Российская академия наук
Сибирское отделение
Институт систем информатики
им. А. П. Ершова

НОВОСИБИРСКАЯ ШКОЛА ПРОГРАММИРОВАНИЯ.

Перекличка времен

Под редакцией
проф. И. В. Поттосина,
к.ф.-м.н. Л. В. Городней

Новосибирск 2004

УДК 007.621.391
ББК 32.81

Новосибирская школа программирования. Перекличка времен. — Новосибирск: Ин-т систем информатики им. А.П. Ершова СО РАН, 2004. — 244 с.

Настоящий сборник содержит статьи с представлением разнообразных явлений, сопутствовавших развитию программирования в России и проявляющихся на уровне управления информатикой как наукой, оценки ее эффективности, сохранения ее результатов и передачи их следующим поколениям. Сборник посвящен И. В. Поттосину, давшему личный пример в освещении пионерской эпохи отечественного программирования. Материалы сборника представляют интерес для специалистов по истории информатики и менеджеров программных проектов.

Работа была поддержана грантом РГНФ № 00-03-00277.

ПРЕДИСЛОВИЕ

Предлагаем читателям очередной сборник материалов, отражающих историю новосибирской школы программирования и информатики, дополняющих изложенную в 2001 году тематическую картину и связывающих ее с современностью. Сборник посвящен И. В. Поттосину, сделавшему значительный вклад как на страницах истории программирования, так и в развитие системного программирования как науки. Влияние личности Игоря Васильевича на молодые умы и становление программисткой профессиональной элиты неоценимо. И. В. Поттосин дал импульс проекту предоставления материалов по истории программирования широкой общественности и подал личный пример в освещении наиболее интересной пионерской эпохи отечественного программирования.

Настоящий сборник содержит статьи с общим анализом разнообразных явлений, сопутствовавших развитию научных исследований в программировании и проявляющихся на уровне управления наукой, оценки ее эффективности, сохранения ее результатов и передачи их следующим поколениям. Представлена информация об истории развития средств и методов верификации программ и различных характеристик процессов, а также комментарии по наиболее известным проектным решениям и методам разработки языков программирования с позиций сегодняшнего дня. Эти материалы естественно дополняют статьи о моделях жизненных циклов программ, динамике представления знаний и человеческом факторе в программировании, включая проблемы преподавания информатики в университетах.

Работа была поддержана грантом РГНФ № 00-03-00277.

к.ф.-м.н. Л. В. Городняя

И. В. Поттосин, Л. В. Городняя, Н. А. Калинина

ИЗЛОЖЕНИЕ ИСТОРИИ ИНФОРМАТИКИ УЧАСТНИКАМИ И ОЧЕВИДЦАМИ

В этой статье объединены фрагменты из заявки и отчетов по теме «Исследование и изложение истории отечественной информатики как вклада в мировую науку», выполненные при личном участии и под руководством И. В. Поттосина.

1. ВВЕДЕНИЕ

Изложение и исследование достижений отечественной информатики и программирования непосредственно их авторами, способными предоставить обществу ретроспективный анализ многих известных им явлений, решений, проектов и идей, дает возможность восстановить и внимательно проанализировать путь этой новой науки от становления и самобытного расцвета до интеграции в мировую информационную культуру и передачи эстафеты следующим поколениям.

Проект «Исследование и изложение социальной истории отечественной информатики как вклада в мировую науку» развернут в 1999 году при поддержке РГНФ. Цель проекта — исследование истории отечественной информатики, излагаемой участниками становления и развития этой науки в нашей стране. В центре проекта — творческие традиции авторитетной школы информатики и программирования, заложенные академиком А.П. Ершовым еще в 60-е годы [5, 10]. Умение А.П. Ершова налаживать результативные формы научного общения создавало у его сотрудников ощущение работы на переднем крае мировой науки. Рядом с ним молодежь знакомилась с классиками компьютерной науки, с ведущими школами отечественного и мирового программирования [7]. Многие деятели информатики, помнящие его, могли бы предоставить интереснейшие материалы исторического характера [8, 9].

