РОССИЙСКАЯ АКАДЕМИЯ НАУК

ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР им. А.А. Дородницына

50 лет ВЦ РАН:

ИСТОРИЯ, ЛЮДИ, ДОСТИЖЕНИЯ

Вычислительный центр им. А.А. Дородницына Российской академии наук

Москва 2005

УДК [51+53+004] (06)

50 лет ВЦ РАН: история, люди, достижения.
- М.: ВЦ РАН, 2005. - 320 с.
ISBN 5-201-09837-1

В 2005 году Вычислительному центру им. А.А. Дородницына Российской академии наук исполнилось 50 лет.

Сборник содержит статьи, отражающие историю создания и развития Института, его подразделений. В нём представлены наиболее важные достижения и результаты, полученные за эти годы.

Компьютерная вёрстка

Е.А. Королёвой

Научное издание

ISBN 5-201-09837-1

ОТДЕЛ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ СИСТЕМ ПРОЕКТИРОВАНИЯ

Л.Л. Вышинский, Ю.А. Флёров, М.Г. Фуругян

(*с.* 248-272)

Отдел математического моделирования систем проектирования создан в 1973 г. по инициативе академиков РАН А.А. Дородницына и Н.Н. Моисеева под первоначальным названием "Отдел теории и методов автоматизации проектирования". Руководил отделом до 1990 г. академик РАН П.С. Краснощёков, в настоящее время – чл.-корр. РАН Ю.А. Флёров.

Научные исследования отдела проводятся по следующим направлениям:

1. Разработка математических методов структурно-параметрического описания объектов проектирования.
2. Разработка системы математического моделирования структурно-параметрического облика сложных объектов и их подсистем.
3. Разработка математических моделей функционирования сложных объектов проектирования и их подсистем.
4. Разработка и исследование иерархических структур принятия проектных решений.
5. Разработка методов последовательного анализа вариантов в задачах проектирования сложных технических объектов.
6. Разработка алгоритмов планирования вычислений в многопроцессорных системах реального времени.
7. Разработка САПР систем реального времени.
8. Комбинаторное представление графов и сетей.
9. Разработка технологии и инструментальных средств проектирования, создания и сопровождения прикладных информационно-вычислительных систем и систем автоматизированного проектирования.
10. Разработка программного обеспечения прикладных информационно-вычислительных систем и систем автоматизированного проектирования.

1. Основные научные результаты отдела

Среди основных научно-технических результатов деятельности отдела можно выделить следующие.

Разработаны начала математической теории проектирования сложных технических систем (СТС) и объектов [1-8].

Исследована и частично решена фундаментальная научная проблема создания и совершенствования теории и методологии решения основной концептуальной задачи параметрического проектирования сложных технических систем – задачи параметрического синтеза облика системы данного целевого назначения. Развиты методы, позволяющие формализовать этап синтеза и тем самым заложить основы математической теории проектирования сложных систем **[7].**

Одной из центральных проблем этапа синтеза является проблема корректной декомпозиции и структуризации процесса проектирования. Получены условия согласованности критериев на различных уровнях иерархии, достаточные для нахождения множества эффективных вариантов в многоуровневых системах автоматизированного проектирования сложных технических объектов [7, 8].

Для многокритериальных моделей синтеза предложены общие конструкции аппроксимирующих отношений, которые могут использоваться для предварительного отсева неперспективных вариантов. Поставлены и решены минимаксные задачи синтеза системы по вспомогательным аппроксимирующим отношениям. Уточнены и детализированы полученные условия согласованности критериев на различных уровнях иерархии, достаточные для нахождения множества эффективных вариантов в многоуровневых системах автоматизированного проектирования сложных технических объектов. Для многокритериальных моделей синтеза изучены общие конструкции аппроксимирующих отношений, которые могут использоваться для предварительного отсева неперспективных вариантов [7, 8].

Изучена корректность декомпозиции в пространстве оценок и в пространстве формальных описаний альтернативных вариантов проектируемой системы [7, 8].

Формализация иерархии описаний СТС и её моделей функционирования даёт основу для решения фундаментальной задачи автоматизации проектирования – задачи синтеза сложной технической системы, удовлетворяющей заданным функциональным характеристикам, и построения иерархической структуры задач проектирования СТС. Изучена проблема проектирования иерархических схем процесса проектирования и управления ими в интерактивном режиме. Изучена проблема согласования критериев оценки эффективности функционирования СТС в иерархической структуре задач проектирования [8, 9].

Исследован один из подходов к проблеме согласованности в иерархических системах автоматизированного проектирования. На содержательном уровне согласованность отношений предпочтения на k-м и $(k + 1)$-м уровнях иерархии означает, что при переходе от агрегированного на более детальный уровень описания системы сохраняется частичная упорядоченность альтернатив по отношению предпочтения на более детальном уровне. Получены достаточные условия согласованности и разработана иерархическая многокритериальная процедура улучшения опорного решения, полученного на агрегированном уровне описания, при наличии условий согласованности [7-9].

Приведённый цикл результатов и его практическая реализация получили широкую известность. Группа сотрудников отдела была отмечена в 1981 г. премией Совета Министров СССР в области науки и техники за разработку и внедрение первой очереди системы автоматизации проектирования сложных технических объектов.

Получен результат, связанный с теоремой Моцкина об альтернативе, но не опирающийся на неё: система строгих однородных линейных неравенств совместна в том и только том случае, если начало координат не принадлежит выпуклой оболочке ненулевых вектор-строк матрицы, определяющей систему неравенств, причём проекция начала координат на выпуклую оболочку является решением системы [10].

Получены необходимые и достаточные условия существования решения, слабо эффективного по векторному критерию, при условиях регулярности, не предполагающих выпуклости множества допустимых решений. Для любого доминируемого решения получен конечный набор направлений приближения к множеству слабо эффективных решений [11]. Предложен сходящийся метод построения множества эффективных векторов, устойчивый к вычислительным погрешностям [12, 13].

Разработана интеллектуальная инструментальная среда для автоматизации конструкторских и инженерных расчётов (ИИС ФАКИР). Система ИИС ФАКИР предназначена для организации и сервисного обеспечения решения различных инженерных задач, возникающих в процессе проектирования, а также для технологической поддержки создания и эксплуатации систем автоматизированного проектирования (САПР). Система может быть использована научно-исследовательскими и проектными организациями, разрабатывающими новые образцы техники, а также разрабатывающими или внедряющими САПР для различных предметных областей.

Инструментальные системы САПР необходимы для автоматизации создания расчётных, информационных и управляющих программ как при разработке САПР, так и в процессе их использования. Применение ИС позволяет существенно сократить сроки создания прикладных подсистем САПР, а впоследствии облегчить различные их модификации. Однако ИС необходимы и в процессе автоматизированного проектирования при постановке конкретных задач. Постановка задач проектирования включает построение моделей объектов проектирования, описание критериев и областей поиска при выборе оптимальных вариантов проекта, описание форм представления текстовой и графической информации о результатах решения, а также установление связей с другими задачами.

Таким образом, назначением ИИС ФАКИР является решение следующих основных задач:

— обеспечение пользователя базовыми средствами описания математических моделей объектов проектирования, частных инженерных задач и проектных процедур;

— автоматизация построения математических моделей, алгоритмов решения и рабочих программ для решения инженерных задач;

— обеспечение информационной связи между частными задачами;

— обеспечение параметрических расчётов и выбор оптимальных вариантов при решении инженерных задач;

— обеспечение хранения, поиска и анализа результатов решения инженерных задач;

— обеспечение представления результатов в удобном для пользователя виде, в том числе в виде графиков, номограмм и т.п. [14].

Разработана система ЭЛМОН для автоматизации проектирования схем соединений радиоэлектронного оборудования объектов машиностроения. Система ЭЛМОН внедрена на ряде предприятий авиационной промышленности [14].

Разработана технология и программное обеспечение автоматизации проектирования комплекса радиоэлектронного оборудования объектов машиностроения — САПР РЭО.

Принципиальная новизна состоит в создании в ЭВМ полной информационной модели комплекса оборудования объекта проектирования, включающей внутреннее представление протоколов согласования, принципиальных схем, схем соединений, монтажных схем, схем разводки щитов и др. технологической информации. Это позволяет говорить о "сквозной" автоматизации проектирования. Кроме того, предполагается максимальная автоматизация некоторых этапов проектирования при возможности вмешательства в любой момент в этот процесс со стороны конструктора. Наличие базы знаний, в которой накапливаются типовые конструкторские решения, позволяет существенно сократить время проектирования. Внедрение комплексной автоматизации проектирования РЭО позволяет поднять производительность труда при разработке РЭО от 2 до 5 раз при многократном повышении качества документации [14].

Создан программный комплекс обработки, хранения и визуализации табличных и функциональных зависимостей — система ТАБУС. В системе ТАБУС реализованы инструментальные средства, математические методы и программное обеспечения обработки, хранения и визуализации табличных и функциональных зависимостей для анализа и синтеза проектов, систем и процедур принятия решений [15].

Создано программное обеспечение автоматизации проектирования вычислительных систем реального времени. Исследована задача составления допустимого расписания в многопроцессорных системах с различными ограничениями на прерывания и связи между процессорами и на число переключений в каждый момент времени. Цикл этих работ подробнее описан в разд. 3.

Охарактеризованы транзитивные графы, допускающие упорядоченное вложение в плоскость, разработан полиномиальный алгоритм распознавания таких графов, и найдена асимптотика их числа [16].

При задании двух весов, принимающих два значения на каждом ребре сети, разработан полиномиальный алгоритм построения остовного дерева, минимального по первому весу при ограничениях по второму весу [16].

Исследованы графы без чередующихся циклов, для которых построено каноническое представление, получены формулы числа неизоморфных графов и долей k-связных графов (k = 0, 1, 2, ...) [16].

Предложена методика оценки запасов газа и коэффициентов перетоков, основанная на использовании блочной газодинамической модели месторождения и решении системы уравнений материального баланса. Метод обладает большей устойчивостью результатов по сравнению с ранее известными методами.

Разработаны новые численные методы решения обратной задачи, основанные на минимизации отклонений средних давлений в блоках от давлений, вычисленных с помощью модели (прямое сопоставление измеряемых параметров). Разработаны алгоритмы оптимального выбора разбиения месторождения на газогидродинамически связанные блоки. В его основу положен принцип минимизации разницы пластовых давлений по скважинам в пределах каждого блока.

Разработан метод прогнозирования оптимальных величин отбора газа по всем частям месторождения, при которых, во-первых, минимизируются перетоки газа между различными частями месторождения, и, во-вторых, выполняются различные ограничения на уровни добычи газа, определяемые экономическими и технологическими факторами.

Разработан программный комплекс, реализующий указанные методы. Проведены расчёты запасов газа и перетоков в Ямбургском, Шебелинском и Советабадском месторождениях [17-20].

Получена точная формула числа минимальных раскрасок интервального графа, что позволило разработать линейные алгоритмы для некоторого класса задач теории расписаний. Получен линейный алгоритм нахождения минимаксных раскрасок графа и асимптотическая формула числа интервальных графов с двумя параметрами. Доказана теорема о представлении графа отрезками 3-мерного пространства [21].

Разработана технология и проектный подход к автоматизации проектирования, создания и сопровождения "клиент – серверных" прикладных информационно-вычислительных систем и систем автоматизированного проектирования, основанный на формализации описания проекта системы и автоматической генерации программного кода ("Генератор проектов") [22-25]. Цикл этих работ подробнее описан в разд. 2.

Поставлены и решены новые задачи распределения нескольких видов ресурсов на сетевых графиках. Разработан эффективный алгоритм построения расстояния Минковского между двумя выпуклыми непересекаю-

щимися многогранными множествами. Реализованы алгоритмы синтеза многопродуктовых сетей. Построены эффективные алгоритмы решения некоторых задач многократного минимакса [26-31].

