размывает" ответственность разработчиков за конечный результат. Следует отметить, что существуют и другие способы разработки программного обеспечения, обладающие особенностями, характерными как для первого, так и для второго подходов к разработке. К настоящему времени сложилась некоторая система классификации как таких подходов, так и особенностей, регламентов их реализации [18].

Каждый из перечисленных выше способов разработки программного обеспечения оказывает влияние и на процессы его внедрения в практику и сопровождения, в которых также, как правило, участвуют большие коллективы специалистов разного профиля. Не останавливаясь на особенностях этих процессов и возникающих при их реализации трудностях, отметим, что данные факторы в еще большей степени стимулируют и активизируют работу по систематизации требований и регламентации деятельности всех специалистов, участвующих в разработке, сопровождении программного обеспечения и в его адаптации под потребности пользователей. В этой работе появляются новые подходы, методы и средства, которые определяют будущее Программной инженерии.

От инженерно-эвристических подходов к фундаментальной программной инженерии

Рассмотрим **сложившиеся подходы и факторы, сдерживающие развитие Программной инженерии**. В традиционной трактовке Программная инженерия [4, 5] рассматривается как методология разработки и внедрения в практику, сопровождения и модернизации программных средств и комплексов программ с использованием систематизированного набора стандартов, представляющих собой перечень рекомендаций, выполнение которых призвано обеспечить реализацию заданных заказчиком (или сформулированных с его участием) на всех этапах жизненного цикла требований:

— по функциям, которые программное обеспечение призвано выполнять по основному его назначению;

— по обеспечению защиты от реализации различных (внутренних и внешних, злоумышленных и непреднамеренных, нарушения регламентированного доступа к данным и ресурсам, целостности данных и др.) угроз собственно программного обеспечения, средств вычислительной техники и других ресурсов (включая информационные активы) автоматизированных систем, с которыми оно взаимодействует;

— по обеспечению качества выполнения всех перечисленных выше функций (в том числе высокой функциональности, надежности, удобства для пользователя, масштабируемости, мобильности и других показателей качества).

Применение положений программной инженерии для отдельного программного продукта обеспечивается профилем стандартов его жизненного цикла. К основным этапам жизненного цикла программного продукта принято (с той или иной степенью точности и используемой нотации) относить следующие.

1. Предпроектные исследования, формирование предварительного технического задания (требования — спецификации, модель, макет и результаты апробации).

2. Проектирование.

2.1. Разработка технического задания (требования по функциям, включая обеспечение безопасности; требования по качеству; профиль стандартов жизненного цикла, программа испытаний, а также ряд других положений).

2.2. Разработка технического проекта, рабочего проекта.

2.3 Разработка и реализация программного кода.

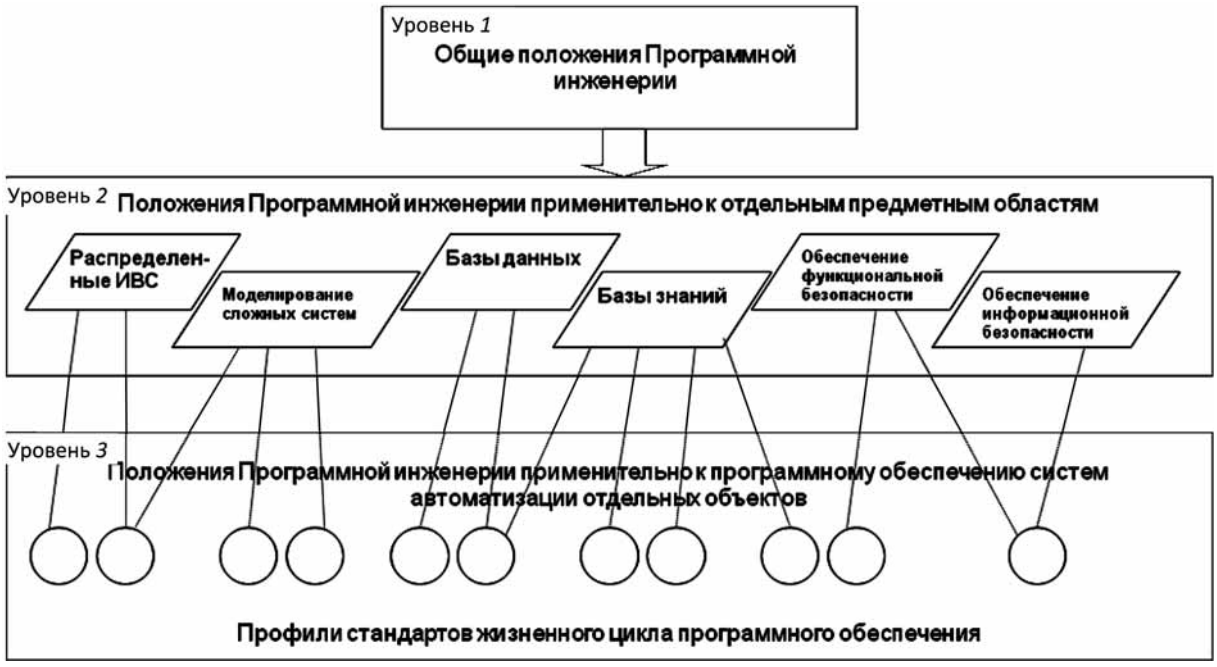
3. Отладка, испытания, сертификация и сдача — приемка в эксплуатацию.

4. Сопровождение и модернизация.

5. Вывод из эксплуатации.

Общая схема реализации положений Программной инженерии условно представлена на рисунке.

Согласно представленной схеме, для сопровождения программного продукта на каждом этапе его жизненного цикла необходимо использовать рекомендации Программной инженерии, соответствующие трем уровням их реализации, которые с разных позиций



характеризуют продукт. Кратко, к такого сорта характеристикам можно отнести перечисленные далее.

- Чтобы сформировать (и затем реализовать) профиль стандартов жизненного цикла программного продукта применительно к конкретному объекту, нужно квалифицированно (со знанием предмета): проанализировать требования, предъявляемые к программному продукту вообще и соотнести их с общими рекомендациями (положениями на уровне *1*); отобразить эти требования и рекомендации на класс (тип) продуктов (уровень *2*), к которому относится данный продукт и объект автоматизации; принять во внимание следующие из анализа на предыдущих уровнях особенности целевого программного продукта, объекта автоматизации и среды окружения (уровень *3*);
- На уровне *1* необходимо использовать традиционные (общие) положения Программной инженерии, которые, к сожалению, в основном изложены на вербальном уровне их описания.
- На уровне *2* должны использоваться рекомендации, которые в последние годы (≈ 20 лет применительно к отдельным востребованным практикой предметным областям) активно разрабатываются на основе положений уровня *1*. Здесь необходимо отметить тот факт, что на схеме представлен далеко не полный перечень предметных областей и соответствующих им классов программных продуктов, которые уже достаточно подробно исследованы и стандартизированы.
- Уровень *3* пока слабо исследован и систематизирован, особенно в его связи с положениями вышележащих уровней *2* и *1*.

Профиль жизненного цикла отдельного программного продукта определяется набором стандартов, соответствующих именно этому продукту в принятых условиях его эксплуатации (среды окружения). Разработка такого профиля представляет собой отдельную фазу на этапе проектирования, требующую от участвующих в этом процессе экспертов не только опыта в сфере программной инженерии, но и глубокого знания в области назначения программного продукта. Более того, профиль стандартов жизненного цикла программного продукта, особенно предназначенного для автоматизации сложно организованных (в архитектурно-технологическом плане) процессов и объектов, по объективным причинам претерпевает изменения. Отметим, что в оценке влияния таких изменений на свойства продукта также необходимо участие квалифицированных экспертов.

Принимая во внимание большую работу, проделанную в течение последних 20 лет в области разработки и систематизации стандартов программной инженерии, следует, однако, подчеркнуть, что их применение на практике в настоящее время наталкивается на ряд трудностей даже в технологически развитых странах. Не претендуя на полноту представления (перечисления) всех таких трудностей, выделим следующие, имеющие более важное значение в контексте целей настоящей публикации.

- Программное обеспечение средств вычислительной техники и автоматизированных систем в массовом сознании не стало еще продуктом индустриального производства и применения. Сопровождающие его на всех этапах жизненного цикла стандарты программной инженерии не представляют собой набор "жесткосвязанных" требований, отдельных строго формализованных положений, которые нужно неукоснительно выполнять применительно к конкретному программному продукту. Это в большей степени перечень рекомендаций, целесообразность и экономическая эффективность применения которых в каж-

дом отдельном случае требует глубокого анализа со стороны квалифицированных экспертов.

- Большинство рекомендаций, составляющих содержание стандартов программной инженерии, описаны на вербальном уровне их представления и плохо поддаются формальным (математическим) способам описания. Как следствие, автоматизация процессов (процедур) анализа в целях определения (оценки) целесообразности их применения и определения рисков неоправданно высоких затрат в случае выполнения рекомендаций для конкретного объекта оказывается сложно реализуемой.
- В силу отсутствия математического обеспечения и средств автоматизации, оценивание (процесс оценки) эффективности применения того или иного стандарта каждого из отмеченных выше классов и формирование профиля стандартов жизненного цикла программного продукта требуют от эксперта глубоких знаний и в программной инженерии, и в предметной области, для которой такой продукт предназначен, и креативного мышления. Подготовка специалистов, обладающих перечисленным набором теоретических знаний и практических навыков, требует многих лет и разнопланового обучения. Как следствие, таких кадров не хватает.
- Перечень сложившихся на настоящее время стандартов Программной инженерии не полностью покрывает все этапы жизненного цикла программного продукта. Разработка и принятие новых стандартов отстает от современных темпов развития средств вычислительной техники, телекоммуникаций и, как следствие — от потребностей практики, особенно в части программного обеспечения систем автоматизации сложно организованных, распределенных объектов, которые остро востребованы реальной экономикой.

Трудности использования положений Программной инженерии на практике сдерживают ее развитие как области научно-технических знаний, в том числе — подготовку необходимых для этого кадров, способных эффективно работать на всех трех, представленных на рисунке уровнях реализации положений Программной инженерии. Возникает некоторый замкнутый круг взаимообусловленных причин. Сложившееся положение дел прямо влияет на темпы разработки конкурентоспособного на мировом рынке информационных технологий программного обеспечения средств и систем автоматизации технологических процессов в разных, в том числе критических для экономики секторах. Последнее обстоятельство не позволяет подготовить необходимые для этого кадры и отрицательно влияет на национальную экономику в целом.

Проанализируем существующие **вызовы и перспективные подходы в инженерии программ**, способные изменить ситуацию. Как следствие перечисленных выше трудностей, даже в среде специалистов бытует мнение, что сложившиеся на настоящее время основы Программной инженерии потеряли свою актуальность по отношению к сложно организованному программному обеспечению объектов и процессов, где ее применение необходимо (целесообразно). В качестве альтернативы предлагается разработка новых, основанных на более формальных методах описания программного продукта подходов, поддающихся верификации в автоматизированном режиме. Принимая во внимание изложенные выше соображения, в качестве первоочередных действий, способных изменить сложившееся положение дел с развитием Программной инженерии как области научно-технических знаний, а также с расширением сферы ее применения на всех этапах жизненного цикла программ, следует рассматривать подготовку кадров должной квалификации и их активное включение в разработку математического и программного обеспечения средств и систем автоматизации, сопровождающих программный продукт на всех этапах его жизненного цикла. О возрастающем понимании вклада Программной инженерии в процессы экономического развития страны и важной роли кадрового потенциала в решении возникающих на этом пути задач свидетельствует открытие в России в 2009 г. образовательного стандарта "Программная инженерия", создание научно-технического журнала "Программная инженерия", большое внимание к нему со стороны научного и образовательного сообщества и хорошая корреспондентская география журнала.

Одним из главных мотивов, определяющих необходимость действий, направленных на более активное использование положений Программной инженерии на всех этапах жизненного цикла программного обеспечения систем автоматизации объектов, значимых для промышленного сектора экономики России, в первую очередь критически важных, является высокая степень изношенности ее основных фондов. Это обстоятельство — причина многочисленных отказов в работе технических систем, последствия которых приводят к ситуациям чрезвычайного характера, к значительным издержкам как экономического, так и неэкономического, в том числе политического плана. В сложившейся ситуации объективно необходима скорейшая (опережающая) модернизация таких фондов, основу которой в современном (информационном) обществе составляет автоматизация технологических процессов, поддерживающих подобные объекты. На настоящее время эти вопросы актуальны не только для энергетики, особенно атомной, авиационного и ракетно-космического сектора, для оборонного комплекса и управления крупными транспортными потоками, но и для других сфер национальной экономики. Следует однако подчеркнуть, что эти действия не должны быть направлены на замену существующих положений, методов и средств, годами сложившихся в Программной инженерии. Целью является их развитие и модернизация (совершенствование) путем применения современных подходов к их реализации, максимального использования математического аппарата и формальных моделей, описывающих семантику программ, поддающихся надежной верификации в автоматизированном режиме. В рамках настоящей публикации хотелось бы отметить именно эту сторону Программной инженерии, способную, по мнению автора, расширить сферу применения ее положений и усилить прикладное значение.

В рамках традиционных подходов, с позиции рекомендаций Программной инженерии, математические (формальные) модели, как правило, представляют интерес (практическую ценность) только на этапе предпроектных исследований, предоставляя материал (базовые сведения) для составления профиля стандартов жизненного цикла программного продукта. Однако такое положение дел не соответствует действительности. Причина в том, что на этапах разработки и внедрения в практику, сопровождения и модернизации программного обеспечения сложно организованных систем происходят перманентные коррекции технических требований к нему, изменение профилей стандартов и других атрибутов. Такие коррекции способны привести к изменению функциональных возможностей программной системы, качества исполнения ими функций, появлению у них тех или иных уязвимостей, других негативных последствий. В случае любой из перечисленных коррекций необходима оперативная проверка соответствия требований к модифицированной системе и ее свойств тем требованиям и свойствам, которыми она обладала до модификации. Математические модели являются самым эффективным средством оперативной проверки такого соответствия, позволяя либо путем получения аналитических оценок, либо в ходе имитационного моделирования хотя бы в первом приближении дать ответы на возникающие вопросы. Таким образом, обеспечивается перманентный (постоянный и оперативный) контроль за корректностью (соответствием требованиям) программной системы на протяжении ее жизненного цикла.

Прикладную значимость отмеченных мер попытаемся продемонстрировать на перечисленных далее задачах, решение которых напрямую влияет на развитие российской экономики. Не претендуя на полноту освещения всех вопросов, которые возникают на этом направлении, остановимся на подходах (моделях, методах и средствах), которые, во-первых, позволяют эффективно контролировать (анализировать) функциональные возможности, качество выполнения функций программными средствами и обеспечение ими требований безопасности, во-вторых, активно используют для этого математические модели, семантические технологии и, в частности — представления в виде онтологий.

При этом еще раз отметим, что традиционные натурные и полунатурные тестовые испытания, особенно на заключительных стадиях разработки программного продукта и после крупномасштабных модификаций, не только не исключаются, а являются необходимыми. Однако на стадиях предпроектных исследований, в ходе проектирования, в процессе разработки и поэтапной модификации программ, в том числе и в процессе подготовки и проведения тестовых испытаний, одной из важнейших задач является верификация на основе формальных моделей. Такой подход позволяет оперативно, в том числе в итерационном режиме, получать как предварительные аналитические оценки, так и более точные оценки по результатам вычислений с использованием имитационного моделирования. Преимущества и недостатки традиционных натурных, полунатурных тестовых испытаний и математического моделирования для сложно организованных программных комплексов систем автоматизации научно-технических исследований в аэрокосмической отрасли, атомной энергетике, машиностроении обсуждаются достаточно давно (например, в работах [19—22]). В последние годы эти вопросы стали активно обсуждаться применительно к медицине, управлению большими транспортными потоками, к другим секторам экономики. С развитием средств вычислительной техники и систем автоматизаций на их основе, соотношения в пользу целесообразности и эффективности использования тех или других подходов изменяются. Необходимо однако отметить, что в последние годы все более настоятельно ставятся вопросы о создании строгих математических моделей, позволяющих в автоматизированном режиме проводить тестовые испытания и адекватно оценивать их результаты.

Способы верификации программных средств на основе математических моделей, позволяющие оперативно получать характеризующие их свойства, предварительные (хотя и, как правило, грубые) аналитические оценки, обсуждались и развивались в гораздо меньшей степени. В первую очередь это связано со сложностью построения моделей, адекватно описывающих семантику процессов, подлежащих автоматизации с помощью контролируемых (подконтрольных) программных средств. Построение таких моделей на основе операционного, детонационного и логического подходов имеют свои преимущества и недостатки. К числу работ на этом направлении, применительно к операционным механизмам можно, например, отнести [23, 24], к предметно-ориентированному программному обеспечению — результаты, изложенные в работе [25]. В меньшей степени для этих целей пока используются денотационные и логические модели.

Другим способом формального описания семантики сложно организованных объектов и процессов, подлежащих автоматизации с помощью программных средств, является аппарат (механизмы) онтологий. На настоящее время известен опыт использования такого аппарата к описанию семантики процессов в информационно-поисковых системах (*Semantic Web* [26, 27]), в распределенных Grid-структурах (*Semantic Grid* [28]). К российским работам в области применения аппарата онтологий для разработки прикладного программного обеспечения можно отнести работу [29].

С позиции использования перечисленных подходов для верификации программных средств важнейшими являются точность (выразительные возможности) описания семантики подлежащих автоматизации с их помощью процессов и эффективность конвертации (преобразования) такого описания на языки программирования, понятые ЭВМ. Другим важным результатом использования подобных моделей является предоставляемая ими возможность оперативно контролировать влияние перманентных модификаций на требования

технического задания. К числу поисковых и практических исследований на этом направлении можно отнести результаты в области применения формализма среды радикалов [30] для описания сложно организованного прикладного программного обеспечения [31]. Дальнейшие исследования этих вопросов могут внести значительный вклад в разработку методов и средств программной инженерии, в верификацию программных продуктов на всех этапах их жизненного цикла.

Для иллюстрации возможностей реализации отмеченных выше **перспективных подходов**, рассмотрим некоторые задачи и примеры их решения **российскими исследователями**. Одним из важнейших этапов в жизненном цикле программного обеспечения, определяющим соответствие его свойств предъявляемым к нему требованиям (спецификациям), является тестирование. В конечном счете, именно по результатам тестовых испытаний определяется качество произведенного программного продукта и возможности его использования по назначению. Для сложно организованных программных комплексов длительность этого этапа может быть соизмерима и даже больше, чем все предшествующие стадии создания.

Однако тестовые испытания и натурное моделирование как методы в процессе разработки программного обеспечения имеют свои недостатки. Главные из них: отсутствие доказательной строгости и "полноты покрытия" программы испытаний; трудности составления и реализации программы испытаний для сложно организованных объектов. Как итог — сложности сопоставления результатов тестовых испытаний, полученных разными группами исследователей. Как следствие, необходимость повышения эффективности таких испытаний, улучшения качества программного продукта в сочетании с потребностью в снижении затрат и сокращении времени на проведение тестовых испытаний диктуют необходимость их автоматизации.

Эффективным способом решения перечисленного (отмеченного) выше комплекса вопросов является тестирование на основе формальных (математических) моделей. Концептуальная схема такого подхода основана на проверке того, что модель тестируемой системы соответствует модели требований (спецификации). Первые исследования на этом направлении начали проводиться еще с 1970-х гг. [32]. Предложено несколько подходов к установлению такого соответствия (конформности — отношения "похожести") моделей, поддающихся выполнению в автоматизированном режиме, когда по спецификации генерируются тесты, позволяющие оценить степень конформности. Однако сложившиеся в рамках этих подходов методы и средства, во-первых, в значительной степени ориентированы на семантику, являющуюся вторичной по отношению к первичной, т. е. общей семантике тестовых испытаний отдельной проблемной области, подлежащей автоматизации с использованием разрабатываемых программных средств. Это создает дополнительные трудности использования таких методов и средств тестирования применительно к программам для других приложений. Во-вторых, эти подходы, как правило, оказываются нереализуемы на практике для тестирования сложных систем, состоящих из большого числа иерархически организованных компонентов. Последнее обстоятельство связано как с корректностью декомпозиции, так и с повышением вероятности ошибок в отдельных компонентах. Заметим, что задача верификации декомпозиции требований в настоящее время общего решения не имеет.

Частично, для класса конформностей типа редукции, семантика которых определяется набором допустимых тестовых воздействий и возможных наблюдений за поведением программной системы, теоретические вопросы решены и обобщены в докторской диссертации И. Б. Бурдонова (ИСП РАН [33]). В качестве модели адекватного описания поведения сложных программных систем в этой работе выбрана система помеченных переходов LTS (*Labelled Transition System*). Предлагается концепция безопасного тестирования, которая позволяет избежать ненаблюдаемых отказов, дивергенции и разрушения системы. Рассматривается асинхронное тестирование, при котором взаимодействие теста и реализации происходит не напрямую, а через конформности при композиции. Дается полное решение задачи верификации декомпозиции системных требований. Спецификация составной системы оказывается согласованной со спецификациями компонентов тогда и только тогда, когда ей конформна композиция преобразованных спецификаций. Результаты исследований на практике сводятся к методам генерации тестов и преобразования спецификаций. Принимая во внимание важную роль рассматриваемой работы, следует, однако отметить, что больших, практически значимых результатов в направлении разработки доказательно строгих формальных моделей, адекватно описывающих процессы тестовых испытаний, пока не получено и многие вопросы остаются открытыми.

Как отмечалось в предыдущем разделе, вопросы эффективного программирования сложных систем, обеспечения должной функциональности и высокого качества выполнения функций стали активно исследоваться, начиная с 1970-х гг. Одним из первых, отвечающих на эти потребности подходов, стало использование библиотек программ, наиболее часто используемых для решения широкого класса задач. Однако важнейшей на этом направлении была и остается в настоящее время задача создания адекватных потребностям практики моделей и методов, программных механизмов автоматизированного накопления и использования программистских знаний. Исследования на этом направлении особенно важны для разработки сложных программных комплексов в режиме реализации крупных проектов с участием территориально распределенных на сетевой среде групп исполнителей, в первую очередь — *open source*-проектов. С одной стороны, это систематизация, хранение типовых организационно-технических мер и действий технологического плана, направленных на эффективное использование уже существующего программного обеспечения и ассоциированных с ним процессов производства, сопровождения и развития (модернизации).