Такой субъективный подход к исследованию призван убедительно и документально отразить вклад российских ученых в мировую науку и компьютерную инженерию. Стремительное развитие информатики позволило в короткий срок проследить закономерности эволюции научных идей, смену

парадигм и форм концентрации знания, ставшие доступными для использования и компьютерного эксперимента.

Представление результатов исследований специалистам гуманитарного профиля (историкам, педагогам, психологам, философам, методологам и др.) может дать многое для более глубокого понимания человеческого фактора в специфике разработки и применения информационных систем, что может быть полезно и информатикам, интересующимся социальной историей становления и развития информатики в Сибири.

Данный проект ориентирован на накопление, представление и использование знаний по истории информатики и программирования в процессе информатизации образования. Новосибирская школа программирования имеет широкую мировую известность. Начало публикации материалов по истории информатики положено Д.А. Поспеловым и Я.И. Фетом [1].

Все, кому это интересно, приглашаются к участию, обсуждению и сотрудничеству. Уже в настоящее время предоставили материалы многие известные в программировании ученые и специалисты [5]. Большинство из них посвящены начальному этапу работ коллектива ИСИ СО РАН им. А. П. Ершова по исследованию теории и практики программирования, а также их становлению в 50–70-е гг., но тематическая картина этих работ далеко не полна.

2. ЦЕНТРАЛЬНАЯ ИДЕЯ

Творческие традиции Новосибирской школы информатики и программирования, заложенные академиком А.П. Ершовым еще в 60-е годы, рассматриваются как ключевая позиция для выстраивания материала. Проекты и результаты этой известной в компьютерном мире школы отражены в ряде материалов исторического характера, подготовленных в связи с полосой юбилейных дат [2–4]. Динамика научных интересов Андрея Петровича, видение им перспектив науки настолько существенно повлияли на тематику исследований многих его коллег и учеников, определили содержание многих проводимых при его участии мероприятий, что могут служить основным ориентиром при организации собираемого материала. Роль академика Ершова в информационном обеспечении сибирских программистов и отечественной науки «информатика» нельзя переоценить [5, 10–12].

Молодость информатики как науки и высокий темп ее развития позволили в короткий срок пронаблюдать закономерности эволюции научных идей, смену поколений техники, программных систем и технологий, пара-

дигм и других форм концентрации знания человеком. Причем на глазах одного поколения осуществился полный цикл развития информатики от становления первых экспериментов до зрелой индустрии [2, 3, 6, 14].

3. ПОДХОД К СБОРУ МАТЕРИАЛОВ И ИХ ОБРАБОТКЕ

Задачей проекта является сбор мозаики воспоминаний о различных видах деятельности отдела программирования ВЦ СО АН СССР и связанных с ним организаций, которая бы представила историю новосибирской школы программирования, а также упорядочение и анализ материалов, отражающих шаги этой достойной уважения и чрезвычайно интересной истории.

В настоящее время во многих странах и научных центрах налажена регулярная работа по сбору и анализу материалов в области развития научной мысли и технического творчества. Результаты такой работы служат основой для исследовательской деятельности и издания научно-популярной литературы и других средств передачи научного знания широкой публике [1, 11].

Современные представления по истории информатики, используемые нашей образовательной системой, отражают преимущественно зарубежные достижения, значение которых очевидно благодаря высокому качеству элементной базы. Информатика и информационные технологии — бурно развивающаяся область, образующая перспективную основу жизни нашего тысячелетия. Поэтому естественно развернуть проект, который выполнял бы работу по упорядочению и сохранению свидетельств истории отечественной вычислительной техники и ее программного обеспечения.