С использованием проектного подхода при участии сотрудников отдела был разработан ряд автоматизированных информационных систем для финансовых приложений: автоматизированная банковская система "ГАМБИТ", система межбанковских платежей, система учёта и погашения взаимных задолженностей хозяйствующих субъектов и ряд других систем [22-23].

С использованием "Генератора проектов" разработана система "Электронный каталог деталей объектов машиностроения" для хранения и оперативного доступа к данным, которые описывают каталоги узлов, агрегатов и деталей разнообразных сложных технических объектов машиностроения.

С использованием "Генератора проектов" разработана новая версия автоматизированной системы весовых расчётов (АСВР-М) для обеспечения проектирования летательных аппаратов.

В стадии реализации находится проект создания комплексной системы инженерных расчётов (КПИР) для автоматизации проектирования летательных аппаратов на этапе эскизного проектирования.

Разработан проект многопроцессорного программного комплекса мониторинга, анализа и прогнозирования поведения многоагентных систем на базе нейрокомпьютерной модели мозжечка [32, 33].

Созданы математические модели для синтеза термодинамически эффективных схем реактивных двигателей [34-40]. Решены проблемы формирования критериев оценки и построения простейших моделей функционирования такого сложного технического объекта как воздушно-реактивный двигатель (ВРД). Исследованы проблемы выбора ВРД известных типов и проблема согласования простейших формализованных описаний эффективных вариантов ВРД с более подробными описаниями в системах автоматизированного проектирования.

Рассмотрена обобщённая схема основных типов ВРД. Поставлена и исследована двухкритериальная задача. Критериями оптимальности идеальных вариантов являются удельная тяга и удельный импульс по теплу. Установлены причины несовпадения отдельных результатов с результатами традиционной теории, даны неформальные оценки влияния факторов неидеальности реальных процессов на результаты сравнения.

Рассмотрены турбореактивные двигатели (ТРД) с регенерацией тепла и упрощённые варианты схемы известного трёхвального двухконтурного ТРД с промежуточной камерой сгорания. Найдены оптимальные варианты идеальных термореактивных систем при фиксированных локальных внешних условиях. Показано, что одноконтурные двигатели с дополнительной камерой сгорания между ступенями турбины экономичнее своих прототипов (ТРД, ТРДФ), что регенерация тепла даёт положительный эффект не при любых условиях.

В простейших представлениях описывается функционирование ВРД заданной конструкции при нерасчётных внешних условиях, возможности рассматриваемых законов управления сравниваются качественно и в численных примерах. Данные исследования служат примером того, что на нулевом уровне описания могут решаться некоторые вопросы регулирования двигателя.

Построены схемы ВРД, позволяющие политропически охлаждать динамически сжатый в воздухозаборнике воздух, отводить тепло в атмосферу, утилизировать рассеиваемое тепло сильно нагретыми элементами конструкции, а также позволяющие реализовать термодинамический цикл, масса рабочего тела которого больше массы газа, покидающего двигатель.

Исследована проблема построения реактивных двигателей периодического действия. Построены варианты поршневого газогенератора, являющиеся кардинальными модификациями двигателей внутреннего сгорания Отто и Дизеля. Рассмотрена проблема моделирования силовых установок для космических транспортных систем и синтеза их схем.

Исследована проблема построения реактивных двигателей периодического действия.

2. Автоматизация проектирования информационно-вычислительных систем

Одной из основных задач автоматизации проектирования как предметной области является разработка инструментальных программных средств, обеспечивающих повышение эффективности разработки проектов сложных объектов различного целевого назначения. Безусловно, специфика проектируемых объектов определяет перечень конкретных решаемых в САПР задач, а также её архитектуру, математическое, программное, информационное и техническое обеспечение. Традиционный класс объектов автоматизации проектирования обычно связывают со сложными техническими объектами в машиностроении, электронике, строительстве и т. п. В рамках этого класса объектов достаточно хорошо проработаны теоретические основы САПР, а также существует целая гамма конкретных реализаций систем автоматизированного проектирования. Вычислительный

центр РАН принимал активное участие в создании САПР летательных аппаратов [41-56]. С тех пор круг наших интересов в области автоматизации проектирования существенно расширился. Нами был получен ряд теоретических и практических результатов в области автоматизации проектирования информационных систем различного применения, в частности при разработке финансовых приложений [57-74]. Такое изменение предметной области должно было привести к полному изменению средств и методов автоматизации проектирования. Но оказалось, что многие проблемы синтеза сложных систем (а информационно-поисковые и информационно-расчётные финансовые системы очень сложны) инвариантны по отношению к предметной области.

Проектирование и разработка современных информационных систем в области финансовых приложений является очень дорогостоящим делом. Характерным примером сложных финансовых приложений являются автоматизированные банковские системы (АБС). Разработка АБС, их внедрение в работу банков, последующая доработка, доводка, сопровождение могут занимать не один год работы больших коллективов высококлассных специалистов. Закономерности современного этапа развития сферы финансовых услуг характеризуются очень высокими темпами роста объёма услуг, расширением спектра финансовых инструментов, усложнением применяемых технологий. Это влечёт за собой усложнение разрабатываемых финансовых информационных систем. Современные клиент-серверные системы с сотнями и тысячами удалённых рабочих мест обрабатывают в режиме реального времени миллионы транзакций в сутки. Практически возникла новая отрасль, связанная с разработкой финансовых информационных приложений.

Динамика потребности в области финансовых приложений и жёсткая конкуренция в этой сфере диктуют необходимость кардинального сокращения сроков их разработки. Для достижения этой цели разрабатывающие компании вынуждены привлекать большое число программистов. Однако рост числа разработчиков, увы, не приводит к сокращению сроков разработки. И если ещё сроки доведения системы до сдачи в эксплуатацию можно как-то регулировать директивным способом, то доводка до рабочего состояния сложных финансовых информационных систем растягивается на неопределённое время и приводит к существенным расходам разрабатывающих компаний.

Эта ситуация очень напоминает период бума научно-технической революции в области создания систем вооружения, когда сложность различных классов технических объектов бурно росла. При этом и сроки проектирования, и особенно сроки натурных испытаний, объём доработок, модификаций катастрофически увеличивались, несмотря на большой приток научных и технических кадров в эти области. Может быть, наша аналогия покажется натянутой, но если сравнить, например, самолёт с несколькими десятками тысяч деталей и современную банковскую программную систему с сотнями тысяч операторов, то станет очевидна, по крайней мере, сопоставимость проблемы роста энтропии при проектировании самолётов и при разработке сложных информационных систем. Энтропию проектов сложных разрабатываемых систем можно оценить как

$$H \approx -\sum (p_i \log p_i + (1 - p_i) \log(1 - p_i)),$$

где $i = 1, ..., N$, N – это характеристика сложности проекта системы (число деталей в проекте самолёта или число операторов в исходном программном коде АБС), а p_i – вероятность того, что в i-м элементе проекта (в чертеже детали или в тексте оператора) разработчик допустил ошибку. В применении к проектированию энтропия – это величина противоположная качеству, мера его неадекватности. Действительно, энтропия пропорциональна числу вопросов, которые необходимо "задать" созданной по этому проекту системе, чтобы определить её состояние, то есть выявить и устранить допущенные в процессе проектирования ошибки. Увеличение энтропии всегда очень дорого стоит. При проектировании самолётов это ведёт к увеличению объёма лётных испытаний. В применении к финансовым информационным системам увеличение энтропии означает увеличение затрат и времени на доведение системы до рабочего состояния, а зачастую и необходимость существенных доработок в процессе их функционирования. Доработки, модификации, обновление версий финансовых систем в процессе эксплуатации является очень неприятным, трудоёмким и дорогим последствием увеличения энтропии исходных проектов. Фирмам – разработчикам АБС и других аналогичных систем приходится держать специальные службы сопровождения, трудозатраты которых, как правило, в несколько раз превышают трудозатраты разработчиков проекта.

Из сказанного выше очевидно, что увеличение численности разработчиков проекта без изменения подхода к проектированию не может существенным образом повлиять на сроки и качество создания систем, а зачастую и ухудшает его, поскольку увеличивается вероятность ошибок при усложнении координации разработки проекта. Единственным выходом из этой ситуации является уменьшение сложности проекта и уменьшение вероятности ошибок проектирования. Только таким образом можно существенно уменьшить значение энтропии проекта и, следовательно, не только уменьшить трудоёмкость разработки, но и существенно сократить затраты сопровождения системы.

Уменьшение сложности проекта – это не уменьшение сложности разрабатываемой системы. Увы, сложность системы – это неизбежность развития современных финансовых технологий. Уменьшение сложности проекта – это упрощение формального описания создаваемого программного продукта, т. е. уменьшение того самого N – количества «операторов» исходного текста проекта системы. Мы заключили слово "оператор" в кавычки потому, что уж если говорить о формальном языке описания проекта, то это должен быть непроцедурный язык высокого уровня, в котором основными элементами являются не операторы, а описания математических моделей, содержательных понятий и объектов, связанных с предметной областью. Проект – это формальный документ или совокупность формальных документов, которые обладают определённой структурой, определённым составом своих компонент, правилами оформления, синтаксисом, семантикой и прочими атрибутами формального объекта. Существенное требование к языку описания проектов состоит в том, чтобы уровень описания и формализации проекта был бы достаточным для однозначного истолкования его на следующем этапе, на этапе создания программного кода.

Другим резервом уменьшения энтропии проекта является уменьшение вероятности возникновения ошибок. Самым эффективным способом реализации этой задачи в программных системах является автоматизация создания программного кода. Создание программного кода фактически является технологическим этапом реализации проекта, и если проект – это формальное описание разрабатываемой системы, то создание программного кода может быть и должно быть полностью автоматизировано. Автоматизация создания программного кода требует специального инструментария – генератора программного кода, который является технологической компонентой проекта. Жизнь проекта программной системы не заканчивается её реализацией. Длительность жизненного цикла программных систем определяется способностью к модификациям. Как уже говорилось выше, модификация работающей системы является весьма дорогим и сложным процессом. Если же при разработке системы используется автоматический генератор программного кода, то процедура модификации существенно упрощается за счёт того, что изменения вносятся только в проект, а все дальнейшие изменения в программном коде, информационном окружении и прочих компонентах системы осуществляются автоматически.

Однако универсальных систем генерации программного кода прикладных систем не существует. Если следовать уже проводимым ранее аналогиям с проектированием самолётов, то для их производства специально проектируется и создаётся оснастка – стапели, шаблоны, сборочный инструмент и прочее. Точно так же для больших информационных систем нужно иметь специальные инструментальные средства автоматизации, которые были бы неотъемлемой технологической компонентой проекта системы. Понятно, что такой проблемно ориентированный инструментарий не может и не должен обладать многими свойствами универсальных CASE-продуктов, но он должен решать те задачи автоматизации, без которых невозможно обойтись при разработке конкретных приложений.

Какими же свойствами должны обладать подобные "встраиваемые" в проект инструментальные средства? Ниже перечислены основные задачи автоматизации, которые приходится решать при проектировании информационных систем:

— создание единого информационного пространства разрабатываемого проекта;

— описание архитектуры проектируемой системы;

— структурное проектирование информационных систем;

— разработка математических моделей объектов предметной области;

— описание логических структур и моделей хранимых данных;

— описание серверных компонент системы;

— описание пользовательского интерфейса, механизмы редактирования пользовательских документов, форм, диалогов и прочее;

— создание опытных образцов, макетов, демонстрационных стендов;

— автоматизация программирования, генерация исходного программного кода; автоматизация разработки программной и эксплуатационной документации системы;

— автоматизация разработки программы, методики и тестов функциональных, ресурсных и других испытаний системы;

— автоматизация разработки средств загрузки и инсталляции системы, средств, обеспечивающих системное обслуживание и эксплуатацию.