К их числу, например, относятся средства, позволяющие в автоматизированном режиме поддерживать положения Программной инженерии на всех этапах разработки программ, включая средства эффективного описания спецификаций, верификации, тестирования и др. С другой стороны, важными для программиста знаниями являются информационные активы, содержащие сведения о большом количестве сущностей, которые приходится учитывать и анализировать при разработке, в ходе эксплуатации и модернизации сложных программных комплексов, предназначенных для массового использования на различных платформах (операционных средах). С третьей стороны, для создания и развития программных средств, поддерживающих совместную работу территориально распределенных коллективов, крайне важным, хотя и сложно реализуемым является выбор отвечающих их потребностям сервисов, которые разработаны и/или получены ранее другими исследователями (группами) и в той или иной форме хранятся и доступны на сетевой среде.

Исследования и практические разработки по каждому из трех отмеченных выше направлений проводятся в мире и, в частности, в России. К числу отечественных исследований на первом из них можно отнести работы, выполненные в 1990-х гг. группой исследователей под руководством В. А. Галатенко в НИИСИ РАН [34]. Их итогом стало создание (с использованием объектного подхода) инструментальной среды программирования ЭКСКОРТ. В ее основе: формальная модель в виде аппарата схем, отображающая семантику процесса накопления и использования знаний на разных этапах жизненного цикла программ; специализированный высокоуровневый язык; объектно-ориентированная СУБД; средства редактирования программ; инкрементальный анализатор программ. Модель программы, с которой взаимодействуют все компоненты ЭКСКОРТ, при таком подходе представляется в виде ориентированного графа объектов, таких как константа, переменная, тип, процедура, пакет со своими характеристиками. Еще одним примером решения задачи на первом из перечисленных выше направлений является работа А. И. Гирчи, выполненная им в НИИ механики МГУ в 2004—2008 гг. [35]. Ее целью было создание инструментальной среды для моделирования сложно организованных программных комплексов в области аэромеханики на основе метода вязких вихревых доменов, позволяющая в рамках положений Программной инженерии повторно использовать знания, ранее полученные программистами на всех этапах жизненного цикла аналогичных систем.

В качестве примера теоретически и практически значимой задачи на упомянутом выше втором направлении автоматизации программистской деятельности, связанной с систематизацией, хранением и предоставлением данных о сущностях, важных на различных этапах жизненного цикла сложных программных систем, можно отметить создание методов и средств информационно-аналитической поддержки процессов разработки и использования стандартов на интерфейсы ОС Linux. В условиях сотен используемых на настоящее время дистрибутивов на интерфейсы этой ОС, основанных на ядре Linux, библиотеках и технологиях, которые реализуются в рамках проекта GNU, разработка и сопровождение приложений, эффективно переносимых на разные платформы, представляет большие сложности. Она требует анализа широкого спектра интерфейсных сущностей, в числе которых библиотечные функции, описания заголовочных файлов, типов данных, константы и макроопределения. Некоторые из них отражают специфику языков программирования и, соответственно, систем трансляции. Заметим, что весь перечисленный спектр вопросов важен и для средств автоматизации упоминавшегося ранее процесса (стадии) формирования профилей, предназначенных для практически значимых, широких классов приложений.

В исследованиях на этом направлении хотелось бы выделить работы Д. В. Силакова [36, 37], выполненные в ИСП РАН в 2006—2010 гг. В них предложена логическая модель системы интерфейсов приложений под ОС Linux, доказательно обоснована ее корректность (условия успешного запуска приложений под дистрибутивом ОС Linux). Разработана методология создания и развития стандартов на интерфейсы программ под ОС Linux, основанная на исследовании информационно-аналитической системы, автоматизирующей возникающие при этом другие ресурсоемкие процессы.

Повышенное внимание к третьей из отмеченных выше задач связано с активным внедрением в реальную экономику сложно организованных распределенных информационно-вычислительных систем на основе Grid и Cloud — подходов. Не вдаваясь в детали сходства и различия применяемых при их реализации методов и средств, которые изложены, например, в работе [38], хотелось бы отметить, во-первых, определенные неудачи в попытках развивать Grid-системы на основе строгих, традиционных спецификаций [39, 40] и инструментальных средств Clobus Toolkit, требующих "запредельно высокой" квалификации разработчика новых приложений и, соответственно, ограничивающих применение этих средств на практике, во-вторых, проблему поиска (обнаружения) созданных ранее и опубликованных в сети Grid-ресурсов (аппаратных, сервисов, информационных активов), необходимых для выполнения того или иного приложения.

Традиционно используемых механизмов поиска таких ресурсов по ключевым словам для разрешения второй из упомянутых проблем недостаточно. Одним из выходов на этом направлении видится использование онтологий — формального способа предоставления знаний о потенциально востребованных ресурсах с помощью конечного множества понятий и отношений между ними. Такой подход на основе Semantik Grid [28] на настоящее время активно развивается за рубежом. Любые результаты российских исследователей на этом направлении представляли бы несомненный и теоретический, и прикладной интерес, явились бы большим вкладом в управление программным продуктом, предназначенным для сложно организованных распределен-

ных систем на всех этапах его жизненного цикла. А потребность экономики в таких системах на сегодня есть в сфере ресурсоемких научных исследований, в области управления авиационным транспортом, в секторах, связанных с обеспечением общественного правопорядка и национальной безопасности, в других, в том числе ее критически важных секторах.

Отдельного внимания заслуживает имеющая свою специфику и стоящая в инженерии программ особняком **методология создания и сопровождения программного обеспечения информационной безопасности средств и систем автоматизации**. Арсенал методов, направленных на создание программных средств обеспечения безопасности объектов информационно-вычислительной и телекоммуникационной среды, на настоящее время достаточно большой. Они, как правило, основаны:

— на критериальных подходах к разработке, внедрению и сопровождению программных средств обеспечения безопасности информационных технологий;

— на использовании формальных, как правило, математических моделей гарантированно защищенных систем;

— на методах и средствах верификации — проверки соответствия таких программных средств некоторому заданному набору требований (спецификаций), в том числе заданных в виде формальных моделей;

— на тестовых испытаниях (валидации) программных средств обеспечения безопасности в соответствии с заранее принятой программой таких испытаний.

Основы критериальных подходов были заложены в конце 1970-х гг. в связи с исследованиями в области новейших, по тем временам, технологий пакетных коммуникаций, инициированных в США агентством DARPA и новыми вызовами безопасности сетевым ресурсам, которые создаются с использованием этих технологий. К числу широко используемых на практике документов, регламентирующих действие на основе таких подходов в США, Европе и России, можно отнести:

- "Критерии оценки доверенных компьютерных систем" и его интерпретация для сетевых конфигураций (Минобороны США);
- "Гармонизированные критерии Европейских стран";
- "Критерии оценки безопасности информационных технологий";
- "Требования безопасности для криптографических модулей" (США);
- "Руководящие документы Гостехкомиссии России" (ФСТЭК России).

Основным недостатком практической реализации этих подходов являются трудности их применения к сложно организованным системам, которые, как правило, и являются главными объектами защиты с их использованием.

Подходы к разработке средств обеспечения информационной безопасности практически значимых систем на основе их гарантированно защищенных моделей начались с работ, описывающих традиционно применяемые в операционных системах (мониторах безопасности) механизмы дискреционного, мандатного (многоуровневого), а затем и ролевого разграничения доступа. В таких моделях (Take-Grand, Белла — Лападула, MLS, безопасных информационных потоков и др.) нашли отражение особенности широко применяемых политик безопасности, аппаратно-программной платформы, среды окружения и ряд других. К недостаткам этих подходов следует также отнести трудности применения базовых моделей к формальному описанию сложно организованных объектов.

Подходы на основе проверки соответствия программных средств обеспечения безопасности некоторой, изначально заданной спецификации, в том числе в виде математических моделей, включают перечисленные далее методы.

Методы статического анализа:

- поиск пути в графовой модели (достижимость определенного состояния при выполнении заданного набора действий, исследование информационных потоков в моделях логического разграничения доступа);
- семейство методов "*model checking*" (исследование/проверка выполнения свойств, заданных на языке одной из темпоральных логик, в системе переходов между состояниями моделируемого объекта — программы или протокола);
- методы логического программирования (логического вывода в логике предикатов первого порядка) и автоматического доказательства теорем (исследование/проверка требований для компонентов компьютерной системы, специфицируемых декларативным образом);
- методы формальной верификации императивных программ.

Методы динамического анализа:

- тестирование и натурное моделирование (эксперименты с подачей тестовой нагрузки на опытный образец или на натурную модель системы с последующей интерпретацией результатов и обоснованием достаточной полноты тестов, согласно некоторой формальной модели).

Методы имитационного моделирования:

- дискретно-событийное (*event-driven*) моделирование исследуемой системы (анализ свойств больших систем в рамках упрощенной имитационной модели, получение оценок статистических характеристик исследуемой системы).

Вместе с тем, все перечисленные выше методы и, как следствие, реализующие их инструментальные средства имеют недостатки, ограничивающие возможности их использования. Методы статического анализа и имитационного моделирования при их применении к сложно организованным объектам представляют большие трудности при их математическом описании, вычислительные сложности и ряд других. Методы динамического анализа наследуют все недостатки подходов на основе тестовых испытаний, которые отмечены ранее.

Представленные соображения свидетельствуют о том, что сложившиеся на настоящее время методы и средства разработки, сопровождения и модернизации программного обеспечения информационной безопасности сложных систем обладают недостатками,

аналогичными тем, которые перечислены выше применительно к другим автоматизируемым процессам. Как следствие, практическое использование стандартов, предназначенных для обеспечения безопасности информационных технологий, автоматизированных систем, особенно для сложных, распределенных систем, сопряжено с теми же трудностями. Отсюда следует, что подходы к их разрешению должны быть похожими (идентичными). Не останавливаясь на деталях, кратко перечислим подходы, позволяющие устранить узкие места в инженерии программного обеспечения информационной безопасности сложных систем путем использования формальных моделей, позволяющих более точно описывать проблемную область и использовать основанные на этих моделях средства автоматизации процессов сопровождения программ на всех этапах их жизненного цикла.

По аналогии с изложенными ранее соображениями, представляется целесообразным большее внимание уделять разработке программного обеспечения на стадии предпроектных поисковых исследований. Первой целью должно стать формирование, как минимум, перечня предварительных спецификаций и, как максимум, построение некоторых формальных моделей, адекватно описывающих семантику процессов, которые должны быть реализованы разрабатываемым комплексом программ. Такие спецификации и модели должны описывать общие требования политики безопасности подлежащего защите объекта. Они призваны учитывать реакцию программного обеспечения подконтрольной системы на модели потенциальных угроз, уязвимостей и способов их реализации, а также следующие за ними требования безопасности. Подобные спецификации и модели для программного комплекса сложно организованной автоматизированной системы могут представлять собой набор отдельных, составляющих его частей и способы, с должной степенью корректности (адекватности) описывающие процессы взаимодействия этих частей.

Исследования на этом направлении проводятся за рубежом. Некоторые их результаты в виде математических моделей, реализующих их программных комплексов представлены в Интернете [41, 42]. Есть и результаты поисковых исследований российских ученых, к числу которых можно отнести работы [43—45]. Однако анализ представленных на настоящее время подобных инструментальных средств показывает, что даже в применении к относительно несложным в архитектурно-технологическом плане практически значимым автоматизированным системам эти средства пока не могут быть эффективно использованы. Этот факт означает, что основная работа еще впереди.

Второй целью и результатом на стадии предпроектных исследований должен быть макет разрабатываемого программного продукта, демонстрирующий в ходе испытаний высокие степень соответствия функциональных свойств и качество исполнения функций, указанных (сформированных) в предварительных требованиях к продукту. Результаты таких исследований дают объективные основания для начала работ по проектированию сложного программного комплекса, включая: формулировку более точной (детализированной) политики и требований безопасности; техническое задание, включая профиль стандартов жизненного цикла, требования по функциональным свойствам и качеству выполнения функций.

К результатам таких исследований, ориентированных на создание программных механизмов в операционных системах, которые используются в службах (сервисах) управления доступом к ресурсам объекта управления, можно отнести описанные в работах [46, 47]. Модели, отражающие (описывающие) семантику процессов логического разграничения доступа пользователей к ресурсам подконтрольных автоматизированных систем, основаны на использовании современного математического аппарата, достаточно легко алгоритмизируются, эффективно программируются и поддаются оперативной верификации.

К сожалению, приходится констатировать, что на настоящее время стадии предпроектных исследований при разработке практически значимых, в том числе критически важных программных средств уделяется очень мало внимания, либо эта стадия вообще игнорируется. Примерами могут служить лоты на разработку программных комплексов, в том числе предназначенных для поддержки критически важных объектов, которые публикуются в рамках уже упоминавшихся ранее Национальных программ Минобрнауки РФ, Минсвязи РФ и рядом других государственных ведомств. В них система требований к программным продуктам, как правило, формируется в самом общем виде, без учета специфики продукта, среды окружения, без каких-либо обосновывающих эти требования результатов и, конечно, с жесткими сроками выполнения проекта. Вместе с тем, издержки ошибок на этой стадии разработки программного обеспечения очень велики. Они оборачиваются огромными потерями финансовых и других ресурсов, которые затрачиваются на разработку программного продукта и его сопровождение на других этапах жизненного цикла, чрезвычайными ситуациями и катастрофами вследствие ошибок на этапах его эксплуатации, значительными политическими издержками для страны в целом.

Стадия проектирования в жизненном цикле программного обеспечения безопасности ресурсов сложных систем автоматизации должна начинаться с анализа результатов предпроектных исследований. По его результатам должна уточняться модель потенциальных угроз, дополняться перечнем уязвимостей защищаемого объекта и способов их реализации. На этой основе более строго формулируется политика безопасности и требования безопасности, формируется профиль стандартов жизненного цикла разрабатываемого продукта (профиль безопасности). Полученные таким образом материалы становятся отправными для написания технического задания на разработку программного продукта. В ходе выполнения перечисленных действий вполне вероятно изменение ранее принятых спецификаций и моделей, описывающих семантику процессов, которые призван реализовать создаваемый програм-

мный продукт. Эти модификации должны фиксироваться, находить свое отражение в программных тестовых испытаний и на других этапах жизненного цикла программного продукта. Отметим, что аналогичные действия на разных этапах жизненного цикла продукта следует предпринимать и по отношению к программным средствам, обеспечивающим "Функциональную безопасность" [9]. В качестве примера публикации, где изложены результаты предпроектных исследований, выполненных с участием и под руководством автора и направленных на разработку различных программных средств обеспечения безопасности критически важных объектов от кибертеррористических воздействий, можно привести двухтомное издание [48, 49].

Одним из важнейших требований к программному продукту, который предлагается для массового использования, особенно в вариативной среде окружения, является его отчуждаемость от разработчика. Это требование означает возможность комфортным и менее ресурсозатратным для пользователя продукта образом сопровождать его по назначению и, при необходимости, модифицировать без участия разработчика. Реализации этого требования призваны способствовать рекомендации стандартов программной инженерии на этапе разработки и реализации программного кода. К числу этих рекомендаций относятся: контроль за подготовкой сопроводительной документации; внесение в исходные коды поясняющих комментариев; фиксация любых его модификаций с соответствующими изменениями в спецификациях и верифицирующих моделях, а также другие положения. Их соблюдение гарантирует не только высокую степень отчуждаемости даже сложных программных комплексов с большим объемом кода, отдельные части которого разрабатываются и модифицируются разными специалистами, но и возможности перманентной верифицируемости и способность к реинжинирингу.

Подтверждением представленных выше соображений является одна из российских проблем настоящего времени, которая напрямую влияет на экономику. Она связана со сложностями перевода активно используемых на практике больших программных комплексов на современные высокопроизводительные системы — параллельные вычислительные платформы. В подобных комплексах, предназначенных для решения стратегически важных задач в области атомной энергетики, авиа-космического машиностроения и других, в том числе критически важных секторах национальной экономики, в течение нескольких десятилетий вкладывались знания большого числа ученых и опыт инженеров. Однако, к сожалению, в ходе разработки, сопровождения и объективной модернизации этих программных средств, далеко не всегда соблюдались перечисленные выше стандарты программной инженерии. Как результат, крайне востребованный в настоящее время национальной экономикой их перенос на высокопроизводительные вычислительные установки вызывает очень большие сложности. Озабоченность состоянием программного обеспечения систем автоматизации технологических процессов на этих направлениях неоднократно звучала на самых высоких уровнях государственной власти. В настоящее время она подкреплена финансированием проектов в рамках крупных государственных программ, таких как "Информационное общество 2011—2018 годы" [50]. Однако, как уже отмечалось ранее, быстро изменить ситуацию без решения перечисленного выше комплекса задач не представляется возможным. Такое положение дел должно стать уроком для будущей инновационно-ориентированной экономики.

Заключение

Конечно, представленные в настоящей статье соображения относительно состояния и перспектив развития программной инженерии, ее роли в процессах модернизации национальной экономики не исчерпывают всего многообразия проблемных вопросов, которые стоят на этом направлении и требуют своего разрешения. Здесь освещены только некоторые из вопросов, которые в большей степени близки автору по его основной деятельности, а именно — могут решаться с использованием методологии и аппарата прикладной математики и информатики. Однако неизменным остается постулат о прямой зависимости результатов в модернизации национальной экономики от совершенствования методологии программной инженерии и ее активного применения на практике. Повод для уверенности, что успешное развитие событий на рассматриваемом направлении в России возможно, есть. Об этом свидетельствуют представленные выше примеры и результаты исследований российских ученых.

Хотел бы обратить внимание на то, что многие результаты исследований по обсуждаемым вопросам опубликованы в журнале "Программная инженерия". Данный факт — дополнительное свидетельство того, что этот, совсем еще молодой по традиционным меркам журнал, стал еще одной представительной площадкой для обсуждения вопросов, возникающих в этой очень перспективной и важной области научно-технических знаний, которая в значительной степени определяет темпы модернизации национальной экономики.

Автор выражает благодарность В. В. Липаеву за ценные замечания по работе, которые помогли упорядочить и улучшить ее содержание.

Список литературы

1. **Экономика** в вопросах и ответах: Учеб. пособие / Под. ред. И. П. Николаевой. — М.: ТК Велби. Изд-во Проспект, 2004. 336 с.

2. **Гринин Л. Е., Коротаева А. В., Малков С. Ю.** История и математика: Модели и теории. М.: ЛКИ/ URSS, 2008. 304 с.

3. **Стрельцов А. А.** Цель, структура и методы правового обеспечения информационной безопасности Российской Федерации. // Сборник "Научные и методологические проблемы информационной безопасности" под. ред. В. П. Шерстюка. М.: МГУ, 2004. С. 67—83.

4. **Липаев В. В.** Проблемы программной инженерии: качество, безопасность, риски, экономика // Программная инженерия. 2010. № 1. С. 7—20.

5. **Орлик С. В.** Программная инженерия и жизненный цикл программных проектов с позиций SWEBOK // Программная инженерия. 2011. № 2, № 4.

6. **ГОСТ** Р ИСО/МЭК 15408-2-2002. Часть 2. Функциональные требования безопасности. М.: ИПК Издательство стандартов, 2002.

7. **ГОСТ Р ИСО/МЭК** 15408-3-2002. Часть 3. Требования доверия к безопасности. М.: ИПК Издательство стандартов, 2002.

8. **Галатенко В. А.** Стандарты информационной безопасности / Под. ред. академика РАН В. Б. Бетелина. М.: ИНТУИТ. РУ, 2004. 328 с.

9. **Липаев В. В.** Функциональная безопасность программных средств. М.: СИНТЕГ, 2004.

10. **Васенин В. А.** Internet: от настоящего к будущему // Открытые системы. 2000. № 12. С. 36—45.

11. **Липаев В. В.** Инженерная психология при производстве компонентов программных продуктов // Программная инженерия. 2011. № 1. С. 7—15.

12. **Королев Л. Н.** Структуры ЭВМ и их математическое обеспечение. М.: Наука, 1978. 352 с.

13. **Липаев В. В.** Отечественная программная инженерия: фрагменты истории и проблемы. М.: СИНТЕГ, 2007. 312 с.

14. **Корнеев В. В.** Вычислительные системы. М.: Гелиос АВВ, 2004. 512 с.

15. **Васенин В. А.** Российские академические сети и Internet (состояние, проблемы, решения). М.: РЭФИА, 1997. 173 с.

16. **Садовничий В. А., Васенин В. А.** и др. Российский Интернет в цифрах и фактах. М.: Изд-во Московского университета, 1999. 32 с.

17. **Васенин В. А.** Высокопроизводительные научно-образовательные сети России. Настоящее и будущее. М.: Изд-во Московского университета, 1999. 32 с.

18. **Костюхин К. А.** Модель разработки программного обеспечения с открытым исходным кодом // Программная инженерия. 2010. № 1. С. 31—40.

19. **Применение** измерительно-вычислительных комплексов в аэродинамическом эксперименте (по материалам иностранной печати) // Обзоры. Издательский отдел ЦАГИ. 1982. № 615. 186 с.

20. **Метрологическое** обеспечение экспериментальных исследований в аэродинамических трубах (по материалам иностранной печати) // Обзоры. Издательский отдел ЦАГИ. 1989. № 698. 50 с.

21. **Экономическая** эффективность использования вычислительной техники в экспериментальных исследованиях // Обзоры, рефераты. Издательский отдел ЦАГИ, 1981. № 603.

22. **Васенин В. А.** Критическая энергетическая инфраструктура: средства противодействия кибертерроризму // Информационные технологии. 2009. № 10. С. 2—9.

23. **Gerwin Klein** et al. seL4: Formal Verification of an OS Kernel // 22nd ACM Symposium on Principles of Operating Systems. October 2009. Big Sky. MT. USA.

24. **Непомнящий В. А., Андреев И. С., Михайлов И. Н., Промский А. В.** На пути к верификации С-программ. Язык C-Light и его формальная семантика // Программирование. 2002. № 6. С. 65—80.

25. **Васенин В. А., Водомеров А. Н., Конев И. М., Степанов Е. А.** Т-подход к автоматизированному распараллеливанию программ: идеи, решения, перспективы / Под. ред. академика РАН В. А. Садовничего. М.: МЦНМО, 2008. 460 с.

26. **Berners-Lee T. and Fischetti M.** Weaving the Web: the original design and ultimate destiny of the World Wide Web by its inventor. Harper San Francisco, 1999.