Многие участники разработок первых ЭВМ, операционных систем, реализаций языков и систем программирования, пакетов прикладных программ и баз данных, средств машинной графики и поддержки проектирования могут представить для исследования уникально ценные свидетельства научной дерзости, самостоятельности и кругозора, позволивших отечественной информатике при бедности ресурсов и сложности условий внести заметный вклад в мировую науку. Начало такому делу, положенное монографией, составленной Д.А. Поспеловым и Я.И. Фетом [1], почти не затрагивает истории отечественного программирования и школы А.П. Ершова. И.В. Поттосин приступил к восполнению этого пробела. Авторы подготовленного под его редакцией и руководством сборника имеют богатый опыт разработки информационных систем в разных областях знаний, а также опыт преподавания программирования и информатики на всех уровнях об-

щеобразовательной системы — с младших классов до факультетов повышения квалификации педагогов, включая преподавание специалистам гуманитарных специальностей, таких как журналистика, гуманитарная информатика, педагогика и др. [12]. Сборник дает достаточно полный обзор основных направлений становления новосибирской школы программирования, представлены наиболее активные участники этого процесса. Даны живые рассказы о связанных с деятельностью событиях прямо так, как помнится. В представленных материалах описаны вычислительная техника и околокомпьютерная деятельность в нашей стране и в мире.

Естественно, это только начало, лишь отчасти отражены наиболее видные разработки, и пока не удалось указать всех тех, кто был как-то связан с новосибирской школой, и привлечь их к участию в первых выпусках сборника, но все, кому это интересно, могут откликнуться и дать свои материалы, дополнить то что уже собрано и описать свое видение истории информатики.

Далее необходим переход к сегодняшнему взгляду на деятельность новосибирской школы (время прошло, и нынешнее понимание может отличаться от тогдашнего), а также анализ и проявление разного рода закономерностей, отраженных в известной нам истории программирования как науки, технологии и деятельности.

4. ПЕРВООЧЕРЕДНЫЕ НАПРАВЛЕНИЯ

При отборе и подготовке материала в качестве первоочередных выделены следующие аспекты.

- Обзор ключевых идей и проектов, определивших лицо отечественного программирования.
- Изыскания в области теории программирования, исследований по схематологии, сетевым моделям, правильности и верификации протоколов общения.
- История информатики, проекты и рабочие группы. Терминология и наукометрия. Оценка результатов и сферы влияния разных научных школ.
- Эволюция истории языков и систем программирования, языковедение, кодировки и лингвистические проблемы информатики.
- Экспериментальная информатика и программирование.

- История искусственного интеллекта, экспериментов по лингвистическим процессорам и глобализации идей информатики до уровня экологических проблем.
- Автоматизация полиграфической деятельности.
- Школьная информатика и информатизация образования. Информатика как гуманитерная наука. Философия информатики и информатика образования.
- Символьные вычисления и развитие методов компьютерной алгебры.
- Машинно-ориентированное программирование.
- Памятные даты и важные мероприятия.

5. ЭСТАФЕТА ПОКОЛЕНИЙ

В связи с памятными датами Новосибирских Летних школ юных программистов (50-летие Г.А. Звенигородского) следует помнить, что в круг интересов академика А. П. Ершова органично входили проблемы обучения программированию [7].

Образовательный потенциал программирования А.П. Ершов отмечал еще в отчете о своей первой поездке в Англию. Знаменитый Альфа-проект, на который до сих ссылаются, сопровождался впечатляющими краткосрочными курсами по программированию, заметно пополнившими ряды хороших программистов не только в Новосибирске. Уже в начале 60-х годов сотрудники отдела Ершова учили программированию как трудовой профессии школьников, многие из которых сохранили верность программированию до сих пор.

Обескураживает огорчительно нередкое среди старшеклассников убеждение, что взрослые в принципе не компетентны в области компьютерных наук. Действительно, что еще думать активно программирующему шестикласснику, когда учительница информатики заставляет его переписывать из книги в тетрадку назначение разделов меню Windows [13].