В Вычислительном центре был разработан ряд программных продуктов с общим названием Генератор проектов, который в той или иной мере решает перечисленные задачи [67-74]. Ниже дано краткое описание технологии их решения с помощью Генератора проектов. При этом мы ориентируемся в основном на свой опыт разработки финансовых и других приложений.

Единое информационное пространство проекта необходимо для организации процесса разработки системы информации и обеспечения коллективного доступа к этой информации. В рамках единого информационного пространства хранятся все документы от технического задания на разработку и вплоть до исполняемых модулей и инсталляционных пакетов всех разработанных версий. Средствами инструментария необходимо поддерживать целостность хранимой информации, т.е. осуществлять контроль полноты и непротиворечивости данных, соответствие дат, версий и других атрибутов различных компонент разрабатываемого проекта. Создание единого информационного пространства проекта — это первый формальный шаг, с которого начинается разработка прикладной информационной системы.

Архитектура проектируемых систем — это один из главных вопросов, который нужно решить на самых первых шагах проектирования информационных систем. Под архитектурой понимается состав основных компонент информационной системы, и связи между ними. Существенной особенностью информационных систем, которые разрабатываются с помощью Генератора проектов является то, что они являются многопользовательскими клиент-серверными системами, работающими с большими объёмами хранимой информации.

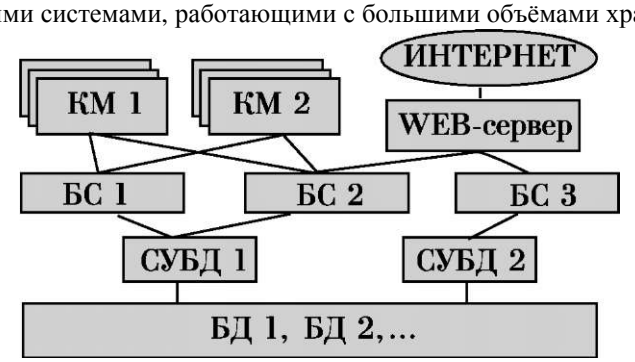

Рис. 1. *Клиент-серверная архитектура*

Основными информационными и программными компонентами таких систем, представленными на рис. 1, являются:

- БД – централизованные или распределённые базы данных и хранилища информации;
- СУБД – системы управления базами данных, которые организуют физический доступ к данным, организуют и обеспечивают надёжное их хранение, поиск и модификацию;
- БС – прикладные бизнес-серверы, программы, которые принимают запросы от пользователей системы на выполнение определённых бизнес-процедур, обрабатывают эти запросы, проводят необходимые вычисления, формируют необходимые логические запросы к базам данных, получают информацию из баз данных, формируют и отправляют ответы пользователям;
- КМ – клиентские модули-программы, которые обеспечивают формирование запросов пользователей на выполнение необходимых бизнес-процедур, передачу этих запрос на БС, приём ответов от БС и представление полученных результатов на терминальных устройствах;
- WEB-серверы – прикладные программы, которые обеспечивают санкционированный доступ к прикладной системе пользователям ИНТЕРНЕТ. WEB-серверы являются специализированными клиентскими модулями системы;
- программные и информационные средства, обеспечивающие связь между отдельными компонентами системы.

Математическое моделирование объектов предметной области. Это наиболее сложная и наиболее ответственная задача всего процесса проектирования прикладных систем. Не существует и не может существовать универсальных методов и средств математического моделирования произвольных систем, объектов и процессов. В то же время для конкретных классов задач существуют специализированные средства и языки моделирования предметной области. Именно это обстоятельство и позволяет предполагать, что наиболее эффективными системами автоматизации проектирования могут быть проблемно ориентированные CASE-средства. При разработке финансовых приложений необходимо уметь моделировать такие сущности, как:

— субъекты финансовых отношений и их организационные структуры;
— финансовые и платёжные инструменты;
— финансовые операции и технологии их выполнения;
— механизмы и технологии бухгалтерского учёта;
— аналитические инструменты финансовой деятельности.

Существует два подхода к реализации математических моделей, т.е. к использованию их в бизнес-процедурах. Первый подход связан с построением на базе этих моделей программ, реализующих конкретные задачи предметной области. Другой подход связан с механизмом интерпретации формального описания математических моделей в конкретных условиях решаемых задач. Выбор способа реализации математических моделей представляет собой один из аспектов разработки архитектуры системы.

Логические структуры данных формируются на основе разработанных для проектируемой системы математических моделей. Логические структуры используются при организации передачи и хранения данных. Характер использования хранимой информации определяет *модель хранимых данных.* В прикладных системах используются реляционные, иерархические и сетевые модели данных. Большинство промышленных баз данных в настоящее время ориентируется на реляционную модель, которая обладает рядом очевидных преимуществ — это простота реализации, простота пользовательского интерфейса, наличие удобного и простого языка запросов. Однако для задач, которые требуют организации сложных структур данных и разнообразных неоднородных процедур обработки, реляционная модель данных может быть недостаточно эффективна. В этом случае целесообразно выбрать сетевую модель. Иерархическая модель является частным случаем сетевой. Генератор проектов допускает работу и с реляционными и с сетевыми структурами данных.

Описание прикладных бизнес-серверов. В прикладной информационной системе может быть один или несколько бизнес-серверов. Прикладные серверы обеспечивают связь пользователей с базами данных и выполнение пользовательских запросов. Для каждого прикладного сервера описывается полный состав выполняемых им бизнес-процедур. Реализация бизнес-процедур основывается на этих описаниях, на описании запросов к базам банных и на описании математических моделей предметной области. Основанием для выполнения бизнес-процедур является пользовательский запрос, который представляет собой входной документ определённой

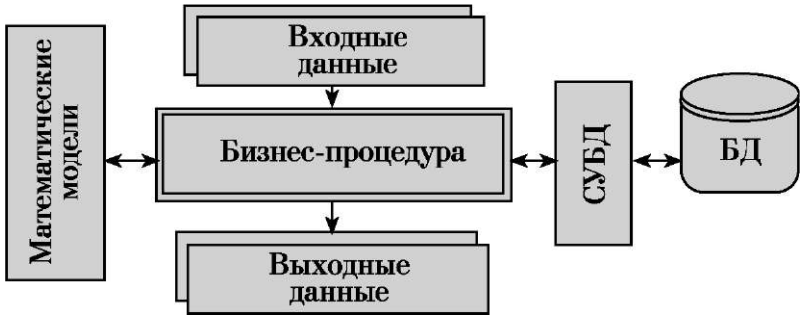

Рис. 2. *Схема выполнения бизнес-процедур прикладным сервером*

структуры или несколько связанных между собой документов. Результат выполнения пользовательского запроса также представляет собой один или несколько выходных документов. На рис. 2. показана общая схема выполнения любого пользовательского запроса.

Входные и выходные документы представляют определённым образом организованную, структурированную информацию. Структура документов определяется их типом. Типы используемых документов должны быть описаны на этапе построения логических структур данных. Бизнес-процедуры в процессе исполнения обращаются к математическим моделям, которые могут быть реализованы в виде библиотек программ или использоваться в режиме интерпретации.

Описание пользовательского интерфейса. Исполнение бизнес-процедур инициируется пользователями, которые посылают на серверные компоненты свои запросы с помощью клиентских модулей. Основными объектами описания клиентских модулей являются пользовательские запросы и сценарии (последовательность) их вызовов. Структура пользовательского запроса содержит ссылки на серверные бизнес-процедуры, ссылки на типы входных и выходных документов бизнес-процедур, способы формирования входных документов и формы интерпретации выходных. Фактически клиентские модули обеспечивают пользовательский интерфейс с системой. Других функций у клиентских модулей в рассматриваемой нами архитектуре нет. Пользовательский интерфейс определяется:

— составом пользовательских запросов к прикладным серверам,

— структурой пользовательских меню и команд на всех уровнях клиентских модулей,

— представлением входных и выходных документов в виде экранных форм, диалогов разного типа и твёрдых копий,

— сценариями формирования и анализа экранных документов,

— структурой диалогов для формирования документов,

— представлением входных и выходных документов в виде форматированных файлов.

Таким образом, если резюмировать всё сказанное выше, полноценный проект прикладной информационной системы должен содержать описание

— архитектуры системы,

— совокупности используемых математических моделей,

— логической структуры и моделей данных,

— прикладных серверов и реализуемых ими бизнес-процессов,

— клиентских модулей и реализуемых ими пользовательских запросов к прикладным серверам.

Для представления всех этих компонент проекта должны быть специальные средства, специальный язык описания проектов.

Язык описания проектов. Проект программного комплекса или системы представляет собой набор файлов, в которых содержится описание содержательных понятий и объектов разрабатываемой системы, логических структур данных, бизнес-процедур системы и пользовательского интерфейса. Основными объектами языка описания проекта являются:

— реквизиты проекта,

— платформы, для которых предполагается генерация программ,

— пользователи системы,

— типы данных,

— документы,

— сетевые структуры данных,

— реляционные базы данных,

— автоматически генерируемые SQL - запросы,

— произвольные SQL - запросы,

— прикладные серверы,

— порты прикладных запросов,

— WEB - порты серверов,

— бизнес-процедуры прикладных серверов,

— "ручные" программы бизнес-процедур,

— пользовательские окна,

— пользовательские диалоги,

— пользовательские приложения.

Реквизиты проекта задают имя проекта и другие выходные данные, связанные с разработкой проекта:

project <идентификатор проекта>:
"<Полное наименование проекта>";
/version = <номер версии>
/copyright = "<информация о copyright>"
/author = <информация об авторах описания проекта>
/baseport = <начальный порт TCP/IP по умолчанию>
/ language = <языки>

Под **платформой** в данном контексте подразумевается аппаратная платформа (Intel, VAX, Alpha, Power PC, ...), операционная система (MS Windows, Linux, Free BSD, Open VMS), система управления базами данных (MS SQL Server, Oracle, Sybase, MySql). С точки зрения пользовательского интерфейса платформа может быть основана на использовании Win32 GDI, Motif, Gtk и пр. С точки зрения web-интерфейса можно использовать web-сервера Apache, Microsoft IIS. В Генераторе проектов процесс генерации предусматривает использование разных платформ в рамках одного проекта. Тексты программ для каждой платформы могут быть сгенерированы независимо в разное время.

Пользователи генерируемой системы подразделяются на группы. Каждой группе пользователей присваиваются определённые права и функции, которые доступны пользователям данной группы. Структуры, связанные с пользователями системы, описаны в статье [74].

Типы данных в описании проекта используются в различных контекстах. Тип данных может задаваться как:

— предописанный тип данных (numb, char, date, money,...),

— перечислимый тип (enum, radio, mask),

— структура (struct).

Каждое описание типа вводит уникальный идентификатор типа, программное представление, компоненты и ряд опций.

type <имя> : <программное представление>

[(<*компоненты*>)] {<*опции*>...}

Понятие документа является одним из центральных в описании проекта. Документ – это описание типа данных, имеющего сложную внутреннюю структуру:

document <имя документа> [: <*имя структурного типа*>];

{ *record <имя записи>* [: < *имя структурного типа* >];}...

{ *set* **<***имя***>**[*owner <владелец>*] *member <член набора>*;}...