27. **Berners-Lee T., Hendler J., and Lassila O.** The Semantic Web // Scientific American. 2001. Vol. 284. No. 5. P. 34—43.

28. **Васенин В. А., Афонин С. А., Голомазов Д. Д.** Использование семантических технологий для обнаружения грид-ресурсов // Программная инженерия. 2011. № 7. С. 2—8.

29. **Загорулько Ю. А., Загорулько Г. Б.** Онтологии и их практическое применение в системах, основанных на знаниях // Материалы Всероссийской конференции с международным участием "Знания — Онтологии — Теории", 3—5 октября 2011, Новосибирск. Новосибирск: РИЦ "Прас — Курьер", 2011.

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

31. **Пирогов М. П.** Интеллектуальный стандарт обеспечения информационно-системной безопасности сложных систем // Нейрокомпьютеры: разработка, применение. 2008. № 7. С. 18—25.

32. **Бурдонов И. Б., Косачев А. С., Кулямин В. В.** Теория соответствия для систем с блокировками и разрушением. М.: Наука, 2008.

33. **Бурдонов И. Б.** Теория конформности для функционального тестирования программных систем на основе формальных моделей: дис. ... д-ра физ.-мат. наук. ИСП РАН, 2008.

34. **Галатенко В. А.** Исследование и реализация методов повышения производительности труда профессиональных программистов: дис. ... д-ра физ.-мат наук. НИИСИ РАН, 2001.

35. **Васенин В. А., Гирча А. И.** Программный комплекс для проведения наукоемкого вычислительного эксперимента на основе вихревых методов численного моделирования процессов нестационарной гидродинамики // Программная инженерия. 2010. № 2. С. 25—32.

36. **Силаков Д. В.** Linux: интерфейсные стандарты и профили // Открытые системы. 2010. № 1. С. 44—47.

37. **Силаков Д. В.** Информационно-аналитическая система для разработки и использования базового стандарта операционной системы Linux (LSB) // Информационные технологии. 2010. № 5. С. 53—58.

38. **Pauntasso C., Zimmermann O., Leymann F.** REST ful Web Services vs. Big Web Services:Making the Right Architectural Deasion // Proc. of the 17th International Word Wide Web Conference 2008. P. 805—814.

39. **Foster I., Kesselman C., Tuecke S.** The Anatomy of the Grid: Enabling Scalable Virtual Organizations // International J. Supercomputer Applications 2001. V. 15. N. 3. P. 200—222.

40. **Foster I., Kesselman C., Nick J., Tuecke S.** The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration // Global Grid Forum 2002. URL: http://www.globus.org/alliance/publications/papers/ogsa.pdf (дата обращения: 1.09.2011).

41. **Sheyner O.** Scenario graphs and attack graphs: Ph. D. dissertation. Carnegie Mellon University. P. Hsburg, PA, USA. April 2004.

42. **Галатенко В. А.** К проблеме автоматизации анализа и обработки информационного наполнения безопасности // Программная инженерия. 2011. № 3. С. 45—48.

43. **Котенко И. В., Степашкин М. В., Богданов В. С.** Интеллектуальная система анализа защищенности компьютерных сетей. URL: http: //www.positif.org/docs/SPIIRAS-NCAI'06-Stepashkin.pdf.

44. **Климовский А. А.** Таксономия кибератак и ее применение к задаче формирования сценариев их проведения // Труды Института системного анализа Российской академии наук. Том 27: Проблемы кибербезопасности информационного общества. М.: ИСА РАН, 2006.

45. **Пальчунов Д. Е., Яхьяева Г. Э., Хамутская А. А.** Программная система управления информационными рисками RiskPanel // Программная инженерия. 2011. № 7. С. 29—36.

46. **Девянин П. Н.** Обзор семейства ДП-моделей безопасности логического управления доступом и информационными потоками в компьютерных системах // Информационные технологии. 2010. № 5. С. 20—25.

47. **Шапченко К. А.** Способ проверки свойств безопасности в моделях логического разграничения доступа с древовидной иерархией объектов доступа // Информационные технологии. 2009. № 10. С. 13—17.

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

49. **Критически** важные объекты и кибертерроризм. Часть 2. Аспекты программной реализации средств противодействия. / Андреев О. О. и др. под ред. В. А. Васенина. М.: МЦНМ, 2008. 607 с.

50. http://fcp.economy.gov.ru/cgi-bin/cis/fcp.cgi/Fcp/Passport/View/2012/369/

УДК 004.436.2

А. В. Жарковский, канд. техн. наук,
А. А. Лямкин, канд. техн. наук,
Т. Ф. Тревгода, канд. техн. наук, e-mail: tat.trevgoda@yandex.ru,
Санкт-Петербургский государственный электротехнический университет "ЛЭТИ"

Объектно-признаковый язык описания сложных технических систем

Предлагается объектно-признаковый язык описания структуры сложных технических систем, их компонентов, топологии, информационных связей и процессов функционирования.

Ключевые слова: *сложная техническая система, объектно-признаковый язык, описание, структура, объект, формуляр, модель*

Цели функционирования любых сложных технических систем (СТС), примерами которых являются самолеты, корабли, зенитные комплексы или различные их объединения, достигаются только благодаря управлению. В основе алгоритмов управления лежат правила поведения системы и ее компонентов, приводящие к достижению заданной цели. Алгоритмы управления реализуются в виде функционального программного обеспечения (ФПО) комплексов управления. В конечном счете, ФПО определяет сроки создания и эффективность функционирования СТС.

Правила поведения системы (правила принятия решений, правила управления, правила боевого использования) формулируются заказчиком системы на естественном языке в терминах конкретной предметной области, а их реализация в виде ФПО осуществляется исполнителем. Вербальность технического задания (ТЗ) ведет к разному пониманию одних и тех же задач заказчиком и исполнителем, что, как всем известно, приводит к длительному и дорогостоящему процессу отладки системы, а нередко и к провалу всего проекта.

Синтаксис и семантика

Ниже предлагается объектно-признаковый язык (ОПЯ) описания предметной области. Это некий метаязык [1], который является промежуточным между вербальным языком и языками программирования высокого уровня. Он может служить для формализации описания СТС и процесса ее функционирования, а также для формулирования ТЗ на ФПО с целью однозначного его понимания заказчиком и исполнителем.

Для понимания сути языка достаточно рассмотреть лишь его синтаксис и семантику. Можно считать, что каждый конкретный объект X принадлежит к определенному множеству (типу) объектов ($T.X$). Любой объект этого множества характеризуется номером (именем конкретного экземпляра) и целым набором признаков, которые перечисляются через запятую в квадратных скобках после имени объекта ($T.X$ [...]). Объекты и их признаки обозначаются на латинице начальными буквами терминов языка предметной области и в виде общепринятых физических величин.

Существуют признаки числовые (например, D — дистанция, V — скорость, U — угол) и нечисловые (например, TC — тип цели, PH — признак высоты и др.). Если признак объекта характеризуется только одним членом некоторого множества, то все члены множества перечисляются в фигурных скобках (например, TC:{C, W, R}, где C — самолет, W — вертолет, R — ракета). Если признак объекта характеризуют все члены множества, то они задаются кортежем в ломаных скобках (например, GR: ⟨LX, LY, LZ⟩, где GR — геометрические размеры объекта вдоль осей его симметрии).

Признаки бывают зависимыми и независимыми. Зависимость (функция) одного признака от другого указывается в круглых скобках (например, $E(D)$ — энергия или мощность E принимаемого излучения зависит от дальности D до источника излучения). Описание типа любого объекта X можно рассматривать как шапку таблицы с именем $T.X$, где в первом столбце указывается такой признак объекта как его имя, а проще номер ($N\_T.X$), а в последующих столбцах — все остальные признаки.

Описание структуры СТС

Любая СТС в обобщенном виде может рассматриваться как совокупность (рис. 1) таких взаимосвязанных средств, как *SZ* — средства технического зрения, с помощью которых СТС получает информацию о состоянии окружения (о других системах, с которыми она взаимодействует); *SV* — средства информационного и/или материального воздействия на противостоящие СТС; *KU* — средство (комплекс) управления; *SO* — оператор или командир системы и *SD* — средство доставки (носитель) всех других средств. Эта пятерка представляет собой ядро, которое может быть использовано как для построения блочной модели структуры различных СТС, так и для формализованного ее описания. При этом отдельные компоненты ядра (кроме *KU*) на разных уровнях иерархии системы могут отсутствовать.

В качестве примера на рис. 2 представлена блочная модель структуры двух взаимодействующих сложных военно-технических систем — воздушной ударной группы (система В) и наземной батареи самоходных зенитных комплексов (система Н). Системы получают информацию друг о друге и воздействуют друг на друга через среду, обозначенную как *Q*.

Каждому уровню иерархии системы (каждому комплексу) можно присвоить буквенное обозначение: *L* — нижний (локальный) уровень, *G* — уровень группы, *S* — уровень соединения и т. д. Тогда формализованное описание двухуровневой системы, например, батареи зенитных комплексов (ЗК) или самолетной ударной группы, можно представить в виде двух строк:

T.KG [*N_T.KG, N_T.SZ, N_T.KU, N_T.SO, N_T.SD, QSV, N_T.SV, N_T.RKG, QKL, N_T.KL,*];

T.KL [*N_T.KL, N_T.SZ, N_T.KU, N_T.SO, N_T.SD, QSV, N_T.SV, N_T.RKL*],

где первое выражение содержит описание типа группы (*T.KG*), а второе — типа входящих в нее зенитных или авиационных комплексов (*T.KL*). Оба выражения, кроме имен (*N_T.KG* и *N_T.KL*) уровней управления (комплексов), содержат имена экземпляров средств технического зрения (*N_T.SZ*), комплекса управления (*N_T.KU*), оператора (*N_T.SO*), носителя (*N_T.SD*), число (*QSV*) и имена (*N_T.SV*) подчиненных средств воздействия, например, ракетных установок, а также имена вида пространственного размещения комплексов (*N_T.RKG* и *N_T.RKL*). Кроме того, в типе *T.KG* указывается число подчиненных локальных комплексов (*QKL*) и их имена (*N_T.KL*). Если комплексы какого-либо уровня являются стационарными (например, такими могут быть командный пункт батареи или зенитный комплекс), то в их описании будут отсутствовать имена носителя *N_T.SD*, а при автоматическом управлении — *N_T.SO* и т. д.

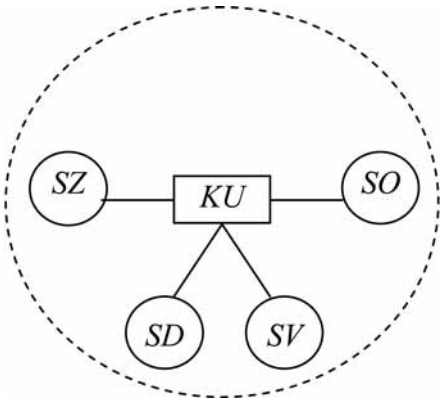


Рис. 1. Обобщенная структура сложной технической системы

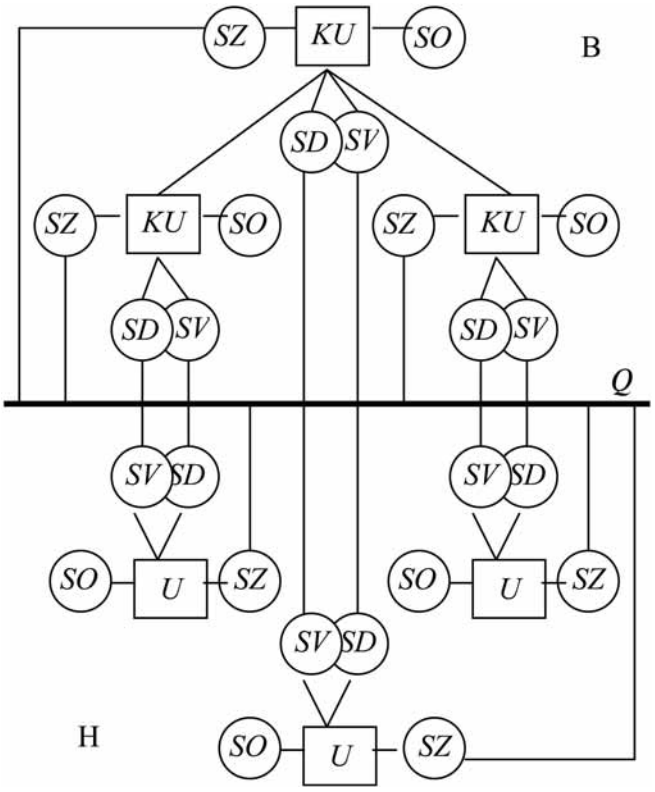


Рис. 2. Модель структуры двух взаимодействующих сложных военно-технических систем

Описание пространственного размещения

Тип пространственного размещения *T.RX* любого объекта (комплекса) относительно заданной (реперной) точки в общем случае характеризуется его координатами (*X,Y,Z*), скоростью (*V*) и направлением движения объекта в горизонтальной (*EZ*) и вертикальной (*EY*) плоскостях неподвижной декартовой системы координат:

T.RX [*N_T.RX, X, Y, Z, V, EZ, EY*].

При неподвижном объекте углы *EZ* и *EY* определяют его пространственную ориентацию. Тип размещения отдельного средства характеризуется координатами относительно геометрического центра носителя. Тип размещения является важной характеристикой начальных условий функционирования системы.

Описание отдельных средств системы

При описании конкретных образцов средств системы (обнаружителей, носителей и др.) следует исходить из обобщенного описания каждого средства, что ведет к единообразию описаний и позволяет выявить или наоборот не упустить существенные детали. При этом важно отдавать себе отчет, для каких целей будет использоваться данное описание. Каждый объект в целом обладает и функциональными, и конструктивными, и экономическими характеристиками. Однако используются они порознь при решении разных по целям задач. Ограничимся двумя примерами.

Так, несмотря на большое разнообразие обнаружителей, их типовое описание как объектов функционирования можно представить в виде:

$$T.SZ\ [N\_T.SZ,\ PD{:}\{A,\ P\},\ TP{:}\{RS,\ RL,\ IK,\ LZ,\ TV\},\ QM,\ FM,\ FH,\ DM,\ DH,$$

$$SPZ,\ SPY,\ UZZ,\ UZY,\ IK{:}\{K,\ M,\ D,\ V\},\ TD,\ TE],$$

где *N_T.SZ* — имя обнаружителя; *PD* — принцип действия (*A* — активный, *P* — пассивный); *TP* — тип рабочего поля, точнее, частотный диапазон единого электромагнитного поля (*RS* — радиосвязной, *RL* — радиолокационный, *IK* — инфракрасный, *LZ* — лазерный, *TV* — телевизионно-оптический); *QM* — максимальное число сопровождаемых целей; *FM* и *FH* — низшая и высшая частоты приема сигнала или обратные им длины волн, определяющие частотный или волновой диапазон приемника; *DM* и *DH* — минимальная и максимальная дистанции обнаружения цели; *SPZ* и *SPY* — секторы поиска цели в горизонтальной и вертикальной плоскостях; *UZZ* и *UZY* — углы зрения (ширина диаграммы направленности) приемника в горизонтальной и вертикальной плоскостях; *IK*:{*K*, *M*, *D*, *V*} — измеряемые координаты (*K* — курсовой угол, *M* — угол места, *D* — дальность, *V* — скорость); *TD* и *TE* — соответственно точности измерения дистанций и угловых величин.

Описание конкретного типа обнаружителя значительно упрощается, так как многие признаки могут быть не указаны и подразумеваться по умолчанию. Например, описание обзорной РЛС с антенной решеткой за счет опускания принципа действия, типа рабочего поля, углов зрения и секторов поиска может выглядеть следующим образом:

$$T.SZ\ [N\_T.SZ,\ QM,\ DM,\ DH,\ IK{:}\{K,\ M,\ D\},\ TD,\ TE].$$

В описании типа носителя (средства доставки) с функциональной точки зрения нет места массогабаритным, стоимостным или эксплуатационным характеристикам. Следовательно, тип носителя может быть представлен как

$$T.SD\ [N\_T.SD,\ HM,\ HH,\ VM,\ VH,\ UZ,\ UY],$$

где *HM* и *HH* — соответственно минимальная и максимальная высоты применения носителя; *VM* и *VH* — крейсерская и максимальная линейные скорости движения носителя; *UZ* и *UY* — максимальные угловые скорости движения в горизонтальной и вертикальной плоскостях. Эти величины выступают в качестве ограничений в модели функционирования носителя.

Аналогичным образом описываются и другие типы объектов, в том числе и алгоритмы управления.

Описание информационных связей

Все информационные связи реализуются в виде кодограмм или формуляров обмена (*F.*), с расширением по имени средства, из которого они выходят или в которое входят. Чтобы различать входные и выходные формуляры одного и того же объекта, для входного формуляра вводится дополнительное расширение *U* (управляющий), а для выходного — *I* (информирующий). Когда есть однозначное понимание, о каком формуляре идет речь (например, при последовательном соединении объектов) нужда в дополнительном расширении отпадает. Различие между типами объектов *T.X* [...] и типами формуляров *F.X* [...] заключается только в том, что признаки в формулярах являются переменными величинами, а в типах объектов — постоянными. Совокупность формуляров обмена образует информационную модель объекта (системы, комплекса управления).

На рис. 3 показана информационная модель комплекса управления (*KU*) СТС нижнего уровня иерар-

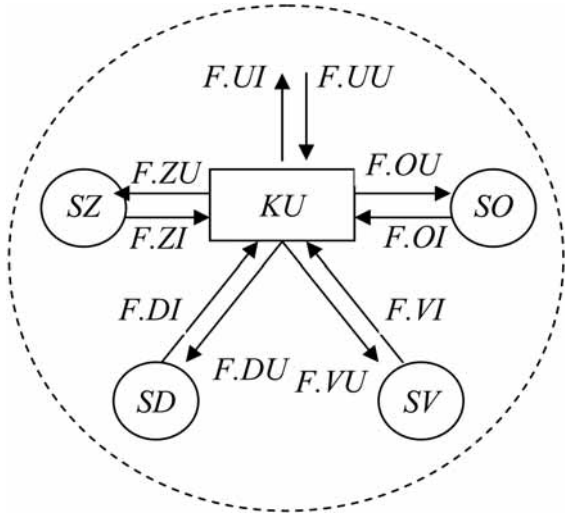


Рис. 3. Информационная модель комплекса управления

хии (информационные связи *KU* со средствами системы и надсистемой). Связи представлены в виде формуляров обмена информацией. Например, информирующий формуляр ранее рассмотренного радиолокационного обнаружителя имеет вид:

$$F.ZI\ [LC,\ N\_C,\ K,\ M,\ D],$$

где LC — число сопровождаемых целей; N_C — номер цели в нумерации данного обнаружителя; K, M, D — измеряемые координаты целей (соответственно курсовой угол, угол места и наклонная дальность до цели).

Управляющий формуляр носителя имеет вид

$$F.DU\ [V,\ EZ,\ EY],$$

где V — задаваемая скорость движения носителя, EZ и EY — задаваемые направления движения соответственно в горизонтальной и вертикальной плоскостях. Аналогичным образом могут быть описаны и все другие связи.

Описание процессов функционирования

Информационная модель объекта может служить для системного описания процесса его функционирования. Модель (алгоритм) функционирования M какого-либо объекта X как преобразование информации из входного формуляра $F.XU$ в выходной $F.XI$ в соответствие с характеристиками объекта, задаваемыми его типом $T.X$, может быть представлена в виде выражения

$$M.X = F.XU \xrightarrow{T.X} F.XI.$$

В общем случае число входных и выходных формуляров не ограничено. Например, модель комплекса управления СТС (см. рис. 3) можно записать как

$$M.KU = F.ZI\ \&\ F.OI\ \&\ F.DI\ \&\ F.VI\ \&\ F.UU \xrightarrow{T.KU}$$
$$\rightarrow F.ZU\ \&\ F.OU\ \&\ F.DU\ \&\ F.VU\ \&\ F.UI.$$

Внутреннее содержание модели (алгоритма) определяется правилами преобразовния информации, которые в общем случае представляют собой совокупность (систему) отдельных правил и вычислительных процедур по расчету входящих в правила переменных величин. Сами отдельные правила представляют собой продукции "если ..., то ...", левая часть которых содержит совокупность элементарных условий, а правая — совокупность элементарных действий. Условия записываются в виде отношений (=, >, ≈ и т. п.) между переменными и постоянными (параметрами) величинами, а действия — в виде операции присвоения (:=) переменным вычисляемых значений или операции следования (перехода к другому правилу, другому алгоритму). Условия и действия соединяются логическими функциями "И" ($\wedge$) и "ИЛИ" ($\vee$). Для записи вычислительных процедур используются стандартные математические функции.

Например, одно из правил боевого использования корабельных средств оптико-электронного подавления в вербальной форме записывается в виде: "Если цель, находящаяся на дистанции более 50 км, является ударным самолетом CU, движущимся на корабль с правого борта, то ставить против нее помеховым комплексом PK № 2 фигуру ложных целей (режим работы RR) № 3". Для распознавания типа целей существуют свои правила (рассматривать которые мы не будем), а чтобы определить, движется ли цель на корабль, необходимо вычислить ее курсовой параметр KP, который является функцией дальности D, угла места цели M, ее магнитного пеленга MP и направления движения EZ:

$$KP = D\,\mathrm{COS}(M)\,\mathrm{SIN}(EZ - (180 - MP)),$$

где SIN и COS— функции синуса и косинуса.

Правило постановки помех может быть записано в виде

$$(TC = SU)\wedge(D \geqslant 50\ 000)\wedge(K \leqslant 180)\wedge(KP \leqslant LX) \Rightarrow$$
$$\Rightarrow (N\_PK := 2)\wedge(RR := 3),$$

где условие $TC = SU$ означает, что тип цели — ударный самолет, $K \leqslant 180$ — означает, что цель находится по правому борту корабля на дистанции более 50 км ($D \geqslant 50\ 000$), а при выполнении условия $KP \leqslant LX$ — цель движется на корабль (LX — его полудлина). В правой части этого выражения указывается номер помехового комплекса и режим его работы RR.

Для представления алгоритмов и моделей с использованием ОПЯ может служить улучшенная авторами форма структурограмм (диаграмм Нэсси—Шнайдермана), в которых для обозначения всех возможных действий используются всего четыре графических символа в виде стрелок (примеры использования структурограмм приведены в работе [1]).