В настоящее время все больше полезной для образования информации доступно по Интернету. Много ли там сведений об отечественных достижениях в области компьютерных наук? Как выразить и сохранить основы предмета, его специфику, моделирующую силу для представления знаний практически из любых областей, способность стимулировать учащихся к творческому развитию познавательных способностей и интегрирующий потенциал, дающий образовательной системе явный шанс выстоять в экономически сложных условиях? По другим наукам уже многое можно уз-

нать через Интернет, а сапожник, увы, без сапог. Естественно — начало в истории.

Самый надежный способ экономии затрат — поручать работу тем, кто может с нею справиться наилучшим образом. Историей программирования и информатики лучше всего владеют ее участники и очевидцы. Мы должны знать свою историю и рассказывать о ней. Идеи и результаты ранней эпохи развития программирования нашли свое выражение и подтверждение во многих проектах сегодняшнего дня. Знание подходов и методов, прочно вошедших в практику современного программирования, является крепкой основой будущих исследований и практических разработок [4, 6, 8, 14].

6. ЗАКЛЮЧЕНИЕ. КОНКРЕТНЫЕ РЕЗУЛЬТАТЫ

Важнейшей задачей первого года было определить подход к сбору и изложению материала, сформировать авторский коллектив и привлечь к нему общественное внимание. Эта непростая задача по существу решена не вполне: многих знающих людей сковывают опасения, что их личное мнение может нарушать чьи-либо интересы. Видимо такая осторожность заслуживает особого уважения.

Тем не менее привлечено большое число компетентных специалистов (кроме участников проекта) к написанию воспоминаний.

В основу собираемого материала положена динамика научных интересов Андрея Петровича Ершова. В собранных материалах отражена уникальность научных и технологических решений, компенсировавших трудоемкость эксплуатации отечественной техники. Перспективы развития информатики и программирования, отслеживаемые А.П. Ершовым, существенно повлияли на тематику исследований участников проекта, предоставивших для дальнейших разработок материалы из личных архивов и написавших интересные воспоминания о становлении отечественной информатики как науки. Установлен состав реально доступных для исследования материалов. Собран начальный комплект очерков и заметок ветеранов информатики [12]. Многие материалы впервые представлены на суд читателей.

Разрабатываются отдельные блоки для учебника по истории отечественной информатики, опробованные в составе разных университетских курсов, а также для старшеклассников и системы повышения квалификации школьных педагогов-информатиков, психологов и журналистов. Особый интерес представляют материалы по работе Комиссии ГКНТ по системно-

му математическому обеспечению, проводившей регулярное обобщение исследований по терминологии [14] (в сравнении с убогой речевой практикой современного информатического производства, спотыкающегося при необходимости говорить по-русски).

Имеются материалы по наукометрии, включающие оценку эффективности незримых коллективов, и некоторый критический обзор ряда выбранных в свое время решений. Упорядочена история особо значимых мероприятий по тематике программирования, проводимых в Новосибирске.

Следующий год весьма оживили дискуссии по истории информатики и этическим проблемам программирования, возникшие при проведении IV Международной конференции памяти академика А.П. Ершова «Перспективы систем информатики» [13]. По сути дела эти дискуссии дали экспертную оценку собираемому материалу и его компетентное уточнение наиболее авторитетными специалистами по истории отечественной информатики.

Выпуск «Мозаика воспоминаний» показал становление информатики в Новосибирском научном центре СО АН, начиная с «досибирского» периода формирования коллектива отдела программирования ВЦ СО РАН, отразил наиболее яркие проекты и общественный отклик на них. Никольское и первые годы. Зарождение Альфа-технологии. Новосибирское начало. Языковедение и внутренний язык в проекте Бета. История языков Эпсилон, Сигма, библиотечных СУБД. Эксперименты с языками Лисп, Сетл, Литтл, Алгол-68. Школьная информатика — Новосибирские Школы Юных программистов, системы Шпага и Школьница, языки Робик и Рапира. Производственное программирование в КБ СП [12].