Имена документов уникальны в пределах проекта. В документе может быть декларировано произвольное количество именованных записей (record). Имена записей уникальны в пределах документа. С записью может быть связан структурный тип. Кроме того, в документе может быть декларировано произвольное количество именованных наборов (set). Имена наборов уникальны в пределах документа. Набор описывает связь между записями документа типа "один ко многим". Набор может быть последовательным и ключевым. В последнем случае задаётся ключ в виде последовательности компонент записи – члена набора, по которым в лексикографическом порядке упорядочиваются члены набора автоматически при включении в набор. В последовательном наборе порядок записей задаётся при их включении указанием номера позиции. Можно также описывать, так называемые сингулярные наборы, у которых отсутствует запись <владелец>. Каждый сингулярный набор в документе представлен ровно одним экземпляром. В некотором смысле владельцем сингулярного набора можно рассматривать сам документ, вернее, экземпляр документа для экземпляра набора.

Описание документа автоматически предполагает наличие в языке описания проекта операторов манипулирования содержимым документа. К таким операторам, в частности, можно отнести следующие действия:

— записать в заданный экземпляр документа структуру;

— считать из заданного документа структуру;

— создать экземпляр записи данного типа с указанием структуры, содержимое которой нужно разместить в записи;

— удалить заданный экземпляр записи;

— считать из заданного экземпляра записи структуру;

— записать в заданный экземпляр записи структуру;

— включить заданный экземпляр записи в экземпляр набора в заданную позицию;

— найти по заданному номеру позиции экземпляр члена набора по экземпляру владельца;

— перейти от заданного экземпляра члена набора к следующему/предыдущему;

— найти экземпляр владельца по заданному экземпляру члена набора;

— для заданного владельца ключевого набора и значения ключа найти соответствующий экземпляр члена набора;

Сетевые структуры данных. Модель документа есть не что иное, как хорошо забытая (и совершенно напрасно) модель сетевой базы данных по спецификации КОДАСИЛ. Похожесть на модель реляционной базы данных не в счёт, так как там, во-первых, primary/foreign key реализуются в виде ограничений целостности, проверяемых во время модификации, а не в виде физических ссылок. Во-вторых, работа с документом предполагает использование операторов перечисленного выше вида, а не использование SQL-запросов (может быть, в будущих версиях Генератора и это будет реализовано, если где-нибудь понадобится).

Структуры реляционных баз данных в проекте описываются в виде совокупности таблиц, индексов и связей между таблицами, которые определяются, как и для сетевых данных, с помощью задания наборов (set):

{ *table <имя таблицы>* : *<имя структурного типа>*

[(<*первичный ключ таблицы*>)];

[<*опции таблицы*>]}...

{ *index* <*имя индекса*> *on* <*имя таблицы*> [*unique*]{<*индекс*>)}...

{ *set* <*имя набора*> *owner* <*имя таблицы-владельца*> *member*

<*имя таблицы — членов набора*> (<*ссылка на владельца*>);}...

Столбцы (поля) таблицы соответствуют компонентам структурного типа, описывающего таблицу. Первичный ключ и индекс – это списки имён столбцов таблиц. Первичный ключ должен быть заявлен в списке уникальных индексов. В описании набора ссылка на владельца – это список имён столбцов таблиц – членов набора. Ссылка на владельца в наборе должна по типам данных совпадать с первичным ключом владельца набора.

Автоматически генерируемые SQL-запросы распространяются только на одну таблицу базы данных. Список автоматически генерируемых запросов задаётся в опциях к таблице:

[*select* <*имя запроса*>(<*вход* >):(<*выход* >);]...

[*insert* < *имя запроса* > (<*вход* >);]...

[*update* < *имя запроса* > (<*вход*>):(<*изменяемые поля*>);]...

[*delete* < *имя запроса* > (<*вход* >);]...

[*cursor* < *имя запроса* > {<*вход*>):(<*выход*>)/(<*порядок*>);]...

Здесь <*вход*> – это список полей, которые используются в предикате поиска (WHERE), <*выход*> – выходные переменные запроса.

Произвольные SQL-запросы, которые могут быть описаны в проекте, представляются так же, как и автоматически генерируемые запросы, и реализуются в виде функций с входными и выходными параметрами. Для описания этих запросов используется стандартный SQL. Описание произвольных SQL-запросов в проекте имеет следующий вид:

sql <*имя запроса*> (<*вход*>):[(<*выход*>)]

<*тело запроса на SQL*>;

В теле запроса входные переменные *(<вход>)* могут явно использоваться как host-переменные в Embedded SQL для C. Выходные переменные (<*выход*>) имеют смысл только для поисковых запросов (*select & cursor for select*).

Прикладные серверы реализуют бизнес-процедуры. С каждым сервером может быть связана одна или несколько баз данных реляционного или сетевого типа. На рис. 3. приведён пример серверной архитектуры и связи серверов с другими компонентами системы.

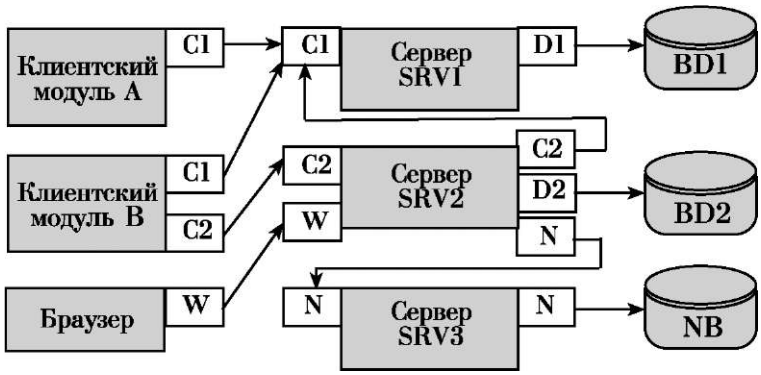

Рис. 3. *Пример клиент-серверной архитектуры*

Сервер может иметь один или несколько **портов,** которые предназначены для связи с разными типами клиентских модулей. Через каждый порт сервера для пользователей доступны только те бизнес-процедуры этого сервера, которые приписаны к данному порту. Сервер может иметь несколько специализированных портов для исполнения запросов к сетевой базе данных. Сервер может иметь несколько портов, работающих по протоколу HTTP, для реализации WEB-интерфейса пользователей. И, наконец, сервер может иметь несколько

клиентских интерфейсов для связи с другими серверами, по отношению к которым данный сервер выполняет роль клиента. На рис. 3

— сервер SRV1 имеет порт C1 для приёма пользовательских запросов и интерфейс D1 с реляционной базой данных BD1;

— сервер SRV2 имеет порт C2 для приёма пользовательских запросов, порт W для приёма HTTP-запросов, клиентское соединение C1 с портом C1 сервера SRV1, интерфейс D2 с реляционной базой данных BD2, клиентское соединение N с портом N сервера SRV3 для запросов к сетевой базе данных;

— сервер SRV3 имеет порт N для приёма запросов к сетевой базе данных и интерфейс ND к сетевой базе данных NBD;

— клиентский модуль A имеет интерфейс C1 с портом C1 сервера SRV1 для передачи пользовательских запросов;

— клиентский модуль B имеет интерфейс C1 с портом C1 сервера SRV1 для передачи пользовательских запросов, интерфейс C2 с портом C2 сервера SRV2 для передачи пользовательских запросов;

— стандартный браузер имеет возможность передать запрос Интернет-порту W сервера SRV2.

Интернет-порт является специальным видом серверного порта, предназначенного для обеспечения санкционированного доступа к бизнес-процедурам серверов системы через Интернет.

Описание прикладного сервера имеет следующий вид:

server <*имя сервера*>;

[*database* <*имя базы данных*>:<*тип реляционной базы данных*>]...

[*genbd* <*имя базы данных*>:<*тип реляционной базы данных*>]...

{*port* <*имя порта запросов*>}...

{*gbdport* <*имя порта сетевой базы*>}...

{*client* <*имя соединения*> *port* <*имя порта*>}...

{*web* <*имя web-порта*>}...

{<*описание функций*>}...

{<*описание бизнес-процедур*>}...

Функции, которые описаны в пределах прикладного сервера, носят служебный характер и используются при описании бизнес-процедур.

Бизнес-процедуры прикладных серверов – это функции сервера, подключённые к определённому порту:

request <*имя порта*>.<*имя бизнес-процедуры*>

input (<*входные документы*>)

output (<*выходные документы*>)

func (<*тело бизнес-процедуры*>)

Бизнес-процедуры принимают входные документы от пользователей, обрабатывают их, формируют ответ в виде выходных документов и передают их обратно пользователям. В теле функции бизнес-процедуры могут использоваться любые описанные выше функции, а также функции SQL-запросов используемых реляционных баз данных и операторы работы с сетевыми базами данных.

Пользовательские окна — это абстрактное понятие, связанное с внешним представлением описанных в проекте документов. В настоящее время в системе имеются предописанные оконные классы текстовых окон, таблиц и деревьев. На основе этих классов разработчик может описать именованные оконные типы, задав имена и типы переменных/документов в составе окна, а также задав программы, описывающие поведение окна в разных ситуациях.

window <*имя окна*> *document* <*имя документа*>

(<*входные переменные*>)

:(<*выходные переменные*>)

{*tableview* | *textview* | *treeview*}

[<*описание компонент окна, а также управляющих команд* >]

Пользовательские диалоги представляют собой специальный вид окна, открытие которого блокирует доступ к окну приложения до тех пор, пока пользователь не закроет это окно.

dialog <имя диалога> document <имя документа>
(*<входные переменные>*)
:(*<выходные переменные>*)
[*<описание компонент и команд диалога>*]

Диалоги используются для ввода данных и для просмотра выводимой в диалог информации. В качестве компонент окна можно использовать разнообразные типовые элементы вроде окон ввода текста, выпадающих списков, птичек и пр. стандартных элементов. Кроме того, в окне можно использовать в качестве своих элементов описанные ранее окна. Размещение элементов окна управляется многочисленными опциями и специальными контейнерными элементами типа вертикальная коробочка, горизонтальная коробочка и пр.

Пользовательские приложения — это клиентские модули, обеспечивающие интерфейс пользователей с бизнес-процедурами серверов. Для приложения может быть объявлено, что оно является клиентом для заданного списка серверов.

application <имя пользовательского приложения>,<опции>
client <имя клиента> port <имя серверного порта>

С точки зрения пользовательского интерфейса в приложении декларируется список разных оконных видов – макетов, а также управляющие элементы для каждого макета.

layout <имя макета размещения окон> document <имя документа>

(*<входные переменные>*):(*<выходные переменные>*)

<геометрия размещения окон>

<описание управляющих команд и других свойств макета>

Макет определяет состав и взаимное расположение описанных выше окон внутри главного окна приложения. Управляющие команды определяют схему активизации окон макета, переход от макета к макету, вызов меню и диалогов, обращение к бизнес-процедурам сервера и тому подобное.

Кроме описанных выше основных понятий и объектов в Генераторе проектов есть ряд предописанных функций и макросов, предназначенных для описания проекта.

Все упомянутые выше понятия и объекты проекта упорядочены и представлены в различных файлах, совокупность которых составляет полное описание проекта.

Программа Генератора проектов реализована в виде консольного приложения для платформ Win32 и Linux. Для генерации проекта необходимо запустить Генератор в рабочем каталоге описания проекта и указать в качестве параметра идентификатор проекта, а также набор опций для указания требуемых платформ. Для каждой указанной платформы генерируется программный код системы.

Вместе с текстами программ для каждой платформы генерируются скрипты для сборки программ для получения дистрибутивов системы.

С точки зрения состава программных компонент в дистрибутиве могут присутствовать программы следующего вида:

— серверы-программы, обрабатывающие запросы от клиентских программ на выполнение бизнес-процедур;

— клиентские модули с оконным интерфейсом;

— библиотеки программ, обеспечивающие выполнение бизнес-процедур;

— вспомогательные программы — конфигураторы баз данных, CGI-интерфейсы и пр.