Заключение

Объектно-признаковый язык позволяет в формализованном виде описать структуру СТС, пространственное размещение системы, характеристики составляющих ее компонентов, внутренние и внешние информационные связи, а также алгоритмы управления и модели функционирования. Это дает возможность представлять технические задания на проектирование в более строгой форме и обеспечивает лучшее взаимопонимание между специалистами заказчика и исполнителя, что ведет к сокращению сроков создания и повышению качества ФПО комплексов управления сложными техническими системами.

Литература

1. **Лямкин А. А.** Концептуальное проектирование средств управления подвижных объектов. СПб.: Изд-во СПбГЭТУ "ЛЭТИ", 2006.

УДК 004.724.4 + 519.872.3

О. И. Заяц, канд. физ.-мат. наук, доц., e-mail: zayats@amd.stu.neva.ru,
В. С. Заборовский, д-р техн. наук, проф., зав. каф., e-mail: vlad@neva.ru,
В. А. Мулюха, канд. техн. наук, асс., e-mail: vladimir@mail.neva.ru,
А. С. Вербенко, аспирант,
Санкт-Петербургский государственный политехнический университет

Управление пакетными коммутациями в телематических устройствах с ограниченным буфером при использовании абсолютного приоритета и вероятностного выталкивающего механизма. Часть 1

Рассматривается математическая модель исследуемой телематической системы, представляющая собой одноканальную двухпотоковую приоритетную систему массового обслуживания ограниченной емкости. Здесь пакетный трафик, сформированный из команд управления, является приоритетным и имеет два вида преимуществ перед остальным трафиком: абсолютный приоритет в обслуживании и вероятностный выталкивающий механизм буферной памяти. Примером применения такого подхода служит проектирование алгоритмов и реализация программных средств удаленного управления робототехническими устройствами в условиях космического эксперимента на МКС.

Ключевые слова: *приоритетная система массового обслуживания, абсолютный приоритет, вероятностный выталкивающий механизм, пакетный трафик, управление телематическими системами*

Введение

Реальные потоки данных в современных компьютерных сетях весьма сложны по своей структуре. Суммарный поток на входе телематического устройства, как правило, формируется несколькими независимыми источниками. Заявки, составляющие этот поток, могут различаться по многим признакам: по скорости передачи через каналы связи; пропускной способности используемых виртуальных соединений; алгоритмам обработки в узлах коммутации; доступному объему буферной памяти; допустимому уровню потерь; способу защиты информации; экономическим показателям трафика и др.

Применяемые в инженерной практике сетевые технологии чрезвычайно разнообразны. Они опережают в своем развитии и практическом внедрении разработку адекватных аналитических моделей и их теоретическое обоснование. Стохастический характер процессов функционирования компьютерных сетей и структура формируемых при этом виртуальных соединений предопределяет возможность использования для их анализа методов и моделей теории массового обслуживания (ТМО). Однако анализ сетевых процессов на основе однопотоковых и однофазных моделей систем массового обслуживания (СМО), характерный для раннего этапа теоретического анализа сетевых взаимодействий, в настоящее время уже не отражает должным образом особенностей современ-

ных компьютерных сетей. Упомянутые выше модели позволяют получить лишь самые общие качественные характеристики процессов сетевого взаимодействия, но не могут использоваться для оценки пропускной способности и задержек в каналах связи при передаче мультимедиа трафика, образованного данными от различных источников информации. В этих условиях многопотоковые модели СМО позволяют более точно описывать сложные процессы передачи трафика в современных компьютерных сетях. При этом повышается не только детальность, точность и надежность результатов, но и открываются возможности решения принципиально новых задач, важнейшими из которых являются задачи управления.

В настоящей статье изучается простейшая многопотоковая модель, учитывающая два потока данных. В этой модели управление пакетными коммутациями осуществляется с помощью следующих механизмов: во-первых, абсолютного приоритета в обслуживании пакетного трафика, сформированного из команд управления, и, во-вторых, вероятностного выталкивающего механизма, которым снабжена буферная память телематических устройств управления. В результате пакеты приоритетного типа с некоторой вероятностью $0 \leqslant \alpha \leqslant 1$ способны выталкивать из буфера телематического устройства управления неприоритетные пакеты. Выбором значения α осуществляется адаптивная настройка алгоритма управления трафиком. Это обстоятельство позволяет эффективно перераспределять доступную для различных виртуальных соединений пропускную способность сети, управляя тем самым взаимной пропускной способностью приоритетных и неприоритетных потоков данных.

Подобного рода задачи чрезвычайно актуальны при управлении удаленными робототехническими системами в тех случаях, когда используются не выделенные каналы связи, а сети общего доступа. Типичным примером такой задачи может служить космический эксперимент "Контур", более подробно описываемый далее.

Заметим, что в классической теории СМО хорошо изучены лишь системы с детерминированным выталкиванием ($\alpha = 1$) и без выталкивания ($\alpha = 0$). Произвольные значения α изучались только в системах с относительным приоритетом, который менее употребителен для рассматриваемого класса задач управления. Полученные результаты показывают, что использование параметра α в алгоритме управления является чрезвычайно простым и эффективным способом оптимизации сетевых взаимодействий. Модели, основанные на использовании СМО с абсолютным приоритетом и вероятностным выталкивающим механизмом, являются достаточно точными и позволяют получить на их основе все необходимые для оценки эффективности управления характеристики каналов связи, а затем в процессе натурального эксперимента идентифицировать параметры используемых моделей.

Функциональная схема организации информационного взаимодействия

Наглядным примером широких возможностей использования вероятностного выталкивающего механизма является задача управления удаленным робототехническим объектом, результаты функционирования которого наряду с данными телеметрии и видеопотоком передаются с помощью широкополосных сетей общего доступа. В этом случае команды управления передаются с использованием транспортного протокола TCP, а результаты наблюдения представляют собой поток видеоданных (рис. 1). В данном космическом эксперименте пропускная способность сети составила для приоритетных TCP-пакетов управления и телеметрии 100 Кбит/с, а для фонового видеопотока UDP-пакетов — 1,2 Мбит/с. При этом отношение коэффициентов загрузки [1, 5] для двух указанных потоков составило 1 к 12.

В рассматриваемом примере использование приоритетного механизма управления в комбинации с выталкивающим механизмом позволяют сбалансировать такие показатели функционирования сети как вероятность потери команд управления и качество получаемой картины видеонаблюдения для различных состояний сетевой среды. Упомянутый ранее параметр α может варьироваться также и в целях изменения задержек пакетов в контуре обратной связи системы управления роботом.

Данная задача важна для дальнейшего развития интерактивного управления удаленными динамическими объектами в случае, когда сетевая среда является составной частью контура обратной связи. Уровень потерь и задержек в рассматриваемом случае является важным параметром, характеризующим эффективность системы управления в целом.

Математическая постановка задачи

Рассмотрим одноканальную СМО ограниченной емкости k ($1 < k < \infty$), на вход которой поступают два типа независимых простейших потоков заявок (пакетов), первый интенсивностью λ_1 и второй — λ_2. Допустим, что длительность обслуживания (обработки любого пакета) распределена по одному и тому же показательному закону. Все эти длительности независимы в совокупности. Интенсивность обслуживания обозначается μ.

Пакетам первого типа предоставлен абсолютный приоритет в обслуживании. Пока все приоритетные запросы на обслуживание не будут обработаны, заявки второго типа остаются в очереди. Вновь поступившие приоритетные запросы прерывают обработку неприоритетных и вытесняют их в накопитель (если есть свободные места ожидания), либо за пределы системы (если свободных мест нет). Снятые с обслуживания пакеты присоединяются к очереди неприоритетных требований.

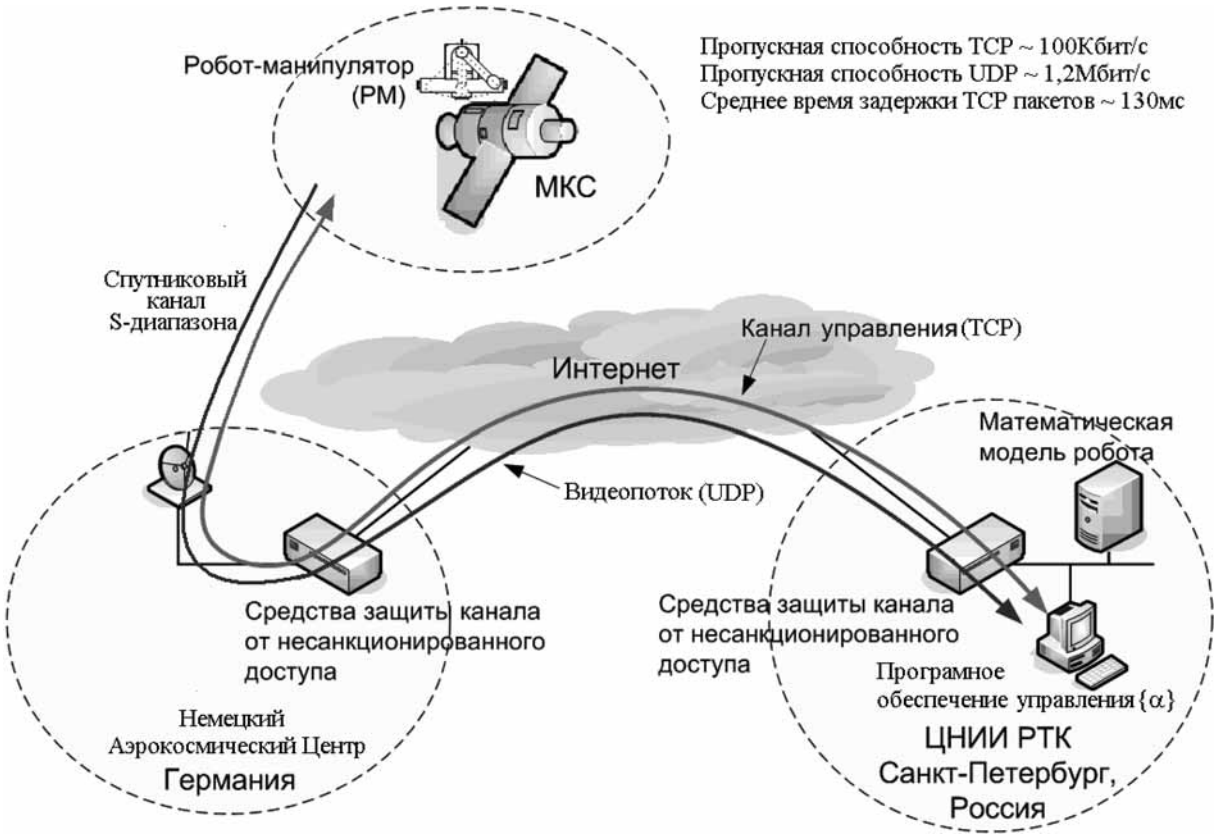


Рис. 1. Схема космического эксперимента "Контур"

Накопитель является общим, свободные места ожидания полнодоступны для любого вновь поступившего запроса. Динамически формируются две отдельные очереди пакетов, их суммарная длина ограничена емкостью накопителя ($k - 1$). В отличие от типовых приоритетных СМО [1, 2] рассматриваемая система снабжена вероятностным выталкивающим механизмом. Приоритетный пакет, заставший все места ожидания занятыми в момент обработки другого приоритетного пакета, с заданной вероятностью α вытесняет из накопителя один из менее приоритетных пакетов и занимает его место. Вытесненный пакет теряется.

Схема описанной СМО приведена на рис. 2. Через $a_i(\tau)$ и $b_i(x)$ обозначены законы распределения, соответственно, интервала между требованиями и времени обслуживания для i-го потока пакетов. В отсутствие выталкивающего механизма ($\alpha = 0$) и при детерминированном выталкивании ($\alpha = 1$) аналогичные системы изучал Г. П. Башарин [3]. Сама концепция вероятностного выталкивающего механизма применительно к сетевым и телекоммуникационным проблемам предложена в работе [4]. Следует, однако, отметить, что в этой работе механизм выталкивания сочетался с относительным, а не с абсолютным, как в рассматриваемом случае, приоритетом.

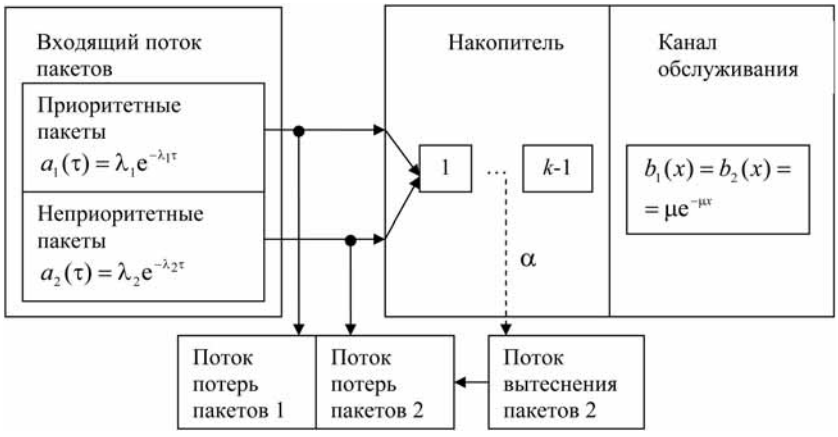


Рис. 2. Схема приоритетной СМО класса $\vec{M}_2/M/1/k/f_2^1$

Суммарный входящий поток СМО, изображенный на рис. 2, будет простейшим с интенсивностью $\lambda = \lambda_1 + \lambda_2$. Если отвлечься от различий между типами требований и отслеживать только общее число пакетов в системе, то такая упрощенная однопотоковая модель имела бы сокращенное обозначение по Д. Кендаллу $M/M/1/k$ [5]. Первый символ в обозначе-

нии показывает распределение интервалов поступления заявок в систему, второй — распределение длительности обслуживания, третий — число каналов обслуживания, а четвертый — емкость накопителя. Специальная модификация системы обозначений Кендалла, предназначенная для приоритетных СМО, принадлежит Г. П. Башарину [3]. В модифицированной системе общая структура обозначения и смысл отдельных его позиций сохраняется, однако в каждой позиции используется векторная символика. Кроме того, предусмотрен дополнительный символ f_i^j, где i указывает на тип приоритета (0 — без приоритета, 1 — относительный, 2 — абсолютный), а j — на тип выталкивающего механизма (0 — без выталкивания, 2 — детерминированное выталкивание).

В оригинальной системе [3] значение $j = 1$ не было задействовано. На наш взгляд, его целесообразно использовать для обозначения вероятностного выталкивающего механизма, как промежуточного между вариантами $j = 0$ и $j = 2$. С этим дополнением по правилам из работы [3] СМО, изображенная на рис. 2, должна быть отнесена к классу $\vec{M}_2/M/1/k/f_2^1$.

История исследования одноканальных двухпотоковых приоритетных систем насчитывает уже более полувека, однако, насколько известно авторам, лишь в работе [4] изучался вероятностный выталкивающий механизм (в сочетании с относительным приоритетом для СМО класса $\vec{M}_2/M/1/k/f_1^1$). Вместе с тем, для типовых моделей выталкивающего механизма ($j = 0$ и $j = 1$) задача в основном решена.

Значительный вклад в развитие теории приоритетных систем массового обслуживания внес Г. П. Башарин и его ученики. Типовые модели $\vec{M}_2/\vec{M}_2/1/k/f_2^0$, а также $\vec{M}_2/\vec{M}_2/1/k/f_2^2$ представлены в работе [6]. Вторая часть указанной работы посвящена анализу систем $\vec{M}_2/\vec{M}_2/1/\vec{k}/f_2^0$ и $\vec{M}_2/\vec{M}_2/1/\vec{k}/f_2^2$ с раздельными очередями. Такая же структура системы, но с конечным числом источников нагрузки l, обозначаемая как $\vec{M}_2/M/1/k/l/f_2^2$, представлена в работе [7]. Там же рассмотрена еще одна более сложная модель $\vec{M}_2/M/1/\vec{k}/\vec{l}/f_2^2$ с раздельными очередями и разделением источников нагрузки по типам требований.

Задачи по исследованию приоритетных СМО, подобных рассмотренным в работах [3, 6, 7], первоначально возникли в телекоммуникационных и информационных системах, связанных в частности с анализом реальных дисциплин диспетчеризации в управляющих ЭВМ. В последние годы подобного рода модели СМО, а также различные их обобщения широко используются при теоретическом анализе реальных интернет-систем.

Интересная задача решена А. Бонди [8]. В этой работе типовые модели с общей очередью $\vec{M}_2/M/1/k/f_2^0$ и раздельными очередями $\vec{M}_2/M/1/\vec{k}/f_2^0$ сравниваются с частично разделенными очередями при сохранении некоторой общей части накопителя. Решена задача оптимального распределения емкости накопителя между его полнодоступной и неполнодоступной частями (по критерию максимума средней суммарной очереди). В работе [9] рассмотрен пороговый выталкивающий механизм, при котором выталкивание разрешено лишь при достаточно большой длине очереди неприоритетных пакетов.

Как показано в работе [4], вероятностный выталкивающий механизм является более удобным и эффективным по сравнению с другими известными по публикациям математическими моделями выталкивания. Он адекватно описывает реальные процессы сетевого трафика и при этом достаточно прост в плане математического представления. Все это делает данное направление исследований актуальным и практически значимым.

Вычисление производящей функции

Будем характеризовать состояние системы двумерным процессом $\vec{N}(t) = \{N_1(t), N_2(t)\}$, где $N_i(t)$ — число требований i-го типа в системе в момент t. При сделанных допущениях, процесс $\vec{N}(t)$ является марковским и будет развиваться в фазовом пространстве

$$\Omega = \{(n_1, n_2): n_1 \geqslant 0,\ n_2 \geqslant 0,\ n_1 + n_2 \leqslant k\},$$

где n_i — число требований i-го типа в системе в установившемся режиме.

Размеченный граф состояний процесса $\vec{N}(t)$ представлен на рис. 3.

По теореме Маркова процесс $\vec{N}$ эргодичен. Далее ограничимся анализом установившегося режима. Финальные вероятности состояний обозначим через $p_{i,j}$ ($i = \overline{0, k}$; $j = \overline{0, k-i}$).

Составляя с помощью рис. 3 по обычным правилам [5] систему уравнений Колмогорова, получим

$$\begin{gathered}-[\lambda_1(1-\delta_{j,\,k-i}) + \alpha\lambda_1(1-\delta_{j,\,k})\delta_{j,\,k-i} + \\ + \lambda_2(1-\delta_{j,\,k-i}) + \mu(1-\delta_{i,\,0}\delta_{j,\,0})]p_{i,j} + \\ + \mu p_{i+1,j} + \mu\delta_{i,\,0}p_{i,j+1} + \lambda_2 p_{i,j-1} + \\ + \lambda_1 p_{i-1,j} + \alpha\lambda_1\delta_{j,\,k-i}p_{i-1,j+1} = 0, \\ (i = \overline{0, k};\ j = \overline{0, k-i}),\end{gathered} \qquad (1)$$

где $\delta_{i,j}$ — дельта-символ Кронекера. Система (1) должна решаться совместно с условием нормировки

$$\sum_{i=0}^{k}\sum_{j=0}^{k-1} p_{i,j} = 1. \qquad (2)$$

При реальных (достаточно больших) k система (1)—(2) плохо обусловлена и ее непосредственное численное решение приводит к большим вычислительным погрешностям. Многие авторы, в частности, Г. П. Башарин обходят эту трудность, применяя метод

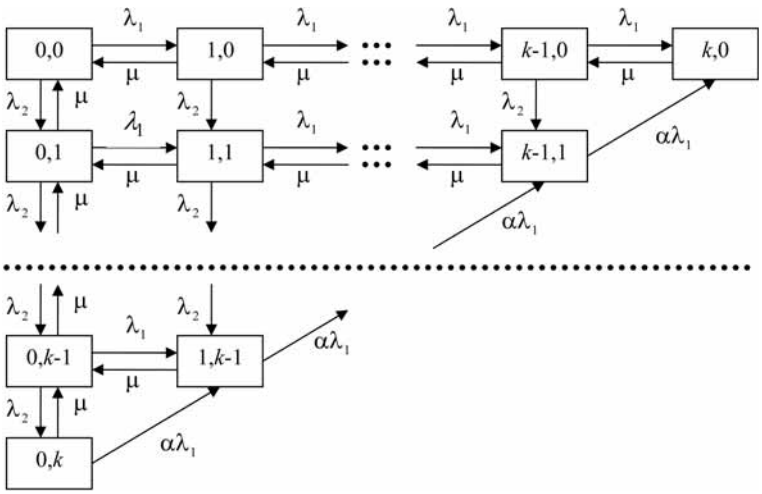


Рис. 3. Размеченный граф состояний системы $\vec{M}_2/M/1/k/f_2^1$

рекуррентных соотношений [6, 7]. В этом методе задача сводится к решению некоторой вспомогательной системы линейных алгебраических уравнений порядка $1/2k(k+1)$ с треугольной матрицей.

В настоящей статье используется альтернативный метод производящих функций [4] в его классическом варианте, предложенном еще полвека назад Х. Уайтом, Л. Кристи и Ф. Стефаном применительно к системе с ожиданием $\vec{M}_2/M/1/f_2$ [10, 11]. В ряде задач, к которым относятся, например, системы $\vec{M}_2/M/1/f_2$ или системы $\vec{M}_2/M/1/k/f_1^2$ [4], данный метод позволяет получить явное аналитическое решение. Далее будет показано, что аналитическое решение получается и для СМО класса $\vec{M}_2/M/1/\vec{k}/f_2^2$, в которую переходит наша система при $\alpha = 1$. Отметим, что выражение для вероятности потери для системы $\vec{M}_2/M/1/k/f_1^2$, найденное в работе [4] методом производящих функций, оказывается более простым, чем аналогичное выражение, полученное в работах [12, 13] методом рекуррентных соотношений.

В случае $\alpha < 1$ метод, описанный в работе [4], вообще говоря, не приводит к замкнутому аналитическому решению. Однако и в этом случае он дает ощутимый выигрыш по сравнению с методом рекуррентных соотношений. Так же, как и при использовании последнего метода, задача сводится к решению системы линейных уравнений с треугольной матрицей, однако порядок системы уменьшается с $1/2k(k+1)$ до $(k+1)$. Этот факт снижает сложность вычислений и при тех же требованиях к точности позволяет охватить более широкий диапазон изменения емкости буфера k.