Выпуск, посвященный Комиссиям ГКНТ, показывает выделение приоритетных направлений и первоочередных аспектов. Они дают исходные позиции для изучения ключевых идей и проектов, рабочих групп, терминологических систем и оценки результатов и сферы влияния разных научных школ, а также для сопоставления эволюции языков и систем программирования, анализа вопросов языковедения и лингвистических проблем информатики [14].

2002 год ознаменовался многочисленными ремонтами служебных помещений и переездами из комнаты в комнату. В результате подвергнуты инвентаризации и упорядочению ряд личных научных архивов участников проекта, в которых отобраны источники для сопоставления аналитических оценок и критического анализа, технические решения и технологические рекомендации по разработке широкого класса информационных систем. Наиболее существенные материалы подвергнуты пересмотру с позиций

сегодняшних воззрений, результаты которого представлены в форме комментариев к текстам источников.

Основные материалы отражены в публикациях, и целесообразно их размещение в Интернете с обсуждением на дискуссионном форуме, который, мы надеемся, позволит проанализировать альтернативные точки зрения на становление и развитие отечественной истории информатики и программирования в Новосибирске, а также, возможно, привлечет к проекту более широкий круг знатоков истории программирования и вычислительной техники в нашей стране и за рубежом.

Подготовлен к изданию очередной выпуск бюллетеня, продолжающий «Мозаику воспоминаний». Новый выпуск «Перекличка времен» отражает развитие информатики в Новосибирском научном центре СО АН в период с начала 70-х годов до конца 90-х, отражает наиболее оригинальные проекты, имеющие общественный отклик и продолжение в наши дни.

В распоряжение проекта предоставили свои воспоминания Э.З. Любимский, М.Р. Шура-Бура, Е.А. Жоголев, Д.Я. Левин, Л.А. Корнева, Л.Л. Змиевская, Е.И. Никольников, А.В. Замулин, Н.В. Шилов, Г.А. Сапрыкина, и другие компетентные специалисты и знающие люди, работавшие в разные годы в отделе программирования ВЦ СО РАН и близких к нему организациях.

Неоценимую помощь в выполнении ряда важнейших технических работ по проекту оказали В.В. Иванова, О.В. Дробышевич, З.В. Скок и И.А. Кирпотина.

СПИСОК ЛИТЕРАТУРЫ

1. Очерки истории информатики в России / Сост. Д.А.Поспелов, Я.И.Фет. — Новосибирск: Наука, 1998. — 662 с.
2. Любимский Э.З., Поттосин И.В., Шура-Бура М.Р. От программирующих программ к системам программирования (российский опыт) // Компьютеры в Европе. Междунар. симп. по вкладу европейских ученых в развитие и достижения компьютерных технологий. — Киев, 1998. — С. 72–78.
3. XL лет Отделу Программирования. X лет Институту систем информатики СО РАН им. А.П. Ершова. —Новосибирск: ИСИ им. А.П.Ершова, 2000. — `http://www.iis.nsk.su/pottosin/40/win/book00.html`
4. Ершов А.П., Шура-Бура М.Р. Становление программирования в СССР Новосибирск, 1976. — (Препр. / СО РАН. ВЦ СССР; № 12,13).
5. Ершов А.П. Избранные труды. — Новосибирск: Наука, 1994.
6. Системное программирование: Сб. материалов Всесоюз. симп., посвященного памяти Г.И. Кожухина. — Новосибирск, 1973.