Эффективность всякой автоматизации, как правило, измеряется коэффициентом повышения производительности труда. Разумеется, и при применении нашего проектного подхода к разработке прикладных информационных систем хотелось бы иметь такие оценки. Мы могли бы, конечно, привести цифры, тем более, что это не трудно сделать, сравнив объём программного кода и объём проекта системы. В частности, для рассмотренного выше примера такое соотношение составит величину второго порядка, т. е. объём автоматически сгенерированного программного кода (сумма объёмов только * .c * .h файлов ~0,7 Мгб) более чем в сто раз превышает суммарный объём исходных файлов проекта (~5 Кбт). Для крупных проектов это соотношение будет поменьше, но всё равно оно достигает величины как минимум порядка десятка. Ниже дано краткое описание тех прикладных проектов, которые были разработаны в разные годы средствами Генератора проектов.

1992 *г.*: *Система обработки почтовых и телеграфных авизо* (*Заказчик ГУ ЦБ РФ по г. Москва*). Автоматизированная система обработки почтовых и телеграфных авизо была внедрена и эксплуатировалась в течение нескольких лет в Главном расчётно-кассовом центре ГУ ЦБ РФ по г. Москве и Шаболовском РКЦ.

1993 *г.*: *Система приёма и обработки электронных межбанковских платежей "АС МБР"* (*Заказчик: ГУ ЦБ РФ по г. Москва*). Система обработки межбанковских платежей в режиме реального времени была внедрена и эксплуатировалась в течение нескольких лет в Главном расчётно-кассовом центре ГУ ЦБ РФ по г. Москве и Шаболовском РКЦ.

1996 – 1999 гг.: *Система автоматизации банковской деятельности коммерческих банков и Учреждений Сбербанка РФ.* Система комплексной автоматизации многофилиального универсального коммерческого банка (ГАМБИТ) разработана, внедрена и эксплуатируется в Башкирском банке Сбербанка России. Система автоматизирует деятельность главной бухгалтерии банка, ОПЕРУ, территориального расчётного центра, отделений банка.

1998 – 1999 гг.: *Система урегулирования взаимной задолженности субъектов экономической деятельности РПБ КЛИРИНГ* (*Заказчик: КБ "Роспромбанк"*). Автоматизированная система представляет собой программно-технологический комплекс, предназначенный для проведения работ по взаимному многостороннему урегулированию просроченной задолженности субъектов экономической деятельности на региональном и федеральном уровнях.

1999 г.: *Пилот-проект "Electronic Transcribing System".* (*Заказчик: "BrainStorm Engineering, LLC", США.*) Прототип многопользовательской автоматизированной системы для распределённой обработки медицинской информации через Internet.

1998 – 2000 гг.: *Проект eCommerce.* (*Заказчик: Сбербанк РФ.*) Был разработан и сертифицирован проект электронной коммерции. Разработана система интернет-платежей, которая позволяет производить оплату товаров и услуг в интернет-магазинах в режиме реального времени с помощью микропроцессорных карт СБЕРКАРТ Сбербанка России.

2002 – 2003 гг.: *Проект системы Duplet.* (*Заказчик: ООО "Про-Кард"*). Разработан и внедрён проект автоматизированной системы DUPLET, предназначенной для обеспечения интернет-торговли и проведения некоторых банковских операций в режиме реального времени с микропроцессорными картами стандарта DUET (карты СБЕРКАРТ Сбербанка России), эмитированных платёжными системами с разными генеральными ключами.

2002 – 2003 гг.: *Проект системы мобильного банкинга.* (*Разработчик системы: ЗАО "СмартКарт Сервис"*). Система мобильного банкинга предназначена для предоставления клиентам коммерческих банков услуг безопасного управления банковским счётом с использованием личного мобильного телефона. Для обмена данными между банком и клиентом используется служба коротких сообщений (SMS).

Это неполный перечень тех прикладных информационных систем, которые были реализованы с помощью Генератора проектов. Безусловно Генератор обеспечивает очень высокую производительность для разработчиков прикладных систем. Однако даже не в конкретных цифрах увеличения производительности труда разработчиков состоит основной эффект от применения Генератора проекта. Главное преимущество проектного подхода состоит в том, что он не перемешивает две области деятельности – содержательное моделирование прикладных задач и технологические проблемы программирования. Благодаря этому появляется возможность достаточно глубокой формализации на этапе постановки задачи, т. е. в процессе проектирования системы. С другой стороны средства проектирования, т. е. правила описания системы, их форма и содержание, однозначно соответствуют доступной в данный момент технологии создания программ, которая заложена в Генераторе проекта и той структурной модели, на которую опирается Генератор. Развитие структурной модели, расширение возможностей автоматической генерации программ будут совершенствовать и средства проектирования. В связи с этим нам представляется, что дальнейшее углубление проектного подхода, разработка более мощных языковых средств описания содержательных объектов, моделей и процедур, может быть даже их специализация и ориентация на конкретные предметные области, являются весьма актуальными задачами для разработчиков прикладных информационных систем.

3. Автоматизации проектирования вычислительных систем реального времени

Вычислительные системы реального времени предназначены для контроля и управления в режиме реального времени процессами в автоматизированных производствах, транспортных системах, системах производства энергии, в нефте- и газодобывающих производствах, при разработке и испытаниях сложных технических объектов, многих других областях человеческой деятельности. В **секторе проектирования систем реального**

времени ВЦ РАН разработана методология создания системы автоматизации программирования вычислительных систем реального времени (САПР ВСРВ) [75, 76]. Система предназначена для автоматизации проектирования и генерации систем реального времени, осуществляющих обработку циклически поступающей информации в темпе поступления при жёстких временных ограничениях. Эта система, во-первых, позволяет в короткие сроки в автоматическом режиме проектировать из готовых прикладных модулей конкретную ВСРВ, описанную пользователем с помощью программы реального времени (РВ-программы), и, во-вторых, заранее, т.е. до обработки информации в реальном времени, проводить оптимизацию вычислений. Для реализации такого подхода необходимо предварительно построить математическую модель ВСРВ. С помощью этой модели могут быть решены такие задачи, как составление допустимого расписания выполнения процессов, синхронизация вычислений, динамическое распределение памяти. При разработке этой системы был получен ряд вспомогательных результатов, которые представляют отдельный теоретический и практический интерес. Например, были созданы и реализованы наглядные языковые средства для описания циклической обработки информации, разработаны различные алгоритмы дискретной оптимизации для ВСРВ, которые составляют основу системы.

3.1. Алгоритмы дискретной оптимизации для многопроцессорных ВСРВ. Предлагается пакет алгоритмов, предназначенных для планирования работ в многопроцессорных системах реального времени. В общем виде задача может быть сформулирована следующим образом. Имеется множество вычислительных заданий (работ) $N = \{1, ..., n\}$ и m процессоров r типов. Для каждого задания $i \in N$ известны следующие характеристики: $[b_i, f_i]$ – директивный интервал (выполнение задания i может быть начато не ранее момента b_i и должно быть завершено не позднее момента f) Q_i – объём работы процессоров, необходимый для полного выполнения задания i. Для каждого процессора задана производительность s_j. Если задание i выполняется процессором j в течение интервалов времени суммарной длины τ_{ij}, то $Q_i \sum_1^m s_j \tau_{ij}$. В случае, когда производительности всех процессоров одинаковые, вместо величин Q_i задаются длительности t_i выполнения заданий. В фиксированный момент времени каждое задание может выполняться только одним процессором, а каждый процессор может выполнять не более одного задания. В случае, когда допускаются прерывания и переключения, предполагается, что прерывание выполнения работы на процессоре j с целью её переключения на одном из последующих тактов на другой процессор или для возобновления на этом же процессоре может потребовать некоторых временных затрат системы. Задан граф связей между процессорами, отражающий возможность переключения работы с одного процессора на другой. Требуется определить, существует ли допустимое расписание выполнения заданий (когда каждое задание полностью выполняется в своём директивном интервале), и если оно существует, то найти его. Сформулированная задача является *NP*-полной. Поэтому нами были рассмотрены различные частные случаи этой задачи и, кроме того, разработаны эвристические алгоритмы.

3.1.1. Построение расписаний с прерываниями. В случае, когда граф связей между процессорами полный, допускаются прерывания и переключения и они не сопряжены с временными затратами, для поиска допустимых расписаний можно воспользоваться полиномиальными алгоритмами, описанными в работах [77-79]. Однако ограничения, связанные со структурой используемой ЭВМ, и специфика каждого конкретного случая требуют рассмотрения следующих задач.

Цикличность поступления требований на выполнение заданий. Требования на выполнение задания $i \in N$ поступают циклически с периодом p_i, начиная с момента времени p_{0i}, (т.е. k-е требование ($k = 1, 2, ...$) на выполнение задания i поступает в момент времени $p_{0i} + (k - 1)p_i$). Для k-го требования на выполнение задания i установлен директивный срок $p_{0i} + (k - 1)p_i + f_i$. Для решения этой задачи разработан полиномиальный алгоритм, основанный на сведении её к задаче поиска максимального потока в сети специального вида [80]. Алгоритм строит периодическое допустимое расписание с периодом P, равным наименьшему общему кратному величин $p_1, ..., p_n$. Вычислительная сложность алгоритма есть $O((rP\sum_{i=1}^{n} 1/p_i)^3)$.

Учёт ограничений на структуру связей между процессорами. Для случая, когда граф связей между процессорами — произвольный связный граф, временные издержки при прерывании выполнения работ и их переключениях с одного процессора на другой не учитываются, все процессоры идентичные, разработан полиномиальный алгоритм нахождения допустимого расписания [81]. Алгоритм основан на преобразовании с помощью ряда новых алгоритмов расписания для полного графа в расписание для заданного неполного графа связей между процессорами. При этом предполагается, что НОД ($b_1, b_2, ..., b_n, f_1, f_2, ..., f_n, t_1, t_2, ..., t_n$) $\geq 2m$ (НОД – наибольший общий делитель). Доказано, что если указанное условие не выполняется, то задача является *NP*-полной в сильном смысле. Вычислительная сложность алгоритма есть $O(n^4m^2 + n^3 \ln T)$.

Рассмотрен также случай, когда учитываются издержки на обработку прерываний и переключений. Предполагается, что при каждом прерывании или переключении некоторой работы с процессора на процессор системой дополнительно выполняется некоторое число тактов для записи текущего состояния работы, а при возобновлении – столько же тактов на чтение. Доказано, что в этом случае сформулированная задача становится *NP*-полной в сильном смысле даже для однопроцессорной системы, когда все работы имеют общий директивный срок. Для решения задачи разработан метод последовательных приближений, основанный на последовательной модификации потоковой сети, используемой в алгоритме, описанном в [77].

Рассмотрен также частный случай, когда все работы имеют общий директивный интервал, издержки на прерывания не учитываются, но при первой загрузке каждой работы, а также при переключении на процессор, на котором эта работа ещё не выполнялась, требуется процессорное время на загрузку данных. Эта задача сведена к многопродуктовой потоковой задаче в сети специального вида. Для решения этой задачи также разработан и другой подход, который основан на её сведении к задаче целочисленного линейного программирования, но с существенно меньшим числом переменных и ограничений.