Введем производящую функцию финальных вероятностей в виде

$$G(u, v) = \sum_{i=0}^{k} \sum_{j=0}^{k-i} p_{i,j} u^i v^j.$$

При этом условие нормировки (2) перепишем в виде

$$G(1, 1) = 1.$$

Умножая обе части равенства (1) на $u^i v^j$ и суммируя по всем допустимым значениям (i, j), получаем

$$G(u, v) = \{\mu(u - v)G(0, v) + \mu u(v - 1)G(0, 0) + [\alpha\lambda_1(u - v) + \lambda_1(1 - u)v + \lambda_2(1 - v)v] \times u \sum_{i=0}^{k} p_{i,\,k-i} u^i v^{k-i} + \alpha\lambda_1 u^{k+1}(v - u)p_{k,0}\} \times \{[\lambda_1 u(1 - u) + \lambda_2 u(1 - v) + \mu(u - 1)]v\}^{-1}. \quad (3)$$

Такое представление позволяет выразить функцию G через вероятности граничных состояний $p_{0,j}$ и $p_{i,k-i}$ размеченного графа (рис. 3). С помощью особого приема, разработанного в работе [4], удается исключить и вероятности $p_{0,j}$. Если затем воспользоваться стандартными рассуждениями [10, 11], основанными на соображениях аналитичности функции G, то задача сведется к решению некоторой системы линейных уравнений относительно неизвестных $p_{i,k-i}$.

Вначале рассмотрим более простую аналитически разрешимую задачу с детерминированным выталкивающим механизмом.

Система $\vec{M}_2/M/1/k/f_2^2$

Аналитическая разрешимость задачи вычисления вероятности потери при $\alpha = 1$ в случае относительного приоритета была показана в работе [4]. Этот результат сохраняет силу и при абсолютном приоритете.

Полагая $\alpha = 1$, после деления числителя и знаменателя (3) на μ находим

$$G(u, v) = [(u - v)G(0, v) + u(v - 1)G(0, 0) + u(1 - v)(\rho_2 v + \rho_1 u) \sum_{i=0}^{k} p_{i,\,k-i} u^i v^{k-i} + \rho_1(v - u)u^{k+1}\rho_{k,0}]\{[\rho_1 u(1 - u) + \rho_2 u(1 - v) + (u - 1)]v\}^{-1}, \quad (4)$$

где $\rho_i = \lambda_i/\mu$ означает коэффициент использования системы по i-му типу требований.

Получим распределение числа приоритетных требований N_1. Положим $q_n = \boldsymbol{P}\{N_1 = n\}$, $(n = \overline{0, k})$ и выразим производящую функцию q_n через G в виде

$$G_1(u) = \sum_{n=0}^{k} q_n u^n = G(u, 1). \quad (5)$$

Подставляя в уравнение (4) аргумент $v = 1$, сокращая на $(u - 1)$ и используя тождества

$$G(0, 1) = \sum_{j=0}^{k} p_{0,j} = q_0,\ p_{k,0} = q_k, \qquad (6)$$

находим

$$G_1(u) = \frac{q_0 - \rho_1 q_k u^{k+1}}{1 - \rho_1 u}. \qquad (7)$$

Функция (7) имеет полюс первого порядка при $u = \rho_1^{-1}$. Вместе с тем, согласно (5), она должна быть полиномом степени k. Чтобы устранить эту особенность, приравняем к нулю вычет G_1 в указанном полюсе

$$\mathbf{Res}_{u=\rho_1^{-1}} G_1(u) = 0,$$

откуда выводим

$$q_k = q_0 \rho_1^k.$$

Затем из условия нормировки $G_1(1) = 1$ вычисляем q_0 и $G_1(u)$

$$q_0 = \frac{1 - \rho_1}{1 - \rho_1^{k+1}},\ G_1(u) = \frac{1 - \rho_1}{1 - \rho_1^{k+1}} \frac{1 - (\rho_1 u)^{k+1}}{1 - \rho_1 u}. \qquad (8)$$

Производящая функция (8) отвечает усеченному геометрическому распределению

$$q_n = \frac{(1 - \rho_1)\rho_1^n}{\left(1 - \rho_1^{k+1}\right)},\ (n = \overline{0, k}).$$

Зная его, нетрудно найти все технические характеристики СМО, касающиеся приоритетных заявок. Основной из них в сетевых задачах является вероятность потери приоритетного запроса $P_{\text{пот}}^{(1)}$. Для более детального анализа приоритетного трафика также представляют интерес: среднее число приоритетных заявок в системе $\bar{n}^{(1)}$; средняя длина очереди приоритетных заявок $\bar{n}_{\text{оч}}^{(1)}$; вероятность ожидания $P_{\text{ож}}^{(1)}$ (вероятность того, что вновь подошедший приоритетный запрос встанет в очередь); среднее число каналов, занятых этими запросами $\bar{n}_{\text{зк}}^{(1)}$.

Опуская несложные вычисления, получим

$$P_{\text{пот}}^{(1)} = q_k = \frac{(1 - \rho_1)\rho_1^k}{\left(1 - \rho_1^{k+1}\right)},$$

$$P_{\text{ож}}^{(1)} = \sum_{n=1}^{k-1} q_n = \frac{\left(1 - \rho_1^{k-1}\right)\rho_1}{\left(1 - \rho_1^{k+1}\right)},$$

$$\bar{n}_{\text{оч}}^{(1)} = \sum_{n=1}^{k} (n-1) q_n = \frac{\rho_1^2}{1 - \rho_1} - \frac{(1 - \rho_1)\rho_1^{k+1}}{\left(1 - \rho_1^{k+1}\right)}, \qquad (9)$$

$$\bar{n}_{\text{зк}}^{(1)} = 1 - q_0 = \frac{\left(1 - \rho_1^k\right)\rho_1}{\left(1 - \rho_1^{k+1}\right)},$$

$$\bar{n}^{(1)} = \bar{n}_{\text{оч}}^{(1)} + \bar{n}_{\text{зк}}^{(1)} = \frac{\rho_1}{1 - \rho_1} - \frac{(k+1)\rho_1^{k+1}}{\left(1 - \rho_1^{k+1}\right)}.$$

Приведенные характеристики полностью идентичны аналогичным показателям для однопотоковой системы $M/M/1/k$ с коэффициентом использования ρ_1 [5]. Этот факт вполне естественен, так как при такой организации СМО приоритетные пакеты как бы "не замечают" остальные требования.

Теперь исследуем общее число пакетов в системе $N = N_1 + N_2$. Определим вероятности $r_n = \boldsymbol{P}\{N = n\}$ и введем их производящую функцию

$$G_\Sigma(u) = \sum_{n=0}^{k} r_n u^n = G(u, u).$$

Полагая $v = u$ в (4), сокращая на $(u - 1)$ и используя вместо (6) тождества, касающиеся вероятностей r_n,

$$G(0, 0) = p_{0,0} = r_0,\ \sum_{i=0}^{k} p_{i,k-i} = r_k,$$

получим для G_Σ представление, аналогичное (7), в котором числовые параметры q_1, ρ_1, q_k следует заменить на r_0, $\rho = \rho_1 + \rho_2$, r_k.

Функция G_Σ, подобно функции (7), будет иметь особенность в полюсе $u = \rho^{-1}$. Устраняя ее так, как это было показано выше, и нормируя результат на единицу при $u = 1$, находим

$$G_\Sigma(u) = \frac{1 - \rho}{1 - \rho^{k+1}} \frac{1 - (\rho u)^{k+1}}{1 - \rho u}.$$

Следовательно, общее число запросов также распределено по усеченному геометрическому закону,

но с параметром $\rho = \rho_1 + \rho_2$, отвечающим полной загрузке

$$r_n = \frac{(1-\rho)\rho^n}{\left(1-\rho^{k+1}\right)},\ (n = \overline{0,k}).$$

Легко заметить, что среднее число всех запросов в системе $\bar{n}$, средняя длина общей очереди $\bar{n}_{\text{оч}}$, среднее число каналов, занятых всеми запросами $\bar{n}_{\text{зк}}$ и вероятность ожидания неприоритетного запроса $P_{\text{ож}}^{(2)}$ находятся по формулам, аналогичным (9), где ρ_1 должно быть заменено на ρ. Для запросов низшего приоритетного класса отсюда получаем

$$\bar{n}^{(2)} = \bar{n} - \bar{n}^{(1)},\ \bar{n}_{\text{оч}}^{(2)} = \bar{n}_{\text{оч}} - \bar{n}_{\text{оч}}^{(1)},$$
$$\bar{n}_{\text{зк}}^{(2)} = \bar{n}_{\text{зк}} - \bar{n}_{\text{зк}}^{(1)}.$$

Как видим, неприоритетные запросы подстраиваются под приоритетные, занимая оставшиеся свободными места на обслуживание и в накопителе таким образом, чтобы суммарные показатели соответствовали однопотоковой СМО $M/M/1/k$ с полной нагрузкой ρ.

Несколько сложнее находится вероятность потери неприоритетного запроса $P_{\text{пот}}^{(2)}$. Представим ее в виде суммы $P_{\text{пот}}^{(2,0)} + P_{\text{пот}}^{(2,1)}$, где $P_{\text{пот}}^{(2,0)} = r_k$ означает вероятность потери запроса на входе системы, а $P_{\text{пот}}^{(2,1)}$ есть вероятность его потери из накопителя путем выталкивания. Введем также вероятность вытеснения приоритетным запросом какого-либо неприоритетного

$$P_{\text{выт}} = r_k - q_k = \frac{(1-\rho)\rho^k}{\left(1-\rho^{k+1}\right)} - \frac{(1-\rho_1)\rho_1^k}{\left(1-\rho_1^{k+1}\right)}.$$

Фиксируем некоторый достаточно большой интервал времени $[0, T]$. В среднем в этом интервале за счет вытеснения в накопитель дополнительно попадут $\lambda_1 T P_{\text{выт}}$ запросов. Вместе с тем, за тот же период в среднем $\lambda_2 P_{\text{пот}}^{(2,1)}$ неприоритетных запросов будут вытеснены из накопителя. В силу установившегося режима работы системы и законов сохранения [14], указанные величины должны совпадать. Вычисляя отсюда $P_{\text{пот}}^{(2,1)}$, находим

$$P_{\text{пот}}^{(2,1)} = \frac{(1-\rho)\rho^k}{\left(1-\rho^{k+1}\right)}\left(1+\frac{\rho_1}{\rho_2}\right) - \left(\frac{\rho_1}{\rho_2}\right)\frac{(1-\rho_1)\rho_1^k}{\left(1-\rho_1^{k+1}\right)}.$$

Общий случай $\alpha \neq 1$ будет разобран во второй части статьи.

Список литературы

1. **Гнеденко Б. В., Даниелян Э. А., Димитров Б. Н., Климов Г. П., Матвеев В. Ф.** Приоритетные системы обслуживания. М.: МГУ, 1973.
2. **Джейсуол Н.** Очереди с приоритетами. М.: Мир, 1973.
3. **Башарин Г. П.** Некоторые результаты для систем с приоритетом // Массовое обслуживание в системах передачи информации. М.: Наука, 1969. С. 39—53.
4. **Avrachenkov K. E., Vilchevsky N. O., Shevlyakov G. L.** Priority queueing with finite buffer size and randomized push-out mechanism // Performance evaluation, 2005. Vol. 61. No 1. P. 1—16.
5. **Клейнрок Л.** Теория массового обслуживания. М.: Машиностроение, 1979.
6. **Башарин Г. П.** Обслуживание двух потоков на однолинейной системе с ограниченным числом мест для ожидания и абсолютным приоритетом // Известия АН СССР. Техническая кибернетика. 1967. № 5. С. 106—116.
7. **Башарин Г. П.** О пуассоновских обслуживающих системах с абсолютным приоритетом // Массовое обслуживание в системах передачи информации. М.: Наука, 1969. С. 3—20.
8. **Bondi A.** An analysis of finite capacity queues with priority scheduling and common or reserved waiting areas // Computers and operations research. 1989. Vol. 16. No. 3. P. 217—233.
9. **Hegde N., Avrachenkov K. E.** Service differentiation and garantees for TCP elastic traffic // Lecture notes in computer science. 2002. Vol. 2511. P. 159—168.
10. **White H., Christie L. S.** Queueing with preemptive priorities or with breakdown // Operations research, 1958. Vol. 6. No. 1. P. 79—95.
11. **Stephan F. F.** Two queues under preemptive priority with Poisson arrival and servive rates // Operations research. 1958. Vol. 6. No. 3. P. 399—418.
12. **Kapadia A. S., Kazmi M. F., Mitchell A. C.** Analysis of a finite capacity non-preemptive priority queue // Computers and operations research. 1984. Vol. 11. No. 3. P. 337—343.
13. **Kapadia A. S., Chiang Y. K., Kazmi M. F.** Finite capacity priority queues with potential health applications // Computers and operations research. 1985. Vol. 12. No. 4. P. 411—420.
14. **Бронштейн О. И., Духовный И. М.** Модели приоритетного обслуживания в информационно-вычислительных системах. М.: Наука, 1976.

ИНФОРМАЦИЯ

УДК 025.4

С. А. Афонин, канд. физ.-мат. наук, вед. науч. сотр., e-mail: serg@msu.ru,
Д. Д. Голомазов, мл. науч. сотр., e-mail: denis.golomazov@gmail.com,
А. С. Козицын, канд. физ.-мат. наук, вед. науч. сотр., e-mail: alexanderkz@mail.ru,
НИИ механики МГУ имени М. В. Ломоносова

Использование систем семантического анализа для организации поиска научно-технической информации

Предлагается подход, позволяющий улучшать результаты обработки запросов поисковыми системами путем их интеграции со специализированными базами знаний, которые могут предоставлять развернутую информацию о персональных интересах пользователя. Предложенные методы и средства опробованы на разрабатываемой в МГУ системе поиска АСТАИ и системе учета и анализа научной информации ИСТИНА.

Ключевые слова: *информационный поиск, персонофицированный поиск, онтология, уточнение запросов*

Введение

Высокие темпы роста объемов текстовой информации, распределенной на современных сетях передачи данных, востребованность той или иной тематической информации в обществе приводят к необходимости создания новых методов, средств и систем, позволяющих более эффективно, в автоматизированном режиме осуществлять ее поиск и анализ, хранение и передачу по запросу [1]. Традиционно используемые для этих целей полнотекстовые поисковые системы нацелены на реализацию универсального поиска, результаты которого не зависят от конкретной предметной области и не учитывают ее специфику. В первую очередь это объясняется отсутствием возможности быстро и адекватно определить интересы конкретного пользователя. Одной из возможных альтернатив традиционным решениям является интеграция поисковых систем со специализированными базами данных и знаний. Такой подход перспективен при построении специализированных поисковых систем, которые создаются как для управления данными, распределенными в хранилищах отдельных организаций и корпораций, так и для сбора и анализа данных по определенным предметным областям в сети Интернет. Острая потребность в таких системах стимулируется также и активным развитием технических средств хранения и обмена информации, которое наблюдается в последние годы.

В контексте целей настоящей публикации появляется возможность объединить отдельные, заинтересованные в сотрудничестве группы ученых и конструкторов из различных территориально распределенных научных центров, лабораторий и институтов в единую структуру, которую принято именовать виртуальной организацией [2]. Необходимость создания таких виртуальных научных организаций обусловлена тем обстоятельством, что проведение большинства исследований в настоящее время невозможно или неэффективно с использованием вычислительных средств, хранилищ данных и других ресурсов только одной организации.

Для проведения распределенных научных исследований и конструкторских работ необходимо поддерживать режимы совместного использования всеми участниками высокопроизводительных вычислительных установок, находящихся в различных вычислительных центрах, архивов, специализированных решателей, библиотек программ, рабочих документов и баз данных (например, характеристик материалов). Таким образом, одной из важнейших задач при организации распределенных исследований является построение эффективной системы информационного обеспечения всех участников работ.

Основными задачами таких систем являются: автоматизация процесса распределенной обработки информации [3]; автоматизация процесса обмена данными между участниками; сбор информации о предмете исследования из внешних и внутренних по отношению к отдельной организации-участнику информационных источников. Для проведения эффективного поиска информации необходимо использовать полнотекстовые поисковые системы, которые способны осуществлять сбор и индексацию информации как из внешних (включая Интернет), так и из внутренних (организации, корпорации) источников, в том числе из общего хранилища данных, а также распределенную обработку информации [4]. Во внутреннем хранилище данных при работе на распределенной информационно-вычислительной среде файлы сохраняются в произвольном формате. Они, как правило, сопровождаются аннотацией и набором характеризующих такие файлы параметров, которые индексируются поисковой системой. Поиск по аннотациям позволяет быстро найти необходимый файл. Поисковая система при этом должна учитывать не только текст вводимого запроса, но и оценивать область интересов конкретного пользователя.

Персонифицированный поиск

Разработчики многих современных поисковых систем в настоящее время стараются поддерживать и развивать механизмы персонифицированного поиска, позволяя системе подстраиваться под пользователя и учитывать его персональные интересы. Одним из таких механизмов, который, например, активно используется в Google, является фиксация переходов по ссылкам и анализ истории запросов как конкретного пользователя, так и всех пользователей системы в целом. Однако такой подход не позволяет в полной мере получать представление о профессиональных интересах пользователя и о наиболее значимых для него темах, поскольку содержит очень много "шума". В качестве основных проблем на этом направлении можно назвать следующие.

- Переход пользователя на найденный документ ("клик") означает только тот факт, что пользователь не отсеял этот документ по краткой аннотации на странице результатов поиска. Поисковая система не может оценить, представляет ли найденный документ интерес для пользователя. Практика показывает, что более 80 % документов, на которые пользователь переходит при поиске, оказываются для него неинтересными. Как следствие, учет контекста этих документов не может значительно улучшить результаты последующего поиска.
- Использование истории запросов лучше отражает персональную заинтересованность пользователя, однако медленно адаптируется к его информационным потребностям, поскольку поисковая система не может оценить смысловую связь между запросами.

Одним из эффективных методов, который способен разрешить отмеченные проблемы, является интеграция поисковой системы с другими существующими в организации (корпорации) информационными приложениями, базами знаний и базами данных, описывающими интересы (предпочтения) отдельного пользователя.

Система ИСТИНА

В качестве источника информации для создания интегрированной поисковой системы в масштабах объединения (корпорации) может выступать система для учета и анализа продукции научного содержания отдельной организации. В рамках работ, направленных на интеграцию в единое информационное пространство и анализ научных данных различных подразделений МГУ имени М. В. Ломоносова, коллективом с участием авторов разрабатывается Интеллектуальная Система Тематического Исследования НАучно-технической информации (ИСТИНА), доступная по адресу URL: http://istina.imec.msu.ru.

Для улучшения работы научных организаций и, как следствие, повышения эффективности развития науки в масштабах государства необходимо перманентно анализировать информацию о результатах деятельности отдельных ученых и коллективов исследователей. Основными результатами деятельности организаций, входящих в научное сообщество, как правило, считаются публикации сотрудников, их руководство курсовыми, дипломными и диссертационными работами, чтение лекций, результаты патентных исследований, участие в конференциях и ряд других. При этом, как показывает практика, далеко не все результаты такой деятельности представлены в открытом доступе в Интернет. Зачастую единственным источником подобной информации могут служить лишь годовые отчеты сотрудников научных организаций, представленные с той или иной степенью подробности.

Естественным образом возникает необходимость автоматизированной (с участием человека) обработки данных из подобных научных отчетов в целях количественного и качественного анализа эффективности научной деятельности отдельного коллектива, вклада каждого его участника и возможной корректировки на основе такого анализа планов, мер и мероприятий по их выполнению.

К системе предъявляются следующие требования по функциям, которые она должна выполнять.

- Автоматизированный ввод данных из полу- и неструктурированных текстовых файлов, содержащих результаты научной и учебной деятельности сотрудников.
- Задание поискового запроса на естественном языке в свободной форме.
- Вычисление типовых аналитических запросов, требующих интеллектуальной обработки исходных данных.

• Реализация логического вывода новых данных из существующих. Например, если пользователь вводит запрос "бигармонические уравнения", а в формальном описании (модели, онтологии и др.) содержится информация о том, что метод конечных элементов может использоваться для решения бигармонических уравнений, то система может выдать среди результатов поиска научные статьи, посвященные методу конечных элементов. При этом система должна в удобном для восприятия пользователем виде указать причину включения этих статей в результаты поиска.

• Автоматизированная настройка системы на конкретное научное направление в рамках более широкой области научного знания.

• Автоматизированное обновление данных для их поддержания в актуальном состоянии.

• Возможность интеграции информации, содержащейся в системе, с другими хранилищами.

• Отсутствие необходимости в использовании большого объема ручного труда экспертов.

• Перечисленные требования определяют специфику разрабатываемого программного комплекса. Одна из главных его особенностей состоит в том, что он предназначен для обработки специальных классов документов (годовых научных отчетов сотрудников, BibTeX[1]-файлов, списков публикаций и др.). Отсюда, в частности, следует, что о конкретной статье сотрудника может быть доступна только ограниченная информация, например, название, авторы, место и год публикации. В большинстве систем анализа научных материалов широко используются списки библиографических ссылок между работами. В рамках задачи, рассматриваемой в настоящей работе, такая информация в общем случае недоступна.

Другой особенностью системы является необходимость использования интеллектуальных алгоритмов выделения, обработки и поиска данных. Она обусловлена требованием выполнения сложных аналитических запросов, адекватно описывающих содержательную сторону документов, на неструктурированных данных в отсутствие обучающих выборок и ручного труда экспертов.

Указанные особенности, в частности, подчеркивают отличие задачи, которая решается в рамках настоящей работы, от широко известного класса задач, связанных с построением электронных библиотек, более ориентированных на индексацию и предоставление доступа к электронным информационным ресурсам. В подобных системах меньше внимания уделяется тематическому анализу содержания работ и "привязке" данных к модели той или иной проблемной области.

В рамках предлагаемого подхода задача выполнения аналитических запросов к коллекции научной информации разбивается на четыре крупных подзадачи, а именно:

• построение формальной модели рассматриваемой области знания;

• загрузка данных о результатах научной деятельности сотрудников из исходных текстов;

• установление связей между загруженной информацией о результатах научной деятельности и понятиями построенной онтологии области знания;

• выполнение аналитических запросов к данным.

Рассмотрим эти подзадачи более подробно.