Эвристические алгоритмы. Для случая, когда прерывания и переключения не требуют временных издержек, один из наиболее эффективных известных точных алгоритмов находит решение, выполняя $O(r^3u^3)$ операций [77-79]. При этом в построенном расписании не более $2(u^2 + 2mn - 3n - m + 1)$ прерываний. Несмотря на то, что этот алгоритм полиномиальный, время его работы часто является недопустимо большим при использовании в ВСРВ. Поэтому для решения сформулированной задачи разработаны два эвристических алгоритма [82]. Оба эти алгоритма были разработаны как варианты обобщения однопроцессорного алгоритма Коффмана [83] на случай, когда имеется несколько процессоров с различными производительностями: готовой к выполнению работы с меньшим директивным сроком назначается больший приоритет. В первом алгоритме на освободившийся процессор назначается работа с наивысшим приоритетом. Во втором алгоритме при завершении одной из работ все процессоры освобождаются и назначается m работ так, что работе с большим приоритетом предоставляется более быстрый процессор. Вычислительная сложность алгоритмов есть $O(mn)$ и $O(n^2 \log_2(n))$, а число прерываний в находимых ими расписаниях не более $n - 1$ и $2m(n - 1)$ соответственно. Как показали многочисленные машинные эксперименты (было проведено свыше 2500 расчётов), время работы обоих алгоритмов в сотни, а в некоторых случаях в тысячи раз меньше времени работы точного алгоритма. Эти эксперименты показали также, что уровень корректной работы первого алгоритма — порядка 80% второго — порядка 95%. Иными совами, второй алгоритм только в 5% случаях выдавал сигнал об отсутствии допустимого расписания, в то время как оно на самом деле существовало. Поэтому, несмотря на то, что второй алгоритм требует несколько больших вычислительных затрат, именно он может быть рекомендован для практического применения.

Разработаны также модификации обоих алгоритмов для случая, когда каждое прерывание требует определённого количества процессорного времени. При этом вычислительная сложность обоих алгоритмов остаётся прежней. Однако из-за существенно большего количества прерываний в расписаниях, которые генерирует второй алгоритм по сравнению с первым, в этом случае не имеет места однозначная предпочтительность применения второго эвристического алгоритма перед первым. Поэтому был исследован вопрос, начиная с какого соотношения характерного интервала между соседними директивными сроками к некоторой величине, характеризующей среднее время, затрачиваемое на прерывание, первый алгоритм становится более предпочтительным, чем второй. Были исследованы три различные модели образования временных затрат на обработку процессорами прерываний и переключений: 1) затраты равны некоторым постоянным величинам; 2) затраты обратно пропорциональны производительности процессора; 3) затраты имеют как постоянную, так и переменную, зависящую от процессора, составляющие. Для каждой из моделей экспериментальным путём определяется искомое соотношение (зависящее от параметров модели), начиная с которого первый алгоритм становится более предпочтительным для практического применения, чем второй.

Учёт ограничений на объём памяти. Дополнительно учитываются временные издержки на обработку прерываний и переключений, а также ограничения на объём памяти процессоров. Для каждой работы предполагается заданным объём памяти процессора, необходимый для её выполнения. Известен также объём памяти каждого процессора. Указанная задача сведена к многопродуктовой потоковой задаче в сети специального вида. Для однопроцессорного случая разработан полиномиальный алгоритм, вычислительная сложность которого есть $O(n^2 \log n)$ [84].

Задача синтеза. При предположении цикличности поступления требований на выполнения заданий определяется множество $S = \{(s_i,...,s_m)\}$ производительностей процессоров, при которых существует допустимое расписание. Показано, что множество S задаётся системой кусочно-линейных неравенств.

3.2. Построение расписаний без прерываний.

3.2.1. Применение агрегирования в задаче составления оптимального расписания без прерываний. Рассматривается многопроцессорная система, состоящая из m идентичных процессоров. Заданы длительности выполнения работ. Требуется построить оптимальное по быстродействию расписание выполнения работ без прерываний. Известно, что данная задача является *NP*-полной. Поэтому при её решении чрезвычайно актуальной становится задача понижения размерности, состоящая в разбиении исходного множества работ на несколько подмножеств меньшего размера. Разработаны алгоритмы, состоящие в построении дерева агрегирования и решения в каждом из узлов подзадачи упорядочения [82]. Система подзадач строится путём декомпозиции исходного набора заданий на систему наборов меньшей размерности. После решения подзадач проводится агрегирование набора заданий. Другими словами, этот подход содержит следующие основные элементы: последовательное разбиение задачи на подзадачи упорядочения существенно меньшей размерности, формирование дерева подзадач, решение подзадач, формирование решения задачи из решения подзадач в соответствии с построенным деревом подзадач.

Проведённые многочисленные машинные эксперименты показали высокую эффективность алгоритма. Так, при решении задач составления оптимального расписания для 1000 работ на 4 процессорах время работы алгоритма на персональном компьютере средней мощности составляло несколько секунд. При этом отклонение полученных решений от нижней оценки расписания не превышало 5 %.

Рассмотрен случай, когда длительности выполнения работ образуют арифметическую прогрессию или последовательность, близкую к арифметической прогрессии. Для такой задачи предложен эффективный приближённый алгоритм и получены формулы для длины расписания и погрешности метода. Полученные результаты многочисленных машинных экспериментов показали, что предложенные методы в большинстве случаев дают решения, отличающиеся по качеству от оптимальных на несколько процентов.

Для практического применения метода агрегирования при решении подзадач меньшей размерности и композиции решения исходной задачи из решений подзадач были разработаны следующие алгоритмы и программы.

Эвристический метод. В качестве эвристики выбран следующий алгоритм. На каждом шаге выбирается задача с наибольшей длительностью и назначается на процессор, для которого суммарное время выполнения уже назначенных задач наименьшее. Вычислительная сложность алгоритма есть $O(n \log m)$.

Переборный метод. В этом методе либо выполняется полный перебор, либо применяется метод "ветвей и границ".

Декомпозиция. Разбиение задачи на подзадачи производится по минимуму расстояния (разности времён выполнения).

Комбинированный метод. Процедура применения этого метода следующая. В каждой из подзадач меньшей размерности упорядочение выполняется путём перебора. Решение же агрегированной задачи строится с помощью эвристического алгоритма.

Многоуровневый метод. Каждая из подзадач, в свою очередь, также разбивается на более мелкие задачи. Этот процесс может повторяться несколько раз, что определяет глубину агрегирования.

Одинаковые директивные интервалы. Для задачи поиска допустимого расписания, когда каждая работа имеет один и тот же директивный интервал [O, T], разработан псевдополиномиальный алгоритм. Вычислительная сложность алгоритма есть $O(T^m)$.

Результаты машинных экспериментов продемонстрировали жизнеспособность предложенных методов. При достаточно малом времени работы рассмотренные алгоритмы показали малое отличие от нижней оценки длины расписания. Это позволит использовать предложенные методы в современных детерминированных системах при решении задач большой размерности. Разработанные методы были также обобщены и реализованы для случая, когда процессоры могут иметь различные производительности.

Алгоритмы организации рестартов. При функционировании вычислительных систем реального времени важное значение имеет подсистема организации рестартов, позволяющая в случае возникновения сбоев определить те программные модули, которые подлежат повторному запуску. Контроль поступающих в систему данных, а также данных, которыми обмениваются программные модули, осуществляется специальными модулями контроля, каждый из которых определяет, принадлежит ли множество значений контролируемых им параметров заданным пределам. Кроме того, в вычислительную систему реального времени вводятся дополнительные модули-буфера, сохраняющие промежуточную информацию для того, чтобы при рестарте системы воспользоваться данными из этих буферов, а не проводить все вычисления заново. В известной ранее работе

[85] строится минимальная зона рестарта и предлагаются схемы обновления расписания работы программы реального времени. Нами решена задача нахождения такого расположения модулей контроля и модулей-буферов, при котором математическое ожидание суммарного времени, затраченного на выполнение всех модулей (включая повторное их выполнение при возникновении ошибок), минимально [86]. При этом для каждого прикладного модуля предполагается заданной вероятность возникновения в нём ошибки. Рассмотрены различные случаи графа частичного порядка между модулями (цепочка, несколько параллельных цепочек, произвольный граф).

Наличие неопределённых факторов. Предполагается, что все работы имеют общий директивный интервал [*O, T*] и что в некоторые неопределённые моменты времени могут поступать запросы на выполнение более приоритетных работ N_1. Если при этом выполняются работы из *N,* то их выполнение прекращается и откладывается на более позднее время. Тем самым нарушается построенное ранее расписание для *N*. Процесс выполнения работ *N* и поступления запросов на выполнение более приоритетных работ N_1 повторяется многократно. Задача заключается в выработке такой стратегии построения расписаний выполнения работ *N*, при которой время выполнения совокупности работ *N* не превосходит *T* (если запросы на выполнение более приоритетных работ N_1 не поступают) и вероятность того, что расписание выполнения работ *N* будет нарушено вследствие поступления запросов на выполнение работ N_1, минимальна. Данная задача сводится к решению антагонистической игры специального вида. Разработан приближённый метод решения этой игры, основанный на аппроксимации её конечными играми [87].

3.3. САПР вычислительных систем реального времени (ВСРВ)

В настоящем разделе даётся краткое описание инструментальной системы автоматизации программирования ВСРВ, разработанной под руководством Б.Г. Сушкова. Рассматриваются основные вопросы методологии разработки системы и её реализации для персональных компьютеров.

3.3.1. Задачи САПР ВСРВ

Для генерации прикладной ВСРВ пользователю необходимо иметь прикладные модули, написанные на языках программирования, например Си, Фортран, Паскаль или Ассемблер, и написать на специальном языке реального времени задание на обработку информации в реальном времени – РВ-программу. В этой программе пользователь задаёт порядок обработки прикладными модулями входных данных и отображения результатов счёта по отношению к периоду поступления кадров данных в систему. Если данный порядок обработки может быть соблюдён, система обеспечит реализацию заказанной обработки. В противном случае выдаётся сообщение о невозможности вести указанную обработку. Предусмотрена возможность работы прикладных модулей с несколькими поколениями данных. Система автоматически обеспечивает хранение этих данных в специальных буферах нужное время. Предусмотрена также возможность быстрой реакции на поступление апериодической информации. Это может быть использовано, например, при возникновении внештатной ситуации с управляемым объектом либо для изменения порядка работы РВ-программы оператором. Также предусмотрено выполнение прикладных модулей в фоновом режиме. Вся остальная работа по генерации прикладной ВСРВ будет выполнена автоматически с помощью разработанной САПР ВСРВ. При этом будут решены следующие проблемы: 1) проблема тупиков при распределении ресурсов вычислительной системы (процессоров, каналов, памяти, периферийных устройств, а также информационных ресурсов, под которыми понимаются используемые в режиме разделения вспомогательные программы и информационные массивы); 2) проблема завершения тех или иных вычислений к определённым моментам времени, указанным пользователем в РВ-программе (соблюдение директивных сроков); 3) проблема сохранения и обновления необходимых наборов поколений данных как поступающих в ВСРВ извне, так и являющихся результатами вычислений с помощью РВ-программы для последующего использования в вычислениях.

При разработке САПР ВСРВ для IBM PC были реализованы: 1) язык реального времени; 2) транслятор с этого языка, включающий блоки синтаксического и семантического анализа, построения сетевой модели вычислений и нахождения допустимого расписания выполнения вычислений, автоматической генерации объектного кода программы реального времени; 3) управляющий монитор, контролирующий вычисления в режиме реального времени. Остановимся кратко на их описании.

Язык реального времени. Синтаксис языка реального времени, опуская несущественные детали, может быть описан следующим образом. Основным элементарным объектом языка является модуль – обычный для многих языков программирования оператор процедуры. Все процедуры, участвующие в образовании модулей, составляются заранее и помещаются в системную библиотеку процедур вместе с необходимыми спецификациями формальных параметров.

Фактические параметры любого модуля могут быть РВ-параметрами, константами, простыми переменными, идентификаторами с индексами и т.д. РВ-параметр определяется именем, поставщиком и поколением. Компонента "поставщик" однозначно указывает программный модуль, в котором вычисляется запрашиваемое модулем-потребителем значение параметра с данным именем. Компонента "поколение" определяет момент времени, в который необходимо взять требуемое значение параметра. В этом разделе указывается время, кратное периоду поступления информации от указанного поставщика данных.

Из модулей может быть составлен следующий по иерархии сложности объект языка – цикл реального времени, который состоит из имени РВ-цикла, периода, фазы и тела РВ-цикла. Компонента "период" задаёт интервал времени, через который будет повторяться выполнение модулей, указанных в теле РВ-цикла. Задание периода аналогично заданию времени для РВ-параметра. Кроме того, период может быть задан в абсолютных величинах времени: секундах, миллисекундах. Параметр "фаза" указывает сдвиг начала работы РВ-цикла относительно начала работы программы. Тело РВ-цикла – это список модулей, РВ-параметры которых могут поставляться из блоков входной информации, из модулей других РВ-циклов и из модулей данного цикла.

Поименованная совокупность РВ циклов образует безусловное, или простое, задание. Безусловное задание может быть снабжено конструкциями предварительной и заключительной части. Они служат для проведения набора специфических действий перед началом работы и соответственно после конца работы простого задания. Здесь может проходить инициализация переменных, каналов связи, переключение режима работы датчиков и т.д. Для облегчения программирования на языке реального времени, в новое простое задание допускается вставить несколько других простых заданий. В этом случае не придётся дублировать исходный текст ранее определённого простого задания. Основную часть простого задания составляют РВ-циклы. Все циклы реального времени, входящие в одно простое задание, должны рассматриваться как выполняющиеся параллельно во времени. Если модули некоторого цикла требуют на свой вход данные, поставляемые другим циклом, то необходимая задержка организуется системой.

В ходе проведения эксперимента может потребоваться изменить режим обработки. Такая возможность предусматривается в языке реального времени введением условного задания, которое состоит из имени задания, вектора условий, таблицы переключений. В разделе "вектор условий" определяется список булевых переменных, от значения которых будет зависеть порядок исполнения программы реального времени. Здесь же указываются начальные значения компонент вектора. Компоненты вектора условий могут быть использованы в качестве входных и выходных параметров РВ-модулем. В разделе "таблица переключений" задаются правила переключения между простыми и условными заданиями в зависимости от конкретных значений вектора условий. Таблица переключений представляется в виде списка значений вектора условий и имени простого или условного задания, на которое требуется переключиться. Таблица переключений должна однозначно определять, на какое задание должно происходить переключение, и содержать строку, соответствующую начальному значению вектора условий. Если для текущего значения вектора условий нет соответствующей записи в таблице переключений, то перехода на другое задание не происходит и продолжается работа программы в старом режиме. Наконец, РВ-программа представляет собой совокупность условных заданий, все таблицы условий которой не содержат ссылок на неописанные задания.

3.3.3. Основные блоки транслятора и управляющий монитор. Блок синтаксического и семантического анализа осуществляет синтаксический и семантический анализ конструкций РВ-программы, выдаёт сообщения о наличии ошибок в РВ-программе, генерирует таблицы данных для работы последующих блоков, вычисляет размеры буферов обмена данными между программными модулями. Блок генерации сетевой модели и расписаний: 1) строит математическую модель вычислений, выполняемых в реальном времени, в виде графа, в котором вершины соответствуют прикладным модулям пользователя, а дуги определяют частичный порядок их выполнения; 2) определяет директивные интервалы выполнения прикладных модулей; 3) определяет существование допустимого расписания выполнения прикладных модулей, и 4) строит его, если оно существует; 5) определяет необходимое количество физических копий для каждого прикладного модуля; 6) вычисляет размеры буферов для входных параметров прикладных модулей. Блок генерации кода формирует на языке С и записывает в текущий каталог исходные тексты получившейся программы, создаёт ряд вспомогательных файлов для компиляции и редактирования связей. Управляющий монитор обеспечивает работу в реальном времени прикладной программы пользователя, сгенерированной на этапе предварительной обработки посредством САПР ВСРВ. Управляющий монитор является составной частью исполняемого EXE-модуля РВ-программы. Его функциями являются: приём, хранение и обработка поступающих извне кадров данных; реакция на внешнее апериодическое сообщение, исполнение команд, вводимых с клавиатуры консоли; переключение между условными и простыми заданиями в соответствии со значениями вектора условий; запуск

процессов согласно построенным ранее расписаниям; обмен данными между процессами; действия по окончании работы процесса.

Литература

1. *Кашин Г.М., Пшеничнов Г.И., Флёров Ю.А.* Методы автоматизированного проектирования самолёта. М.: Машиностроение, 1979. 165 с.
2. *Краснощёков П.С., Флёров Ю.А.* Методология создания систем автоматизированного проектирования сложных технических объектов // Системный анализ в науке и технике. М.: Наука, 1992.
3. *Флёров Ю.А.* Математическое моделирование объектов проектирования в задачах формирования облика // В сб. "Математическое моделирование и программное обеспечение". М.: Наука, 1993. 35 с.
4. *Флёров Ю.А.* Декомпозиция и агрегирование в иерархических системах проектирования // Математика и проектирование. М.: Машиностроение, 1994. 44 с.
5. *Краснощёков П.С., Фёдоров В.В., Флёров Ю.А.* Информационные технологии и автоматизация проектирования сложных технических объектов // Информационные технологии и вычислительные системы. 1995. № 1. 8 с.
6. *Краснощёков П.С., Савин Г.И., Флёров Ю.А.* Современное состояние и тенденции развития информационных технологий в России. Министерство науки и технической политики. М., 1996. 8 п.л.
7. *Краснощёков П.С., Савин Г.И., Фёдоров В.В., Флёров Ю.А.* Автоматизация проектирования сложных объектов машиностроения // Автоматизация проектирования. 1996. № 1. 8 с.
8. *Краснощёков П.С., Фёдоров В.В., Флёров Ю.А.* Элементы математической теории принятия проектных решений // Автоматизация проектирования. 1996. № 2. 12 с.
9. *Краснощёков П.С. Фёдоров В.В., Флёров Ю.А.* Развитие математической теории принятия проектных решений // Автоматизация проектирования, 1999. №3(12). 11 с.
10. *Рабинович Я.И.* Многокритериальная иерархическая процедура улучшения опорного решения // Доклады РАН, Т. 338, № 2. С. 177-179.
11. *Рабинович Я.И.* Иерархическая многокритериальная процедура улучшения опорного решения в задачах большой размерности // ЖВМ и МФ. 1994. Т. 34, № 12. С. 1770-1781.
12. *Рабинович Я.И.* Об отыскании слабо эффективных решений // Ж. вычисл. матем. и матем. физ. 2004. Т. 44, № 4.
13. *Рабинович Я.И.* Построение множества эффективных векторов методом е-возмущений // Ж. вычисл. матем. и матем. физ. 2005. Т. 45, № 5.
14. Задачи и методы автоматизированного проектирования в авиастроении. Сб. работ. М.: ВЦ АНСССР, 1991. 121 с.
15. *Катунин В.П.* ТАБУС — интегрированная система хранения, обработки и визуализации табличных функций. М.: ВЦ РАН, 1995. 126 с.
16. *Kozyrev V.P., Yashmanov* 5. У. Representations of graphs and networks (coding, lay-out, embedding) *11* J. of Sov. Math. 1992. V.61, № 3. P. 2152-2194.
17. *Гереш П.А., Сушков Б.Г., Фуругян М.Г. и др.* Оценка параметров газовой залежи с помощью обобщённой динамической модели. М.: ВЦ РАН, 1994. 39 с.
18. *Гереш Г.М., Сушков Б.Г., Фуругян М.Г.* Регрессионный анализ падения пластового давления в системе "газовая залежь–водонапорный бассейн" // Вопросы методологии и новых технологий разработки месторождений природного газа. Ч. 1. М.: ВНИИГАЗ, 1994. 10 с.
19. *Гереш П.А., Гереш Г.М., Сушков Б.Г., Фуругян М.Г.* Газодинамическая блочная модель — универсальная методика подсчёта запасов газа и управления разработкой месторождений // Современное состояние и перспективы совершенствования методов подсчёта запасов газа по данным истории разработки. Материалы н/т совета ОАО Газпром, М., ноябрь, 1999. М.: ИРЦ ОАО ГАЗПРОМ, 2000. С. 191-203.
20. *Гереш Г.М., Лебедев В.Ю., Сушков Б.Г., Фуругян М.Г.* Особенности оценки начальных запасов при поэтапном вводе в эксплуатацию уникальных залежей углеводородов // Геология, бурение, разработка и эксплуатация газовых и газоконденсатных месторождений на суше и на шельфе. №6. М.: ИРЦ ОАО ГАЗПРОМ, 1998. С. 30-40.
21. *Козырев В.П.* Описание и порождение всех минимальных раскрасок интервального графа и решение смежных задач // ЖВМ и МФ. 1996. Т.36, № 5. С. 146-153.
22. *Вышинский Л. Л., Гринёв И. Л., Катунин В.П., Лабу тин И.В., Флёров Ю.А., Широков П.И.* Банковские информационные технологии. 4.1. М.: ВЦ РАН, 1999. 99 с.