Построение формальной модели области знания. В настоящей работе модель области знания задается с помощью онтологии. Для ее построения используются методы лингвистического и статистического анализа текстов, в частности, метод шаблонов Херста [6] и контекстный анализ результатов поиска в Интернете с помощью поисковых систем. В качестве основного внешнего источника информации об области знания, которая является отдельным разделом науки, принято решение использовать анонсы проходящих научных конференций, называемые в научной среде "*Call for Papers*" (CFP), которые можно получить из открытых источников. Документы CFP содержат основные сведения о конференциях, в том числе название, место и даты проведения, состав программного комитета, описание конференции, список направлений области знания, работы по которым принимаются на рассмотрение. Этот подход имеет несколько преимуществ по сравнению с традиционно используемыми. В частности, один из них — тот факт, что конференции, как правило, содержат актуальную, полную и достаточно надежную информацию, отражающую текущее состояние рассматриваемого раздела науки. Они позволяют также получить информацию на различных уровнях детализации при затрате сравнительно небольших усилий. На основе информации о конференциях можно выделить ключевые понятия области знания и связи между ними, т. е. заполнить онтологию рассматриваемой области научного знания.

Алгоритм заполнения онтологии области знания состоит из следующих этапов.

• *Выделение ключевых слов (терминов)* [7]. Из коллекции CFP извлекаются ключевые слова, встречающиеся в списках направлений исследований, которым посвящены конференции, а также в названиях конференций. Методика выделения основана на применении методов статистического анализа с использованием результатов поиска в Интернете с помощью поисковых систем. Каждое ключевое слово или словосочетание попадает в класс "термин" онтологии. Таким образом, после завершения данного этапа формируется онтология, заполненная экземплярами понятия "термин". На дальнейших этапах на основе лингвистического и статистического анализа ключевых слов создаются экземпляры отношений, связывающие термины между собой и со специальными классами онтологии.

• *Классификация терминов по специальным понятиям онтологии.* С помощью методов лингвистического анализа (в частности, шаблонов Херста) для каждого ключевого слова или словосочетания проверяется, относится ли оно к одному из специальных кон-

[1] http://www.bibtex.org/

цептов онтологии. Если предварительно удалось выделить название некоторого научного направления из названия конференции, то в онтологии создается экземпляр отношения типа "связан" между рассматриваемым ключевым словом и выделенным научным направлением.

- *Организация терминов в иерархию в рамках каждого из специальных понятий.* Для каждого специального понятия выполняется процесс организации терминов, которые являются его экземплярами, в иерархическую структуру. В частности, на этом этапе происходит построение дерева направлений исследований в рамках рассматриваемой области знания.

- *Установление связей между экземплярами.* На данном этапе устанавливается связь каждого из терминов с одним или несколькими научными направлениями. Кроме того, между некоторыми ключевыми словами, являющимися экземплярами специальных понятий, устанавливаются отношения, определяемые спецификой этих понятий. Например, экземпляр "метод *k* ближайших соседей" концепта "метод" может находиться в отношении "используется для решения" с экземпляром "задача классификации объектов" понятия "задача".

Загрузка данных. Процесс загрузки данных включает первичную обработку вводимых пользователем данных и выделение из них необходимой информации. К такой информации относятся, например, список публикаций сотрудника с указанием даты, места, названия публикации, информация о конференциях и проектах, в которых участвовал сотрудник, а также некоторые другие сведения.

Для представления выделенной информации используется онтология SWRC (*Semantic Web for Research Communities*) [5]. Она включает такие концепты, как человек, организация, публикация, конференция, проект, а также связи между ними. Все необходимые понятия для описания сведений, содержащихся в исходных документах, с точки зрения внешних результатов научной деятельности (т. е. без анализа содержательной составляющей публикаций, конференций и других видов деятельности) уже формализованы в этой онтологии.

Таким образом, задача загрузки данных состоит в заполнении онтологии SWRC экземплярами, выделяемыми из информации, которую вводят сотрудники. Ключевым требованием эффективного решения этой задачи является необходимость создания удобного интерфейса ввода данных с поддержкой большого числа различных форматов. В частности, в упомянутой выше системе ИСТИНА реализован автоматизированный разбор библиографических ссылок в различных формах, BibTeX-записей, а также коллекций научных результатов, расположенных на широко распространенных системах хранения научных публикаций, например, на портале eLibrary.ru.

Установление связей между загруженной информацией и онтологией. Установление связей между информацией из загруженных текстов, содержащих результаты научной деятельности сотрудников, и формализованной моделью области знания необходимо для выполнения аналитических запросов. На предшествующих этапах из исходных документов выделяется лишь общая, количественная информация о научной деятельности сотрудника, например, в каких конференциях он участвовал и какие работы опубликовал. Этап связи необходим для получения информации о содержательной стороне деятельности сотрудника, например, каким областям знания посвящены его работы, какие задачи в этих областях он решал, какие методы и средства применял для решения поставленных задач.

Следует заметить, что процесс установления отмеченных выше связей приводит к необходимости классификации публикаций, конференций и других объектов по направлениям исследований в рамках рассматриваемой области знания. Однако этим он не ограничивается. Для более полного описания результатов научной деятельности необходимо выделить из публикаций дополнительную информацию, например, решаемые в работе задачи, используемые методы и средства. Отметим, что в рамках процесса установления связей проводится поиск дополнительной информации об объектах, выделенных из загруженных в систему документов. В частности, выполняется поиск аннотаций публикаций, списков литературы, программ конференций, тематики журналов, в которых опубликованы работы, и других данных подобного характера. Поиск выполняется с помощью поисковых систем в Интернете.

В рамках используемой модели представления знаний процесс установления связей между данными из загружаемых документов и моделью области знания заключается в слиянии двух заполненных онтологий, а именно онтологии SWRC и онтологии области знания. Более конкретно он сводится к фиксации отношений (ссылок) между экземплярами этих двух онтологий. Примером такой ссылки может быть отношение "is About" между экземпляром онтологии SWRC "публикация в журнале Nature под таким-то названием" и понятием онтологии биологии "структура белка". Проставление подобных ссылок необходимо для дальнейшего выполнения таких запросов, как "выдать все публикации, посвященные изучению белков".

Выполнение аналитических запросов к данным. Процесс выполнения аналитических запросов к данным состоит в организации взаимодействия между конечным пользователем системы и построенной моделью области знания. Такая модель включает как общую информацию об области знания, так и данные о результатах научных исследований сотрудников организации. Пользователь должен иметь возможность выполнять аналитические запросы, используя максимально удобный интерфейс. При этом необходимо реализовать функцию перезаписи запроса, в частности, его расширения и сужения. Например, если пользователь ищет все математические публикации за последний год, посвященные группам Ли, то в случае, ес-

ли их найдено мало, система должна предложить пользователю выдать все публикации по более широким направлениям, например, теории групп или алгебре в целом. И, наоборот, при поиске публикаций или конференций, посвященных математическому анализу, система может предложить пользователю уточнить запрос, предоставив возможность выбора конкретного направления в рамках этой области знания.

Отметим, что использование онтологий позволяет частично свести решение задачи выполнения аналитических запросов к решению технической задачи реализации языка запросов к онтологиям. В частности, перезапись запроса выполняется автоматически с помощью механизмов логического вывода. В качестве языка запросов к онтологиям в предлагаемом подходе используется язык SPARQL, получивший в 2008 г. статус рекомендации консорциума W3C.

В настоящее время система ИСТИНА позволяет в удобном для конечного пользователя режиме вводить данные о публикациях путем автоматизированного разбора информации из библиографических ссылок и BibTeX-записей. На основе введенных в хранилище системы данных о публикациях для каждого сотрудника автоматически создается отдельная "домашняя" страница, содержащая список его публикаций. В настоящее время проводится внедрение системы ИСТИНА в НИИ механики МГУ имени М. В. Ломоносова, в дальнейшем планируется использовать систему в других подразделениях Московского университета.

Отдельный модуль системы ИСТИНА на основе анализа научной деятельности пользователя строит его информационный портрет и передает это описание поисковой системе.

Автоматизированная система тематического анализа информации — АСТАИ

Разрабатываемая коллективом с участием авторов поисковая система АСТАИ [8] позволяет осуществлять: разовый поиск информации; непрерывный сбор информации; группировку и предварительную обработку данных для последующего анализа; классификацию информации по древовидным классификаторам; контекстный поиск; персонифицированный поиск; индексацию документов из Интернет; индексацию аннотаций документов из внутреннего хранилища. Ниже кратко представлены основные особенности этой системы [9].

Автоматическая рубрикация текстов с использованием неограниченного числа рубрикаторов позволяет определять тематику текстов и лучше описывать информационную потребность пользователя при построении поискового запроса.

Наличие персональных рубрикаторов и архивов позволяет пользователю держать под рукой все необходимые ему документы, содержащиеся в хранилище и полученные путем сканирования Интернета, удобно располагать такие документы на экране и быстро получать к ним доступ. Кроме того, система может на основании персональных архивов корректировать результаты поиска в соответствии с реальной информационной потребностью пользователя.

Сервисно-ориентированная архитектура (SOAP) позволяет проводить интеграцию поисковой системы с другими информационными системами, использующимися различными участниками работ. Это обстоятельство дает возможность осуществлять индексацию специализированных хранилищ (например, банка моделей, материалов или проектов), а также встраивать интерфейс поиска в привычную для участника среду разработки.

Открытый код системы позволяет ее модифицировать, совершенствовать и подстраивать под задачи, отражающие специфику деятельности отдельного коллектива.

Использование такой системы при распределенной работе большого научного коллектива значительно упрощает процессы распределения и использования информации всеми участниками совместно выполняемых работ.

Использование описания пользователя для уточнения запросов

Обеспечение политики безопасности в системах АСТАИ и ИСТИНА осуществляется на основе совместного репозитария пользователей. Такой подход позволяет этим системам обмениваться данными, имеющими отношение к конкретному пользователю, в том числе характеризующими его профессиональные интересы.

Используя онтологию описания предметной области, система ИСТИНА способна обеспечить построение иерархии направлений, задач, алгоритмов и ключевых понятий, которые описывают интересы конкретного пользователя. Первичной информацией для такого построения являются его научные работы, зарегистрированные в системе, а также данные из Интернета.

Описание передается в систему поиска в виде связанного графа классифицированных понятий. При осуществлении поиска пользователь может отметить пункт "Учитывать мои работы". После отметки этого пункта поисковая система осуществляет расширение и уточнение запроса пользователя на основе полученного графа понятий.

На первом этапе обработки поискового запроса оценивается близость этого запроса к одному из основных направлений работы пользователя. Такая оценка необходима, поскольку пользователь может являться автором работ сразу по нескольким научным направлениям, каждое из которых характеризуется собственными ключевыми понятиями. Далее запрос уточняется и расширяется на основе подграфа ключевых понятий направления, связанного с запросом.

Постановка задачи. Автоматическое уточнение запроса пользователя применяется для выделения из всех результатов выполнения запроса тех документов,

которые представляют для него интерес. Ручное составление дополнительных подзапросов является очень трудоемким процессом и требует от пользователя высокой квалификации не только в его профессиональной области, но и в области теории информационного поиска. Автоматизация построения множества уточняющих и расширяющих слов возможна как на основе онтологий, так и на основе кластеризации [10, 11] и тезаурусов [12].

При поиске специалист ищет информацию по конкретному понятию в своей предметной области и, как правило, не хочет задумываться над его возможным значением в других областях. Например, если авиаконструктор ищет информацию по запросу "тормозные диски" в своей области интересов, то его запрос необходимо автоматически уточнять ключевыми понятиями предметной области "тормозные диски" AND ("самолет" OR "авиация" OR ...).

После проведения поиска по словам должна осуществляться классификация полученного массива документов и оценка их близости классу интересов пользователя. В результате пользователю будут на первой же странице результатов поиска показаны нужные документы, и ему не придется просматривать десятки не представляющих для него интереса страниц с описанием автомобильных и прочих дисков.

Расширение запроса. Подход, основанный на расширении запросов, используется для устранения трудностей, обусловленных наличием синонимов и близких понятий в языке, в целях улучшения полноты поиска. В системе АСТАИ для расширения запроса предлагается использовать информацию о связях слов запроса с ключевыми понятиями, которые могут быть получены от системы ИСТИНА.

Процесс расширения подобен процессу расширения по тезаурусам. Каждое слово дополняется списком связанных с ним дочерних понятий с учетом типа, степени и направления связи. Например, запрос "алгоритм RSA" будет дополнен понятиями "теорема Эйлера", "открытый ключ" и др. Такое дополнение позволит найти документы, которые посвящены обсуждению связанных с алгоритмом RSA вопросов, однако не содержат его явного упоминания.

Для апробации предложенного подхода использовалось описание пользователя, построенное в системе ИСТИНА по конкретному пользователю по его научным статьям. Информационный портрет пользователя описывался понятиями: "Персональные данные"; "Качество программного обеспечения"; "Метрика программного обеспечения"; "Тестирование программного обеспечения". Сравнение результатов выдачи по запросам "microsoft" и "microsoft AND ("Персональные данные" OR "Качество программного обеспечения" OR "Метрика программного обеспечения" OR "Тестирование программного обеспечения")" показывает, что в результатах первого запроса представляющий интерес для пользователя документ находился только на семнадцатой позиции, а в результатах второго запроса только один документ из первых двадцати не соответствовал интересам пользователя.

Заключение

Представленные в настоящей работе подходы на основе интеграции поисковой системы с системами описания области интересов пользователя для автоматической или автоматизированной коррекции запроса пользователя могут применяться для улучшения точности и полноты результатов поиска.

Работа выполнена при частичной финансовой поддержке гранта РФФИ 09-07-00366-а и частичной финансовой поддержке гос. контракта 07.514.11.4116 Министерства образования и науки РФ.

Список литературы

1. **Васенин В. А., Афонин С. А.** К разработке моделей эффективного поиска информации в сети Интернет // Сб. трудов Всероссийской научной конференции "Научный сервис в сети Интернет 2003". 22—27 сентября 2003 г., г. Новороссийск. М.: Изд-во МГУ, 2003. С. 252—255.

2. **Уорнер М., Витцель М.** (пер. Ю. Леонова) Виртуальные организации. Новая форма ведения бизнеса в XXI веке. М.: Добрая книга, 2005.

3. **Васин Р. А., Козицын А. С., Васенин В. А., Шундеев А. С.** Информационно-экспертная система для исследования структурных и механических свойств материалов // Автоматика и телемеханика. 2005. № 7. С. 171—179.

4. **Васенин В. А., Афонин С. А., Козицын А. С., Шундеев А. С.** Поиск в сверхбольших хранилищах данных и высокопроизводительные системы с массовым параллелизмом // Труды международной конференции "Программные системы: теория и приложения ИПС РАН". Переславль-Залесский: Физматлит, 2004. С. 211—228.

5. **Sure Y., Bloehdorn S., Haase P., Hartmann J., and Oberle D.** The swrc ontology — semantic web for research communities // In Proc. of the 12th Portuguese Conf. On Articial Intelligence (EPIA 2005). Springer. 2005. Vol. 3803. P. 218—231.

6. **Hearst M. A.** Automatic acquisition of hyponyms from large text corpora // Proc. of the 14th conf. on Computational linguistics. Nantes, France. 1992. Vol. 2. P. 539—545.

7. **Голомазов Д. Д.** Выделение терминов из коллекции текстов с заданным тематическим делением // Информационные технологии. 2010. № 2. С. 8—13.

8. **Афонин С. А., Козицын А. С., Васенин В. А**. Автоматизированная система тематического анализа информации // Информационные технологии. Приложение. 2009. № 4. 32 с.

9. **Афонин С. А., Козицын А. С.** Использование онтологий в поисковых системах // Материалы Всероссийской конференции с международным участием "Знания-Онтологии-Теории". 2009, 22—24 ноября, г. Новосибирск. 2009. Vol. 2. P. 47—52.

10. **Cutting D. R., Karger D. R., Pedersen J. O., Tukey J. W.** Scatter/gather: A cluster based approach to browsing large document collections // 15th Ann Int'l SIGIR. Copenhagen, Denmark. 1992. P. 318—329.

11. **Титов А. С.** Кластеризация на основе гравитационного метода // Межвузовский сборник статей. 2003. Выпуск 2 (7). С. 51—60.

12. **Браславский П. И.** Тезаурус для расширения запросов к машинам поиска Интернета: структура и функции // Компьютерная лингвистика и интеллектуальные технологии. Труды Междунар. конф. Диалог'2003. 11—16 июня 2003 г., Протвино. М.: Наука, 2003. С. 95—100.

УДК 004.4'416

Н. И. Вьюкова, стар. науч. сотр., e-mail: niva@niisi.msk.ru,
В. А. Галатенко, д-р физ.-мат. наук, стар. науч. сотр., e-mail: galat@niisi.msk.ru,
С. В. Самборский, стар. науч. сотр.,
НИИ системных исследований РАН

Совместное решение задач выбора и планирования команд в условиях дефицита регистров

Рассмотрен метод генерации кода, обеспечивающий оптимальное совместное решение задач выбора и планирования команд при заданном ограничении на число доступных регистров. Метод позволяет максимально использовать параллелизм исполнения команд, а при дефиците регистров — автоматически порождать код для временного сохранения и восстановления регистров или, возможно, перевычисления значений.

Ключевые слова: *оптимизация кода, выбор команд, планирование потока команд, ЦЛП*

Введение

Задачей генерации кода в компиляторе является преобразование промежуточного представления высокого (или среднего) уровня, используемого на стадии оптимизации, в последовательность машинных команд. Обычно этот процесс включает проходы выбора команд, планирования и распределения регистров. Цель выбора команд — сформировать некоторый набор машинных команд, реализующих заданное вычисление. Цель планирования — составить расписание выполнения выбранного набора команд. Поскольку во время распределения регистров при их недостатке приходится добавлять команды для временного сохранения некоторых регистров в памяти и их последующего восстановления, то после распределения регистров может выполняться повторное планирование.

Недостатком описанного выше традиционного подхода к генерации кода является то обстоятельство, что задачи выбора команд, их планирования и распределения регистров взаимосвязаны и поэтому результат их последовательного решения может быть неоптимальным. В качестве примера рассмотрим генерацию кода для архитектуры, включающей два вычислительных сопроцессора, которые могут работать параллельно. При этом некоторые вычислительные команды могут выполняться как на одном, так и на другом сопроцессоре. Если один из сопроцессоров выполняет данные команды быстрее, то классические методы выбора инструкций, такие как [1], будут генерировать код, использующий только этот, более быстрый сопроцессор. В результате при планировании кода не будет использован параллелизм на уровне команд. Кроме того, может возникнуть дефицит регистров быстрого сопроцессора. Устранить этот недостаток можно путем совместного решения задач выбора и планирования инструкций с учетом ограничения по использованию регистров. Такой подход способен обеспечить оптимальную загрузку и параллельную работу вычислительных устройств обоих сопроцессоров и сбалансированное использование их регистров.

Идея представленного в работе решения состоит в том, чтобы свести задачи выбора и планирования команд с ограничениями на число доступных регистров к некоторой задаче математического программирования, например к задаче целочисленного линейного программирования (ЦЛП) или к задаче определения истинности логической формулы без кванторов (SAT, задача выполнимости булевых формул). В контексте настоящей работы используются методы ЦЛП, поскольку их применение к планированию команд хорошо изучено [2].

В случае дефицита регистров предлагаемый подход к генерации кода обеспечивает также автоматический спиллинг (временное сохранение регистра с последующим восстановлением). В качестве временного хра-

нилища значений может выступать как память, так и регистры других, менее дефицитных классов. При этом обеспечивается оптимальный выбор регистров для спиллинга и планирование команд сохранения и восстановления. При наличии общих подвыражений вместо спиллинга может также применяться перевычисление значений, если это оказывается выгоднее.

Далее в статье рассматривается использование метода генерации кода, основанного на совместном решении задач выбора и планирования команд, для линейных участков программ. Однако он может быть применен также к простым циклическим участкам (конвейеризация циклов с выбором команд).

Предлагаемый в настоящей работе подход ориентирован, прежде всего, на применение в компиляторах для микропроцессоров, используемых во встроенных и бортовых системах. Обычно архитектуру таких процессоров стараются максимально упростить ввиду ограничений на размер плат, энергопотребление, тепловыделение. По этой причине они, как правило, не содержат аппаратных средств оптимизации, таких как аппаратное планирование, переименование регистров, ротация регистрового файла. Предполагается, что высокая эффективность приложений в таких системах должна достигаться за счет высокого качества машинного кода, которое обеспечивается оптимизирующим компилятором (или программистами, разрабатывающими приложения на ассемблере).

Архитектура микропроцессора, как правило, включает средства, которые дают возможность порождать высокоэффективный код:

- специализированные команды, ориентированные на определенный класс приложений;
- параллелизм на уровне команд (суперскалярные процессоры или процессоры с очень длинным машинным словом (VLIW — *Very Large Instruction Word*)).

Вместе с тем, также в силу аппаратных ограничений, такие микропроцессоры могут иметь ряд свойств, затрудняющих генерацию кода. В их числе — малочисленные классы специализированных регистров, сложные правила, описывающие возможность параллельного исполнения команд. Отмеченные обстоятельства делают оправданным применение в компиляторе трудоемких методов оптимизации кода, в том числе основанных на метода математического программирования.

```
N = {R, M}                    # Нетерминальные символы
A = {x, r}                    # Терминальные символы
P = {                         # Правила
    1. R = r :2               # Загрузка значения на регистр */
    2. R = x(R,R):3           # Операция умножения
    3. M = R :2               # Сохранение регистра в памяти
    4. R = M :2               # Восстановление регистра из памяти
}
```

Рис. 1. Пример грамматики, описывающей систему команд

Метод отложенного выбора инструкций

Рассмотрим основные составляющие предлагаемого метода, который можно назвать *генерацией кода с отложенным выбором инструкций* [3]. Входными данными для генератора кода служит грамматика, описывающая систему команд целевого процессора, а также дерево (или лес деревьев), задающее входной линейный участок. Далее будем использовать простой пример грамматики, представленной на рис. 1.

Здесь нетерминальный символ `M` соответствует ячейке памяти, `R` — регистру. Терминальный символ `x` обозначает инструкцию умножения, а `r` — переменную или константу в памяти. Правило *1* описывает команду чтения значения переменной или константы из памяти в регистр; правило *2* описывает команду умножения. Правила *3* и *4* описывают, соответственно, запись значения регистра во временную переменную в памяти и восстановление регистра. Они введены для реализации спиллинга. Фактически правила *1* и *4* соответствуют одной и той же машинной инструкции (`load`). Однако семантически это разные действия, поэтому для них введены разные правила. Целые константы, заданные через двоеточия после правил, описывают латентность соответствующих машинных команд.

Генерация кода состоит из двух шагов:

1. Первичный выбор инструкций;
2. Планирование с окончательным выбором инструкций.

Для первичного выбора команд используется модификация известного алгоритма BURG [1], одного из декларативных методов выбора команд, обзор которых можно найти в работе [4]. Алгоритм BURG основан на методах синтаксического разбора, где командам процессора соответствуют правила вывода в некоторой особого вида контекстно-свободной грамматике (без однозначности разбора). Применение каждого правила имеет определенную цену. Цена дерева разбора при этом представляет собой сумму цен примененных правил, что позволяет находить оптимальный разбор методом динамического программирования.

Модификация алгоритма BURG заключается в том, что запоминаются все возможные команды (правила), применимые в узлах дерева, а не только наиболее дешевые. Окончательный выбор инструкций выполняется на втором шаге, во время планирования.

Пример входного дерева для выражения (*AB*)(*CD*) представлен на рис. 2. На этом примере будет продемонстрирован автоматический спиллинг регистров.

Индексы задают нумерацию узлов дерева. Каждому узлу сопоставляется множество виртуальных регистров, соответствующих набору нетерминальных символов во входной грамматике: R_1, M_1 для узла 1, R_2, M_2 для узла 2, и так далее, исходя из того, что результат

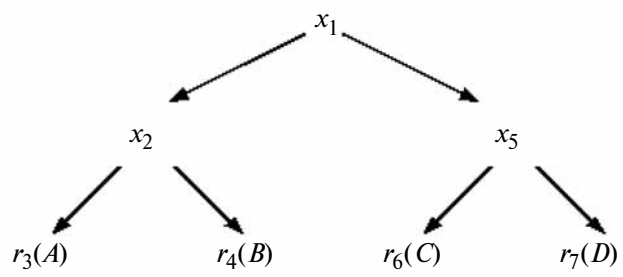


Рис. 2. Дерево, описывающее входное выражение

подвыражения в узле *i* может быть храниться либо в регистре R_i, либо в ячейке памяти M_i.

Заметим, что здесь понятие "виртуальный регистр" несколько отличается от общепринятого. Обычно в компиляторе под виртуальным регистром понимается просто переменная промежуточного представления программы. Предполагается также, что эта переменная простого типа (близкого к типам поддерживаемым аппаратно), которая может быть размещена на одном или нескольких физических (аппаратных) регистрах. Кроме этого, в обычно используемом SSA-представлении (*static single assignment form*) [5] виртуальные регистры используются таким образом, что присваивание значения каждому из них происходит только в одном месте. Для линейного участка это означает, что присваивание виртуальному регистру происходит ровно один раз.

В подходе, совмещающем выбор инструкций и их планирование, виртуальному регистру соответствует не просто определенное промежуточное значение вычисления (узел дерева в нашем примере), а промежуточное значение, лежащее в определенном хранилище (разные наборы регистров или память). По этой причине одному узлу с индексом 1 соответствует два разных виртуальных регистра R_1 и M_1. Кроме того, ослабляется SSA-требование, а именно допускается повторное вычисление виртуального регистра.

Более того, значение узла (результат подвыражения) может повторно вычисляться, помещаться в хранилища разного типа, храниться в нескольких экземплярах. Важно при этом, что где бы не находилось это значение, оно всегда одно и то же, так как это значение одного подвыражения. Таким образом, в случае линейного участка требование "однократного присваивания" подменяется на "присваивание одного и того же значения".

Для рассматриваемого примера на шаге первичного выбора команд будет сгенерировано множество *Com*, представленное на рис. 3.

Команды вида $M_i = R_i$ и $R_i = M_i$, соответствующие применению правил *3*, *4* входной грамматики, обычно являются избыточными. Однако в условиях дефицита регистров они могут быть включены в расписание так, чтобы обеспечить временное высвобождение регистров (спиллинг). Ниже будет показан пример автоматически сгенерированного выходного кода с использованием спиллинга.

Формулировка ЦЛП-задачи выбора и планирования инструкций

Для того чтобы найти оптимальный код для заданного входного вычисления, будем последовательно формулировать и решать ЦЛП-задачи для построения кода, укладывающегося в заданное число тактов $T = T_{\min}, T_{\min} + 1, \ldots$, пока решение очеред-

Узел	Команды			
	Правило *1*	Правило *2*	Правило *3*	Правило *4*
1		$R_1 = x(R_2, R_5)$	$M_1 = R_1$	$R_1 = M_1$
2		$R_2 = x(R_3, R_4)$	$M_2 = R_2$	$R_3 = M_2$
3	$R_3 = r_3$		$M_3 = R_3$	$R_3 = M_3$
4	$R_4 = r_4$		$M_4 = R_4$	$R_4 = M_4$
5		$R_5 = x(R_6, R_7)$	$M_5 = R_5$	$R_5 = M_5$
6	$R_6 = r_6$		$M_6 = R_6$	$R_6 = M_6$
7	$R_7 = r_7$		$M_7 = R_7$	$R_7 = M_7$

Рис. 3. Результат первичного выбора команд (множество *Com*) для дерева, представленного на рис. 2

ной ЦЛП-задачи не закончится успешно (подробнее об этом см. в работе [2]).

Для каждого значения T необходимо построить ЦЛП-задачу, эквивалентную совмещенной задаче выбора и планирования инструкций для линейного участка. Под эквивалентностью имеется в виду то, что каждой допустимой точке ЦЛП-задачи (набору переменных, удовлетворяющих ограничениям) соответствует программа, корректно вычисляющая заданное выражение, и, наоборот, каждой программе соответствует некоторая допустимая точка.

Так как задача определения эквивалентности программ алгоритмически неразрешима, то в общем случае не существует ЦЛП-формулировки, эквивалентной данной программе (или она не может быть конструктивно построена). Однако рассматриваемая задача значительно проще: построить ЦЛП-описание программ фиксированной длины T, составленных из инструкций конечного множества *Com*, описывая возможное взаимодействие команд при помощи виртуальных регистров.

Принимая во внимание отмеченные выше упрощения, построить эквивалентную ЦЛП-формулировку для задачи планирования с выбором команд несложно. В целом она аналогична формулировке работы [1]. Отличия рассматриваемой задачи касаются методов подсчета числа необходимых регистров и требования о том, чтобы каждая инструкция выполнялась ровно один раз. Здесь оно заменяется требованием того, что должно быть запланировано выполнение (возможно неоднократное) подмножества инструкций из *Com*, достаточного для вычисления заданного входного выражения. Неоднократное выполнение нужно для перевычисления значений и повторного спиллинга.

Множества *Com* и *Vreg*, используемые в формулировке задачи, строятся по заданному входному выражению:

- *Com* — множество команд, полученных на стадии первичного выбора (см. рис. 3);
- *Vreg* — множество виртуальных регистров, которое состоит из всех регистров, упоминаемых в командах из *Com*.

Перечисленные далее константы, используемые в формулировке, определяются свойствами целевого процессора.

Константа $T > 0$, определяет число тактов, за которое должна выполниться программа (через T тактов результаты должны быть готовы).

Константа *Nreg* задает число доступных физических регистров. В реальных процессорах присутствуют регистры разного типа, соответственно, для каждого типа требуется задать число регистров.

Цена регистра $v - H_v$, $\forall v \in Vreg$, H_v задает число физических регистров, которые необходимы для хранения значения на виртуальном регистре. В рассматриваемом примере $H_{\mathbf{R}i} = 1$, $H_{\mathbf{M}i} = 0$.

Множества *Write* и *Read* определяют зависимости по данным между командами из множества *Com*.

Множество $Write \subset Com \times Vreg \times N_0$, где N_0 — множество неотрицательных целых чисел, задает выходные регистры и латентности команд из *Com*: $(c, v, lat) \in Write$, если команда c записывает результат в регистр v с латентностью *lat*. В рассматриваемом примере команды вида $\mathtt{R}_i = \mathtt{x}(\mathtt{R}_j, \mathtt{R}_k)$ имеют латентность 3, латентность остальных команд равна 2.

Множество $Read \subset Com \times Vreg \times N_0$ задает входные регистры команд и "латентности чтения": $(c, v, lat) \in Read$, если команда c читает регистр v. Значение *lat* показывает, на каком такте с момента запуска команды считывается значение этого регистра. В данном случае предполагается, что латентность чтения всегда равна 0.

Ниже описаны переменные, используемые в формулировке задачи.

Массив $S_{c,\,t}$: $\forall c \in Com$, $\forall t \in [1,\ T]$, $S_{c,\,t} \in \{0,\ 1\}$. Массив S задает кратность запуска команд. $S_{c,\,t} = 1$ соответствует факту запуска команды c на такте t. Массив S полностью определяет набор исполняемых команд и расписание их запуска. В результате решения ЦЛП-задачи вычислим массив S, который однозначно соответствует выходному машинному коду с точностью до распределения физических регистров.

Массив $V_{v,\,t}$: $\forall v \in Vreg$, $\forall t \in [1,\ T+1]$, $V_{v,\,t} \in N_0$. Массив переменных V для каждого виртуального регистра определяет число его копий, хранимых на данном такте. Обычно это значение 0 или 1, хотя нет каких-либо запретов на хранение нескольких копий. Это бывает полезно при генерации кода для процессоров с командами, помещающими выходное значение на место одного из входных. В данном примере переменные V можно считать принимающими значения 0 и 1.

Для того чтобы в компактном виде записать ограничения ЦЛП-задачи, доопределим значения S и V для всех целых t:

$$\forall c \in Com,\ \forall t \notin [1,\ T],\ S_{c,\,t} = 0;$$

$$\forall v \in Vreg,\ \forall t > T+1,\ V_{v,\,t} = V_{v,\,T+1};$$

$\forall v \in Vreg$, $\forall t \leqslant 0$, $V_{v,\,t} = 1$, если v — входной регистр линейного участка; для остальных регистров $V_{v,\,t} = 0$.

Построим ограничения на переменные S и V такие, чтобы формулируемая ЦЛП-задача оказалась эквивалентна задаче построения кода в том смысле, как это описано в начале раздела.

Начнем со следующих ограничений, описывающих тот факт, что любое хранимое на виртуальном регистре значение должно быть получено путем запуска подходящей команды:

$$V_{v,\,t} - V_{v,\,t-1} = \sum_{\substack{(c,\,v',\,lat) \in Write \\ v' = v}} S_{c,\,t-lat}\ \forall t \in [1,\ T+1],\ \forall v \in Vreg.$$

Значение V для регистра v увеличивается на такте t только в результате запуска на такте $t - lat$ команды, пишущей в v, где *lat* — латентность этой команды. Пе-

ременная v' здесь является вспомогательной. Запись $(c, v', lat) \in Write_{v' = v}$ в операции суммирования означает, что суммирование проводится по всем тройкам (c, v', lat) из W, в которых $v' = v$.

Приведенные ниже ограничения разрешают запуск команд только в том случае, если их входные регистры готовы:

$$S_{c,\,t} \leqslant V_{v,\,t+lat} \;\forall\; t \in [1,\, T],\; \forall\; (c,\, v,\, lat) \in Read.$$

Заметим, что не для всех процессорных архитектур достаточно таких ограничений. В данном примере их достаточно, поскольку задана нулевая латентность чтения для всех команд.

Вообще, условия готовности входных данных для команды, ограничения на число используемых физических регистров и возможность совместного исполнения инструкций существенно зависят от нюансов архитектуры процессора. В первую очередь важны ответы на следующие вопросы: отслеживает ли процессор готовность входных регистров и поддерживается ли переименование регистров?

Теперь сформулируем условия на число используемых физических регистров:

$$\sum_{v \in Vreg} H_v \Big(V_{v,\,t} + \sum_{(c,\,v',\,lat) \in Write,\; v' = v} \;\sum_{t' = t(lat-1)} S_{c,\,t'}\Big) \leqslant Nreg,$$
$$\forall t \in [1,\, T+1].$$

Для каждого регистра v его цена H_v умножается на число хранимых копий v вместе с числом копий, которые находятся в процессе изготовления.

В рассматриваемом примере процессорная архитектура допускает запуск двух команд за такт, причем без каких-либо условий их совместимости. Эти ограничения записываются очень просто:

$$\sum_{c \in Com} S_{c,\,t} \leqslant 2,\; \forall t \in [1,\, T].$$

Для реальных процессоров совместимость команд описывается путем задания их таблиц резервирования. В таблице резервирования для каждого такта исполнения команды указывается множество ресурсов, требующихся ей на этом такте. Ресурсом может быть функциональное устройство, доступ к регистровому файлу и подобные им.

Последнее необходимое условие описывает то, что должны быть получены результаты — значение корневого узла входного дерева (иначе получим тривиальное решение $S \equiv 0,\; V \equiv 0$):

$$V_{R_1,\,T+1} + V_{M_1,\,T+1} \geqslant 1.$$

Заметим, что здесь безразлично, где будет получен искомый результат — на регистре R_1 или в памяти M_1. Если у линейного участка несколько выходных значений, то следует сформулировать соответствующее условие для каждого из них.

t	$\{c: S_{c,t} = 1\}$	Виртуальные регистры		Ассемблерный код
		Хранимые	"Недописанные"	
1	$R_3 = r_3 \quad R_4 = r_4$			load f0,A; load f1,B
2			$R_3\ R_4$	nop
3	$R_2 = x(R_3, R_4)$	$R_3\ R_4$		mul f0,f0,f1
4			R_2	nop
5	$R_7 = r_7$		R_2	load f1,D
6	$M_2 = R_2 \quad R_6 = r_6$	R_2	R_7	store f0,tmp_R2; load f0,C
7		R_7	$R_6\ M_2$	nop
8	$R_5 = x(R_6, R_7)$	$R_6\ R_7\ M_2$		mul f0,f0,f1
9	$R_2 = M_2$	M_2	R_5	load f1,tmp_R2
10			$R_2\ R_5$	nop
11	$R_1 = x(R_2, R_5)$	$R_2\ R_5$		mul f0,f0,f1
12			R_1	nop
13			R_1	nop

Рис. 4. Код, сгенерированный для входного выражения рис. 2, при $Nreg = 2$, $T = 13$

С учетом представленных выше соображений, все ограничения ЦЛП-задачи сформулированы и можно искать ее решение. Вместе с тем, из множества решений имеет смысл выбрать самое простое, без запуска избыточных команд и хранения лишних значений на регистрах. Для этого дополним задачу минимизируемой целевой функцией. Можно, например, минимизировать сумму всех элементов массивов S и V

$$\sum_{\substack{t \in [1, T] \\ c \in Com}} S_{c, t} + \sum_{\substack{t \in [1, T+1] \\ v \in Vreg}} V_{v, t}.$$

Рассмотрим результат решения построенной ЦЛП-задачи, причем в наиболее интересном случае, а именно — в условиях острого дефицита физических регистров. Не выталкивая регистры в память (и не изменяя заданный скобками порядок операций), нельзя вычислить выражение $(AB)(CD)$ менее чем на трех регистрах. Попробуем решить построенную ЦЛП-задачу для $Nreg = 2$. При $T < 13$ решения нет, однако выделив 13 тактов, удается сгенерировать код. Решение приведено на рис. 4.

Легко видеть, что сгенерированный код действительно использует всего два физических регистра (M_2 — значение второго виртуального регистра, размещенное в памяти). Результат достигается за счет того, что на такте 6 содержимое виртуального регистра R_2 сохраняется в память (и замещается значением переменной C), а на такте 9 загружается обратно. Таким образом, полностью автоматически выбран регистр для спиллинга и сгенерирован спилл-код.

Заключение

Совмещение выбора и планирования инструкций естественно возникает, если рассматривать как основную сущность не инструкции, а значения (виртуальные регистры). Заранее известны выходные виртуальные регистры (функции входных данных линейного участка). Некоторые инструкции из набора возможных "конкурируют между собой" за право изготовить эти значения. При этом каждой инструкции требуются определенные входные виртуальные регистры, которые изготовляются на конкурентной основе другими инструкциями и так далее.

При одновременном выборе инструкций и их планировании несложно учесть ограничения по числу регистров. В рассмотренной формулировке совмещенной задачи выбора и планирования инструкций разрешено повторное вычисление любого виртуального регистра, а также исполнение любой инструкции более одного раза. Набор возможных инструкций пополнен всеми допустимыми командами копирования между физическими регистрами разных типов, а также между регистрами и памятью.

В результате автоматически генерируется код для спиллинга регистров, сохранения значений на регистрах другого типа (менее дефицитных), или повторного вычисления некоторого значения, если это дешевле, чем хранение. Реализуется также дублирование значений, если это необходимо для параллельной обработки.

Обычно эти преобразования реализуются в компиляторах по отдельности, и, как правило, не в полном объеме. Часто реализуется единственное преобразование, без которого нельзя обойтись (спиллинг). При этом нет гарантии, что будет оптимально выбран регистр для выталкивания в память. В предлагаемом подходе все управление регистрами происходит комплексно и гарантированно оптимальным образом.

Список литературы

1. **Fraser C. W., Henry R. R., Proebsting T. A.** BURG — Fast optimal instruction selection and tree parsing // SIGPLAN Notices 27. 4 (Apr. 1992). P. 68—76.

2. **Самборский С. В.** Формулировка задачи планирования линейных и циклических участков кода // Программные продукты и системы. 2007. № 3 (79). С. 12—16.

3. **Вьюкова Н. И.** Генерация кода с отложенным выбором инструкций. — М.: Изд-во НИИСИ РАН, 2010. С. 80—96.

4. **Вьюкова Н. И., Галатенко В. А., Самборский С. В.** К проблеме выбора машинных инструкций в генераторах кода // Моделирование и визуализация. Многопроцессорные системы. Инструментальные средства разработки ПО: сб. статей. М.: Изд-во НИИСИ РАН, 2009. С. 3—31.

5. **Allen R., Kennedy K.** Optimizing Compilers for Modern Architectures. Morgan Kaufmann Publishers, 2002.

ИНФОРМАЦИЯ

Международный конгресс по интеллектуальным системам и информационным технологиям

IS&IT'12

2—9 сентября 2012 года,
Россия, Черноморское побережье, Геленджик-Дивноморское.
Официальный сайт конгресса URL: http://icai.tsure.ru

УДК 004.62:004.42

В. В. Вейбер, канд. техн. наук, e-mail: webvad@tpu.ru,
А. В. Кудинов, канд. техн. наук, доц., зав. лаб., e-mail: kudinovav@tpu.ru,
Томский политехнический университет,
Н. Г. Марков, д-р техн. наук, проф., вице-президент,
ОАО "Востокгазпром", e-mail: markovng@vostokgazprom.ru

Интеграция информационных систем нефтегазового предприятия на основе отраслевого стандарта и принципов SOA

*Рассматривается технология интеграции данных нефтегазового предприятия, объединяющая SOA и подход на основе модели предметной области с использованием отраслевого стандарта **PRODML** в качестве метамодели.*

***Ключевые слова**: MDA, SOA, PRODML, метамодель данных, интеграционная платформа*

Введение

В настоящее время большинство предприятий нефтегазовой отрасли не имеет рациональной стратегии их автоматизации, что приводит к отсутствию интеграции между множеством закупленных программных продуктов, необходимых различным службам, и является препятствием к образованию единого информационного пространства предприятия [1]. Разнородность этих программных продуктов не позволяет эффективно обмениваться данными между ними, принимать и передавать управляющую информацию.

В данной статье рассмотрены следующие вопросы:

- особенности автоматизации промышленных предприятий с непрерывным производством;
- проблемы интеграции технологических данных внутри предприятия;
- особенности информатизации промышленных предприятий нефтегазовой отрасли;
- подход к интеграции приложений с использованием метамодели данных;
- оригинальная технология интеграции данных нефтегазового предприятия с использованием метамодели предметной области на основе стандарта PRODML;
- пример решения практической задачи интеграции для нефтегазового предприятия.

Модель CIM

Общепризнанной моделью информационного обеспечения предприятия, отвечающего современному уровню автоматизации, является иерархическая модель *Computer Integrated Manufacturing* (CIM) [2]. Согласно этой модели, системы верхнего уровня оперируют агрегированными данными на относительно больших временных промежутках, а системы нижнего уровня имеют дело с большим потоком данных реального времени. Для связи по дискретной оси времени систем верхнего уровня с событиями реального времени, управляемыми системами нижнего уровня, применяются промежуточные системы цехового уровня (*Manufacturing Execution Systems* — MES). Нижний уровень модели представляют элементы сбора данных (датчики), устройства с программным управлением (например, контроллеры станков с числовым программным управлением) и автоматизированные системы диспетчерского управления SCADA (*Supervisory Control and Data Acquisition*), взаимодействующие с оборудованием. С системами SCADA взаимодействуют MES-системы, собирающие от них данные о техно-

логических процессах и позволяющие оперативно управлять производством предприятия. Решения MES, в свою очередь, предоставляют агрегированную информацию для имеющихся на предприятии систем класса *Enterprise Recourse Planning* (ERP) (для управления ресурсами предприятия) и *Business Intelligence* (BI) (аналитических систем). На практике для интеграции разных уровней предприятия часто используются универсальные интеграционные платформы, такие как Microsoft BizTalk [3], IBM WebSphere [4] и т. п. Но использование специализированного интеграционного решения для определенной предметной области может быть намного эффективнее универсального в части как производительности, так и стоимости.

Существуют различные классификации видов интеграции [5]. В данной работе рассмотрен тип интеграции по данным для интеграции информационных систем (ИС) и приложений нефтегазовых предприятий.

Особенности интеграции данных нефтегазодобывающего предприятия

Согласно модели CIM, взаимодействие ИС предприятия выполняется по двум направлениям — горизонтальному и вертикальному. Вертикальная интеграция обеспечивает автоматизацию обмена данными, во-первых, между уровнем диспетчерского управления технологическими процессами (SCADA) и уровнем управления производством (цехами) предприятия (MES), во-вторых, между уровнем MES и уровнем управления предприятием (ERP, BI). Горизонтальная интеграция обеспечивает сбор данных внутри одного уровня управления, например интеграция MES с прикладными ИС уровня управления производственными процессами (обработка данных геологоразведки, моделирование нефтяного резервуара и т. д.).