23. *Вышинский Л. Л., Гринёв И. Л., Катунин В.П., Лабу тин И.В., Флёров Ю.А., Широков П.И.* Банковские информационные технологии. 4.II. М.: ВЦ РАН, 1999. 175 с.
24. *Вышинский Л.Л., Гринёв И.Л., Флёров Ю.А., Широков А.Н., Широков П.И.* Автоматизация проектирования прикладных информационных систем. М.: ВЦ РАН, 2003. 61 с.
25. *Вышинский Л.Л., Гринёв И.Л., Флёров Ю.А., Широков А.Н., Широков П.И.* Генератор проектов — инструментальный комплекс для разработки "клиент-серверных" систем // Информационные технологии. 2003. № 1. С. 23-42.
26. *Давыдов Э.Г.* О некоторых динамических задачах на быстродействие при распределении нескольких видов ресурсов на сетевых графиках // Вестник МГУ. 1992. Сер. 15, № 1.
27. *Давыдов Э.Г.* О расстоянии Минковского между двумя выпуклыми многогранными множествами // Вестник МГУ. 1992. Сер. 15. № 2.
28. *Давыдов Э.Г.* О редукции некоторых задач многократного минимакса со связанными ограничениями // Вестник МГУ. 1992. Сер. 15. № 2.
29. *Давыдов Э.Г.* О решении некоторых задач многократного минимакса // // Вестник МГУ. 1992. Сер. 15. № 2.
30. *Давыдов Э.Г., Багинская С.Н.* Об уравнениях Фредгольма I рода // Обратные задачи естествознания, М.: МГУ, 1997.
31. *Давыдов Э.Г., Михайлова Н.А.* О задачах линейного синтеза многопродуктовых сетей // Вестник МГУ. 1997. Сер. 15. № 2.
32. *Вышинский Л., Дунин-Барковский В., Флёров Ю., Новодворский И.* Моделирование мозжечка — необходимая оставляющая создания электронного мозга // Сумма технологий. 2002. № 1(19). С.20-28.
33. *Вышинский Л.Л., Гринёв И.Л., Дунин-Барковский В.Л., Флёров Ю.А., Широков Н.И.* Мониторинг, анализ и прогнозирование поведения многоагентных систем на базе нейрокомпьютерной модели мозжечка. М.: ВЦ РАН, 2003. 61 с.
34. *Жуков А.Н.* Двухкритериальная задача выбора авиадвигателей // Системный анализ в технике. М.: Изд-во МАИ им. С. Орджоникидзе, 1995. № 4. С. 18-36.
35. *Жуков А.Н.* Об одной модификации ракетно-турбинного двигателя // Вестник Московского Авиационного института. 1996. Т. 2, № 2. С. 30-35.
36. *Жуков А.Н.* Реактивные двигатели периодического действия: возможные схемы и оценки перспектив применения // Вестник Московского Авиационного института. 1996. Т. 2, № 3. С. 20-31.
37. *Жуков А.Н.* О возможном направлении развития силовых установок для космических транспортных систем // Вестник Московского Авиационного института. 1996. Т. 3, № 2.
38. *Жуков А.Н.* Дезагрегирование в задачах проектирования авиадвигателей // Информационные технологии. 2002. № 12. 11 с.
39. *Жуков А.Н.* Исследование некоторых задач теории реактивных двигателей методами векторной оптимизации // Информационные технологии. 2003. № 10. 10 с.
40. *Жуков А.Н.* Некоторые вопросы теории и проектирования авиадвигателей. М.: ВЦ АН СССР, 1990. 168 с.
41. *Краснощёков П.С., Флёров Ю.А.* Иерархия задач проектирования // Задачи и методы автоматизированного проектирования. М.: ВЦ РАН, 1991. С. 3-23.
42. *Вышинский Л.Л.* Управляющая система и планирование вычислений в САПР // Интерактивная технология в САПР / Тез. докл. IV всесоюзн. совещания по автоматизации проектирования электротехнических устройств. Таллин, 1981. С. 49-50.
43. *Вышинский Л.Л., Фёдоров В.В., Флёров Ю.А.* "Опыт организации диалога в САПР". "Интерактивная технология в САПР", тезисы докладов IV всесоюзного совещания по автоматизации проектирования электротехнических устройств. Таллин, 1981г. С. 51-52.
44. *Вышинский Л.Л., Медведев А.Е., Шиленко В.И., Широков Н.И.* Система формирования алгоритмов конструкторско-инженерных расчётов // Автоматизация проектирования и конструирования. II Всесоюзн. совещание. Ленинград, февраль 1983. / Тез. докл. М., 1983 г. С. 103-104.
45. *Вышинский Л.Л., Шиленко В.И., Широков Н.И.* Инструментальная система ФАКИР для описания моделей и задач в САПР. Организация математического моделирования и управления пакетами прикладных программ в САПР ЛА // Тематический сборник научных трудов. М.: МАИ, 1984, С. 3134.
46. *Вышинский Л.Л., Прибытков Ю.Д., Шиленко В.И., Широков Н.И.* Инструментальная система ФАКИР // Известия АН СССР. Техн. кибернетика. 1986. № 3. 6 с.
47. *Вышинский Л.Л., Самойлович О.С., Флёров Ю.А.* Программный комплекс формирования облика летательных аппаратов // Задачи и методы автоматизированного проектирования в авиастроении. М.: ВЦ АН СССР, 1991. С. 24-42.

48. *Вышинский Л.Л.* Структура моделей в задачах проектирования // Задачи и методы автоматизированного проектирования в авиастроении. М.: ВЦ АН СССР, 1991. С. 43-51.

49. *Вышинский Л.Л., Гринёв И.Л., Шиленко В.И., Широков Н.И.* Инструментальные средства САПР // Задачи и методы автоматизированного проектирования в авиастроении. М.: ВЦ АН СССР, 1991. С.52-70.

50. *Гринёв И.Л., Широков Н.И.* Средства управления данными в САПР // Задачи и методы автоматизированного проектирования в авиастроении". М.: ВЦ АН СССР, 1991. С. 71-80.

51. *Катунин В.П.* Основы построения программного комплекса лётно-технических характеристик в САПР Л А // Задачи и методы автоматизированного проектирования в авиастроении. М.: ВЦ АН СССР, 1991. С. 81-91.

52. *Скобелев С.И., Широков Н.И.* Весовой анализ и контроль в САПР ЛА // Задачи и методы автоматизированного проектирования в авиастроении. М.: ВЦ АН СССР, 1991. С. 52-70.

53. *Шапиро М.Ю., Шиленко В.И.* Автоматизация проектирования электросхем // Задачи и методы автоматизированного проектирования в авиастроении. М.: ВЦ АН СССР, 1991. С. 101-105.

54. *Прибытков Ю.Д., Шиленко В.И.* Программный комплекс ЭЛМОН // Задачи и методы автоматизированного проектирования в авиастроении. М.: ВЦ АН СССР, 1991. С. 106-120.

55. *Катунин В.П.* ТАБУС — интегрированная система хранения, обработки и визуализации табличных функций // М.: ВЦ РАН., 1995. 125 с.

56. *Краснощёков П.С., Савин Г.И., Фёдоров В.В., Флёров Ю.А.* Автоматизация проектирования сложных объектов машиностроения // Автоматизация проектирования. 1996. № 1.

57. *Вышинский Л.Л., Гринёв И.Л., Демидов А.Ю., Широков Н.И.* Технологии разработки и сопровождения АБС // Банковские технологии. 1997. № 6. С. 44-49.

58. *Вышинский Л.Л.* Новые правила бухгалтерского учёта в системе ГАМБИТ // Банки и технологии. 1997. № 4. С. 36-37.

59. *Вышинский Л.Л.* Технология перехода на новый план счетов в системе ГАМБИТ // Третий международный форум разработчиков банковских систем. Тез. докл., Москва. 1997. С. 4-6.

60. *Гринёв И.Л.* Проблемы перевода банка на новые информационные технологии. Опыт внедрения системы ГАМБИТ в Башкирском банке СБ РФ // Третий международный форум разработчиков банковских систем. Тез. докл., Москва. 1997. С. 101-105.

61. *Вышинский Л.Л., Гринёв И.Л., Катунин В.П., Лабутин И.В., Флёров Ю.А. Широков Н.И.* Банковские Информационные технологии (Ч. I и И). М.: ВЦ РАН 1999. 272 с.

62. *Вышинский Л.Л., Широкова Е.Н. Автоматизированная система биллинга* // Автоматизация проектирования финансовых информационных систем. М. ВЦ РАН, 2004. С. 75-86.

63. *Гринёв И.Л., Логинов А.А., Широков А.Н., Широков Н.И.* Система банковского самообслуживания // Автоматизация проектирования финансовых информационных систем. М.: ВЦ РАН, 2004. С.87-94.

64. *Гринёв И.Л., Логинов А.А., Широков А.Н., Широков Н.И.* Система мобильного банковского обслуживания // Автоматизация проектирования финансовых информационных систем. М.: ВЦ РАН, 2004. С. 95-101.

65. *Гринёв И.Л., Логинов А.А., Широков Н.И.* Система ИНТЕРНЕТ-платежей // Автоматизация проектирования финансовых информационных систем. М.: ВЦ РАН, 2004. С. 102-109.

66. *Вышинский Л.Л., Гринёв И.Л., Широков Н.И., Щедрое В.И.* Система урегулирования задолженностей // Автоматизация проектирования финансовых информационных систем. М.: ВЦ РАН, 2004. С. 110-118.

67. *Вышинский Л.Л., Гринёв И.Л., Флёров Ю.А., Широков Н.И.* Проектный подход к разработке информационных систем // Автоматизация проектирования финансовых информационных систем. М.: ВЦ РАН, 2004. С. 8-22.

68. *Вышинский Л.Л.* Проект банковской системы // Автоматизация проектирования финансовых информационных систем. М.: ВЦ РАН, 2004. С. 63-74.

69. *Вышинский Л.Л., Гринёв И.Л., Флёров Ю.А., Широков А.Н., Широков Н.И.* Генератор проектов – инструментальный комплекс для разработки "клиент-серверных" систем // Информационные технологии и вычислительные системы. 2003. № 1-2. С. 6-25.

70. *Вышинский Л.Л., Гринёв И.Л., Флёров Ю.А., Широков А.Н., Широков Н.И.* Автоматизация проектирования прикладных информационных систем. М.: ВЦ РАН, 2003. 61 с.

71. *Широков А.Н.* Организация сетевого взаимодействия при автоматизации разработки программных комплексов. М.: ВЦ РАН, 2004, 26 с.

72. *Широков Н.И.* Генератор проектов // Автоматизация проектирования финансовых информационных систем. М. ВЦ РАН, 2004. С. 23-43.

73. *Широкова Е.Н.* Генерация программного кода в Генераторе проектов // Автоматизация проектирования финансовых информационных систем". М.: ВЦ РАН, 2004. С. 57-62.
74. *Широков А.Н.* Обеспечение информационной безопасности в архитектуре клиент-сервер // Автоматизация проектирования финансовых информационных систем. М.: ВЦ РАН, 2004. С. 44-56.
75. САПР систем реального времени для IBM PC / Под ред. *Ю.А. Флёрова*. М.:ВЦ РАН, 1993.
76. Теория и реализация вычислительных систем реального времени / Под ред. *А.В. Мищенко*. М.:ВЦ РАН, 1999.
77. *Танаев В.С., Гордон В.С., Шафранский Я.М.* Теория расписаний. Одностадийные системы. М.: Наука, 1984.
78. *Federgruen A., Groenevel H.T.* Preemptive Scheduling of Uniform Machines by Ordinary Network Flow Technique 11 Management Science, March 1986. V. 32, No. 3.
79. *Gonzales T., Sahni S.* Preemptive Scheduling of Uniform Processor Systems // Journal of the Association for Computing Machinery. January 1978. V. 25, № 1.
80. *Фуругян М.Г.* Построение периодических расписаний в многопроцессорных АСУ реального времени // Автоматика и телемеханика. 2000. № 9. С. 202-205.
81. *Гречук Б.В., Фуругян М.Г.* Составление оптимальных расписаний с прерываниями в многопроцессорных системах с неполным графом связей. М.: ВЦ РАН, 2004.
82. *Гуз Д.С., Красовский, Фуругян М.Г.* Эффективные алгоритмы планирования вычислений в многопроцессорных системах реального времени. М.: ВЦ РАН, 2004.
83. *Коффман Э.Г.* Введение в детерминированную теорию расписаний // Теория расписаний и вычислительные машины / Под ред. Э.Г. Коффмана М.: Наука, 1984. С. 9-64.
84. *Гуз Д.С., Фуругян М.Г.* Планирование вычислений в многопроцессорных АСУ реального времени с ограничениями на память процессоров // Автоматика и телемеханика. 2005. № 2. С. 138-147.
85. *Белый Д.В., Сушков Б.Г.* Модель организации рестартов в системах реального времени. М.: ВЦ РАН, 1996.
86. *Гречук Б.В., Фуругян М.Г.* Алгоритмы организации рестартов в системах реального времени. М.: ВЦ РАН, 2004.
87. *Фуругян М.Г.* Решение одной задачи распределения ресурсов в АСУ реального времени при наличии неопределённых факторов // Автоматика и телемеханика. 2002. № 11. С. 167-171.
88. *Вышинский Л.Л., Гринёв И.Л., Демидов А.Ю., Широков Н.И.* Технологические аспекты разработки и сопровождения банковских систем // Банковские технологии, июль-август 1997. № 6. С. 44-49.
89. *Жуков А.Н.* Об одной задаче теории авиадвигателей // Доклады РАН. Т. 132, № 6. С. 699-701.

Из книги:

50 лет ВЦ РАН: история, люди, достижения (М.: ВЦ РАН, 2005.
ISBN 5-201-09837-1. С. 248-272)

Подписано в печать 05.10.2005 Формат бумаги 60x84 1/8 Уч.-изд. л. 40,3.
Усл.-печ. л. 40. Тираж 500 экз.

Отпечатано в ППП «Типография «Наука» Академиздатдентра РАН 121099
Москва Г-99, Шубинский пер., 6 Заказ № 1790