Рассмотрим процесс движения технологических данных более подробно (рис. 1, см. третью сторону обложки). Технологические данные собираются на нефтегазовых промыслах средствами автоматизированных средств управления технологическими процессами (АСУТП) при добыче, подготовке и транспортировке углеводородного сырья. Автоматизация верхнего уровня управления в АСУТП обеспечивается SCADA-системами. Значительная часть данных с технологического уровня управления поступает на уровень управления производственной деятельностью в MES-систему. MES-система, в первую очередь, автоматизирует работу производственных служб разных уровней управления производством, позволяет планировать и контролировать выполнение технологических мероприятий по обслуживанию и ремонту оборудования, координировать работу всех производственных служб компании. В качестве входных данных в эти системы кроме ряда технологических параметров поступают данные о состоянии скважинного фонда, об объемах добычи и подготовки углеводородного сырья, результаты химических анализов передаваемой в магистральные газо- и нефтепроводы продукции. Часть данных MES-системы (агрегированные технологические данные, данные по фонду скважин и т. д.) необходима информационным системам, решающим специализированные инженерные задачи. Эти задачи для большинства MES-систем находятся за рамками ее функционала (например, моделирование трубопроводов, построения геологических моделей месторождений и т. п.). Часть данных из MES-системы, обычно агрегированных и консолидированных (объемы добычи углеводородного сырья, объемы его потерь, использования продукции на собственные нужды и т. п.), должна попадать в системы автоматизации управления ресурсами предприятия и интеллектуального анализа, т. е. на более высокий уровень управления компанией (ERP-системы и BI-системы) [6].

Таким образом, в соответствии с моделью CIM, можно выделить основные интеграционные задачи, которые необходимо решить для создания единого информационного пространства типичного нефтегазового предприятия. Ими являются задачи: вертикальной интеграции (MES ←→ SCADA, MES ←→ ERP) и горизонтальной интеграции (MES ←→ специализированные ИС).

Для вышеупомянутых задач в нефтегазовой отрасли не существует высоко унифицированных стандартов интеграции, исключение составляет стандарт OPC (*OLE for Process Control*) [7], который хоть и не является специализированным стандартом для этой предметной области, но прекрасно решает задачу интеграции систем SCADA- и MES-уровней.

Процесс интеграции данных нефтегазового предприятия сопряжен с рядом трудностей, которые в значительной мере усложняют решение задач вертикальной и горизонтальной интеграции.

Во-первых, сложность горизонтального взаимодействия систем заключается в территориальной распределенности существующих АСУТП и разнородности протоколов передачи данных. По мере развития производства, оснащения их разнородными АСУТП и технологическими базами данных (БД) на предприятии возникает сложная информационная структура сбора и обработки производственных данных.

Во-вторых, на каждом предприятии формируется свой собственный набор используемых прикладных программных средств и приложений. Как пример, на одном предприятии могут сосуществовать средство моделирования нефтяных резервуаров *Eclipse* от компании Schlumberger [8], средство контроля производства *Prosper and Gap* [9] от Petroleum Experts и система визуализации *ProcessNet* от Matrikon [10]. Проблемы, вызванные фрагментированным выбором систем, препятствуют их интеграции. Интеграция продуктов, взаимодействие которых не предусмотрено их разработчиками, является дорогостоящей и трудоемкой задачей, требующей много ресурсов. Вместе с тем,

ввиду больших денежных затрат и времени уже потраченного на эти приложения, их замена является нерациональной.

В-третьих, на современном нефтегазовом предприятии существуют большие объемы технологической информации (информация с датчиков производственного оборудования, результаты химических анализов, результаты диагностических исследований, данные по техническому обслуживанию и ремонту технологического оборудования, геологические данные и т. п.). Большие объемы данных усложняют процесс их передачи при интеграции ИС и приложений.

Все перечисленное указывает на то, что решение задач интеграции данных ИС нефтегазового предприятия является достаточно трудоемким процессом.

Подход к интеграции ИС по данным с использованием метамодели

Использование метамодели предметной области позволяет сделать процесс обмена данными между ИС простым и прозрачным, этот подход гарантирует, что все сообщения между ИС будут правильно поняты и интерпретированы [11]. Наличие такой метамодели данных значительно сокращает число адаптеров, интегрирующих приложения и ИС предприятия, и позволяет вместо связи друг с другом через дорогие парные адаптеры связать их через общую модель данных (рис. 2, см. третью сторону обложки).

Предлагается применить этот подход для решения задачи интеграции ИС по данным нефтегазового предприятия ввиду его явных преимуществ.

В каждой ИС существует своя интерпретация предметной области, в связи с этим задача сопоставления сущностей разных ИС при их интеграции является достаточно сложной. Наличие метамодели данных предприятия, промежуточной для всех задействованных ИС, значительно облегчает эту задачу [12].

Метамодель предприятия либо может быть создана на основе концептуальных моделей ИС, используемых на предприятии (по сути, это создание новой оригинальной метамодели данных), либо может быть использована уже существующая метамодель, разработанная для нефтегазовой отрасли. Второй вариант наименее трудоемкий. Существует ряд готовых стандартов для интеграции ИС в нефтегазовой предметной области, среди них как самый зрелый мы выделяем стандарт PRODML [13], который был взят за основу для формирования общей модели данных нефтегазодобывающего предприятия.

Стандарт PRODML

PRODML (*Production XML*) — это XML-стандарт, разрабатываемый ассоциацией Energistics при поддержке ведущих нефтяных компаний BP, Chevron, Shell, Statoil, Halliburton, Schlumberger, корпорации Microsoft и других организаций для нефтегазовой промышленности. Он призван поддерживать обмен информацией между приложениями и хранилищами данных [13]. Этот стандарт — расширение стандарта WITSML, широко используемого для передачи информации в процессе бурения скважин.

Основные области применения рассматриваемого стандарта следующие.

- Информация по скважинному фонду:
  - перечень скважин;
  - состояние скважин;
  - динамические показатели скважин (давление, температура, дебит).
- Производственное оборудование.
- Ответственный персонал.
- Бурение и буровое оборудование.
- Исследования скважин.
- Химический анализ продукции.

Основная задача, которую стандарт PRODML должен решить, состоит в том, чтобы определить стандарт передачи данных между пунктом *A* и пунктом *B*, при этом пункты *A* и *B* могут быть различными нефтяными компаниями, поставщиками оборудования и услуг, программным обеспечением, и данные при этом могут быть различных временных масштабов [14]. Это могут быть, например, достаточно редкие данные по исследованию скважин, либо же данные реального времени для оперативного контроля производственного процесса добычи углеводородов.

Основа PRODML — это иерархичная структура, описывающая бизнес-объекты предприятия (*Product Flow Model*). Она формируется из трех основных элементов: модуль (*unit*), сеть (*network*) и модель (*model*) (рис. 3). Модуль — это любой объект для моделирования или получения информации. Модуль может представлять как сложные объекты (завод по переработке сырья или сепаратор), так и простые объекты (кран или задвижка). Каждый модуль содержит порты (*port*), которые позволяют соединять модули между собой через узлы (*node*), отражая направление движения продукции. Существует возможность подключения модулей "многие ко многим". Сеть — коллекция связанных модулей.

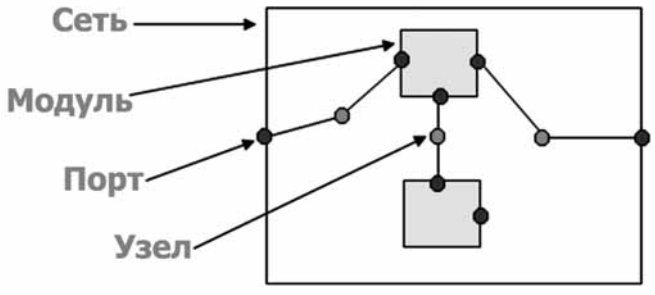


Рис. 3. Основные элементы Product Flow Model

И наконец, модель — объектное представление предприятия, которое является совокупностью сетей. Иерархия определяет также движение потоков продукции между бизнес-объектами предприятия и их направление (рис. 4, см. третью сторону обложки).

Характеристики модулей и потока продукции отражены в документе *Product Volume Report.* Он используется для передачи такой информации как ежедневный дебет нефти или газа каждой скважины либо таких характеристик, как буферное, линейное и затрубное давления, температура, скорость потока и т. д. Сообщение может быть как простым (данные по одной скважине), так и сложным (информация о целом месторождении).

Модель данных, предлагаемая стандартом PRODML, покрывает практически весь процесс с момента добычи нефтегазовой продукции до момента ее реализации. Использование этой модели данных значительно упрощает решение задачи интеграции, но стандартная модель не может учесть все особенности предприятия, каждое из которых уникально и имеет свою специфику. Поэтому необходимы инструменты, позволяющие расширять PRODML для конкретного предприятия. На данный момент использование стандарта PRODML не поддержано инструментальными средствами, поэтому актуальной является задача разработки этих средств и технологии их использования.

Технология SOA

Важным аспектом интеграции является то, как реализован транспортный уровень движения данных взаимодействующих систем. Общепризнанным подходом к интеграции является сервисно-ориентированная архитектура (*Service Oriented Architecture*, SOA) [5]. SOA представляет собой подход к организации архитектуры программных систем, предлагающий компоновать функциональные модули распределенных систем в виде служб (web-сервисов), которые могут быть вызваны любой программой формирования запросов сервиса. SOA и web-сервисы используют программный язык и платформонезависимые интерфейсы между приложениями. Открытые стандарты, описывающие web-сервисы, позволяют применять SOA для взаимодействия со всеми ИС и приложениями, используемыми на предприятии. Web-сервисы базируются на широко распространенных и открытых протоколах: HTTP, XML, UDDI, WSDL и SOAP. Web-службы и SOA становятся популярным и полезным средством усиления интеграционных решений для улучшения бизнес-процессов в нефтегазовой промышленности [15]. В связи с этим предлагается использовать технологию SOA для взаимодействия интегрируемых ИС и приложений нефтегазовых предприятий.

Предлагаемая технология интеграции

Суть предлагаемой технологии в том, чтобы объединить подход к интеграции на основе метамодели предметной области с архитектурой SOA и при этом использовать отраслевой стандарт PRODML в качестве основы метамодели для создания интеграционной платформы нефтегазового предприятия. Архитектура предлагаемого интеграционного решения для нефтегазовых предприятий приведена на рис. 5 (см. четвертую сторону обложки).

Метамодель данных позволяет описать все сущности предметной области и взаимоотношения между ними (например: скважинный фонд, исследования скважин, бурения, ремонтные работы, заявки и т. д.). Она отражает потенциально возможную структуру предприятия, от уровня сбора технологических данных до уровня финансовой отчетности и планирования.

На основе метамодели строится **частная модель**, отражающая структуру объектов конкретного предприятия — единый реестр объектов, которые необходимо задействовать в процессе интеграции ИС. Эта модель дополняется новыми объектами при необходимости их участия в процессе интеграции. Наличие частной модели решает задачу сопоставления объектов между различными ИС предприятия.

Интеграционная платформа представляет собой отдельное приложение, обеспечивающее взаимодействие всех задействованных ИС. Она не является местом хранения данных, платформа — посредник, позволяющий организовать интеграционные процессы внутри предприятия. Интеграционная платформа выполняет следующие функции:

- служит местом хранения частной модели предприятия и предоставляет возможность для ее модификации (расширения);
- служит инструментом для создания, настройки, исполнения и контроля всех интеграционных процессов (настройка интеграционного процесса включает в себя сопоставление объектов интеграции между ИС на основе частной модели предприятия);
- предоставляет возможность создания правил преобразования данных при передаче в ИС, приведения к общим единицам измерения;
- определяет режим обмена данными (по расписанию, по запросу пользователя, по изменению данных).

Для каждой ИС, задействованной в процессе интеграции, необходим **адаптер**, предоставляющий интеграционной платформе интерфейс к данным этой ИС в виде, описанном метамоделью предприятия. Создание адаптера для подключения программного продукта к интеграционной платформе является достаточно трудоемкой задачей, так как адаптер должен преобразовывать информацию из схемы данных интегрируемой ИС в схему данных метамодели предпри-

ятия, а этот процесс нельзя строго формализовать. Адаптер может представлять собой как сервис, так и приложение с пользовательским интерфейсом (если есть необходимость в настройке процесса пользователем, например, фильтрация объектов интеграции, ручной ввод или контроль в процессе передачи данных и т. п.). Обмен данными между адаптерами и интеграционной платформой может быть организован посредством web-сервисов в соответствии с архитектурой SOA, что обеспечит гибкость и прозрачность организации всей системы.

Вышеописанная архитектура положена в основу разработанных нами инструментальных средств, помогающих в реализации предлагаемой технологии на практике:

- *Product Flow Model Builder* — инструмент построения частной модели предприятия в соответствии с стандартом PRODML (рис. 6, см. четвертую сторону обложки);
- шаблон адаптера для ИС, позволяющий ускорить процесс его создания, избавив разработчика от реализации стандарта PRODML и транспортной логики;
- интеграционная платформа как приложение, реализующее вышеописанные функции.

Пример решения задач интеграции

Рассмотрим применение предложенной технологии на примере интеграции нескольких систем по данным. Интеграции по горизонтали подлежала корпоративная геоинформационная система управления производством "Магистраль—Восток" [16] (система класса MES) с двумя специализированными ИС "БАСПРО" для построения геологических моделей газоконденсатных месторождений [17] и ИС "OISPipe" для моделирования внутрипромысловых газосборных сетей [18]. Эта же MES интегрировалась по вертикали на основе стандарта OPC по технологическим данным с рядом SCADA-систем (Delta-V, RS/3 и т. д.), являющихся верхним уровнем АСУТП кустов скважин и установок подготовки газа и конденсата.

Сначала, с помощью оригинальной утилиты *Product Flow Model Builder* была построена частная модель газодобывающего предприятия ОАО "Томскгазпром" (см. рис. 6) на основе стандарта PRODML.

Затем для каждой из систем с использованием шаблона был создан адаптер. Все адаптеры подключались к интеграционной платформе, где была выполнена настройка взаимодействия всех систем. Пример настройки соответствий объектов показан на рис. 7. После выполнения этой процедуры все системы оказались включенными в единое информационное пространство ОАО "Томскгазпром".

В итоге, с помощью внедренного решения была реализована передача данных по производственным объектам промысла и фонду скважин между этими системами.

Основные показатели по которым была организована передача данных:

- скважины — давление (буферное, затрубное), температура (линейная, устьевая), состояние скважины, способ эксплуатации скважины, время работы/простоя, обводненность скважины, компонентный состав сырья;

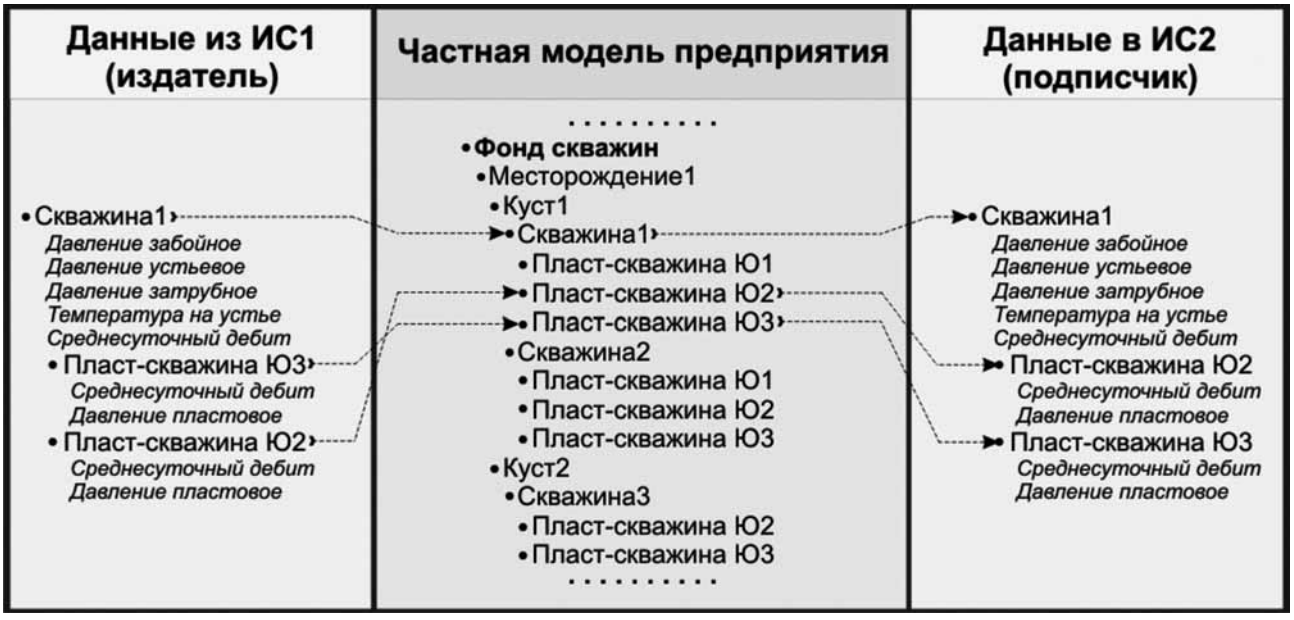


Рис. 7. Настройка соответствий объектов интеграции на основе частной модели

- пластовые характеристики — плотность воды, плотность нефти, газосодержание;
- пласт-скважины (эксплуатационные пласты) — добыча/выпуск сырья, добыча воды.

Внедренное решение позволило упростить наполнение БД систем "БАСПРО" и "OISPipe" историческими технологическими данными, а также обеспечить эти системы новыми данными при их поступлении из системы "Магистраль—Восток" без повторного ручного ввода пользователями.

Заключение

Рассмотрена проблема интеграции ИС по технологическим/производственным данным и предложена технология, позволяющая минимизировать затраты нефтегазового предприятия на решение этой проблемы. Положительный момент рассматриваемой технологии состоит в том, что она не требует реорганизации IT-структуры предприятия и может внедряться постепенно, по мере появления задач интеграции конкретных ИС. Она позволяет значительно ускорить и упростить интеграцию ИС, а также обеспечивает успешное эволюционирование ИС в ходе развития информационного пространства нефтегазового предприятия.

Список литературы

1. **ИТ** в газовой промышленности: оценка экспертов региональных компаний ОАО "Газпром". URL: http://www.connect.ru/article.asp?id=8850.
2. **ISO.** Industrial automation systems and integration — Open systems application integration framework — Part 1: Generic reference description — URL: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=30418.
3. **Microsoft** BizTalk Server. URL: http://www.microsoft.com/Rus/biztalk.
4. **WebSphere** software. URL: http://www-01.ibm.com/software/ru/websphere.
5. **Juric M.** SOA approach to integration. Birmingham: Packt Publishing Ltd., 2007. 366 p.
6. **Марков Н. Г., Сарайкин А. В.** Формирование единого информационного пространства газодобывающей компании // Oil&Gas journal. Russia. 2008. № 3 (16). С. 34—41.
7. **What** is OPC? URL: http://www.opcfoundation.org/Default.aspx/01_about/01_whatis.asp?MID=AboutOPC.
8. **ECLIPSE** 2010 — Reservoir Engineering Software. URL: http://www.slb.com/services/software/reseng/eclipse2010.aspx.
9. **IPM** Products. URL: http://www.petex.com/products.
10. **Matrikon** releases ProcessNet v3.1 for web-based decision support. URL: http://www.matrikon.com/news/56/index.aspx.
11. **Hollander D., High M.** Common Models in SOA: Tackling the Data Integration Problem. URL: http://www.progress.com/progress/dataxtend/docs/wp_do_not_forget_data.pdf.
12. **Bravo C., Aguilar J., Ríos-Bolívar A., Aguilar-Martin J., Rivas-Echeverría F.** A Generalized Data Meta-Model for Production Companies Ontology Definition // International journal of system applications, engineering & development. 2008. Issue 4. Vol. 2.
13. **PRODML**. Reference Architecture PRODML 1.0. URL: http://www.prodml.org.
14. **Blaker J.** PRODML scope statement version 2.0. URL: http://www.prodml.org.
15. **Zhang C., Orangi A., Bakshi A., Will Da Sie, and Prasanna V. K.** A service-oriented data composition architecture for integrated asset management // SPE Intelligent Energy Conference and Exhibition (IECE). April 2006. URL: http://pgroup.usc.edu/iam/papers/SPE-99983-DataComposition.pdf.
16. **Bogdan S., Kudinov A., Markov N.** Example of implementation of MES "Magistral-Vostok" for oil and gas production enterprise // Сборник трудов конференции CEE-SECR 2009. С. 131—136.
17. **Баспро** Оптима. URL: http://www.baspro.ru/programm.
18. **OISPipe.** URL: http://www.ois.ru/product.php? productName =OISPipe.

ИНФОРМАЦИЯ

Липаев В. В.

Проектирование и производство сложных заказных программных продуктов. – М.: СИНТЕГ, 2011. – 400 с.

ISBN 978-5-89638-119-8

Монография состоит из двух частей, в которых изложены методы и процессы проектирования и производства сложных заказных программных продуктов для технических систем реального времени. Все компоненты и комплексы программ должны соответствовать требованиям заказчика, высокому качеству и минимальным рискам посредством верификации, тестирования, испытаний и сертификации, обеспечиваемыми коллективами квалифицированных специалистов. При изложении активно используются современные международные и отечественные стандарты, планирование производственных процессов с учетом ограниченных экономических ресурсов крупных проектов.

Часть 1 посвящена методам системного проектирования комплексов программ, подбору и подготовке коллектива специалистов для проектирования и производства сложных программных продуктов. Изложено проектирование требований к компонентам и комплексам программ, а также требований к характеристикам качества и допустимым рискам при проектировании процессов производства программных комплексов. Представлено оценивание и прогнозирование сложности проектирования и экономических характеристик процессов производства заказных программных продуктов.

Часть 2 содержит основы промышленного производства сложных заказных программных продуктов. Изложены организация и реализация верификации и тестирования комплексов программ, тестирования потоков управления и потоков данных программных модулей и компонентов, планирование производства и тестирования компонентов и комплексов программ. Представлены тестирование сложных динамических программных продуктов и методы сопровождения программных комплексов. Изложены методы и процессы управления конфигурацией и документированием программных комплексов, а также испытания, удостоверение качества и сертификация сложных заказных программных продуктов с учетом стандартов.

Монография предназначена для руководителей предприятий и проектов технических систем, для специалистов, ответственных за проектирование и производство сложных заказных программных продуктов реального времени высокого качества, также может использоваться в качестве учебного пособия по программной инженерии.

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 13.01.2012 г. Подписано в печать 21.02.2012 г. Формат 60×88 1/8.
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.