7. Проблемы школьной информатики: Сб. науч. тр., посвященный памяти Г.А. Звенигородского. — Новосибирск, 1986.
8. Бульонков М.А., Городняя Л.В., Касьянов В.Н., Котляров В.П., Цейтлин Г.Е., Шилов Н.В. Творческое наследие В.Э. Иткина // Кибернетика и системный анализ. — 1993. — № 2. — С. 175–183.
9. Городняя Л.В., Евстигнеев В.А., Калинина Н.А., Касьянов В.Н., Мурзин Ф.А. Изложение отечественной истории информатики для школы // Тр. Междунар. конф. «Использование новых информационных технологий в образовании». — Троицк, 1997.
10. Поттосин И.В. А.П. Ершов и становление Новосибирской школы программирования. — Там же. — С. 100–107
11. Левитин К. Прощание с Алголом. — М.: «Знание», 1989. — 223 с.
12. Становление новосибирской школы программирования. Мозаика воспоминаний: Сб. статей / Под ред. И.В. Поттосина. — Новосибирск, 2001. — 195 с.
13. Перспективы систем информатики. <Школьная информатика>: Тез. докладов IV Междунар. конф. — Новосибирск, 2001. — 100 с.
14. Евстигнеев В.А. Деятельность новосибирских ученых в области программирования (по материалам Комиссии по системному математическому обеспечению ККВТ АН СССР). — Новосибирск, 2002. — 55 с. — (Препр. / СО РАН. ИСИ. № 83).

Н. А. Калинина

ПЕРВЫЙ РУКОВОДИТЕЛЬ

Памяти
Игоря Васильевича Поттосина

15 декабря 2003 года исполнилось 2 года со дня смерти Игоря Васильевича Поттосина. Большая часть моей рабочей жизни была связана с Игорем Васильевичем. Он был научным руководителем моих курсовых и дипломных работ, научным руководителем диссертационной работы, заведующим кафедрой программирования, где я долгие годы была ученым секретарем.

Первый раз я увидела Игоря Васильевича на семинаре отдела программирования на ВЦ, где Андрей Петрович Ершов представлял темы курсовых работ для студентов. Андрей Петрович исписал доску сверху донизу темами работ, потом представлял тему, говорил, сколько студентов требуется для каждой темы, определял желающих. И одной из последних тем была тема «Системная программа выполнения аналитических выкладок на ЭВМ». Требовалось на эту тему два человека, и мы с Олей Мальковой решили записаться на нее.

Руководителем был И.В. Поттосин. После окончания семинара мы пошли в комнату к Игорю Васильевичу. Он был худой, шея особенно выделялась.

И начались еженедельные обсуждения, которые часто проходили вчетвером. Игорь Васильевич Поттосин, Геннадий Иссаакович Кожухин и я с Олей Мальковой. Игорь Васильевич был необыкновенно внимателен и деликатен.

Как-то раз Оля не пришла, заболела. Он выяснял, что с ней, не надо ли чего. Потом решил, что Ольге требуется одеяло, и я пошла к нему домой, они тогда жили в «хрущевке» на Терешковой, и его жена передала одеяло для заболевшей Ольги.

До системной программы «АНАЛИТИК», которая выполнялась в рамках проекта АИСТ-0, в отделе программирования велись работы по созданию ДИФПРОЦЕССОРА. Вначале мы разбирали эту программу, потом решили, что будем писать по-другому, что у нас будет система, которая должна выполнять не только символьное дифференцирование, но главное — различные подстановки, используя которые, можно делать очень многие преобразования.

Встал вопрос о внутреннем представлении данных в системной программе АНАЛИТИК.

Кстати, это название придумал Андрей Петрович, и потом, когда мы уже к нему привыкли, Игорь Васильевич, смущенно как-то раз сказал, что у нас «АНАЛИТИК» и у киевлян «АНАЛИТИК». Но, наверное, такое название в то время витало в воздухе.

Мы остановились для представления символьных данных на схемах Канторовича.

Была определяющая работа Л.В. Канторовича, опубликованная в ДАН СССР в 1957 году. Были работы ленинградской группы математиков, успешно использовавшие схемы Канторовича в полиномиальных прорабах.

Тема дипломной работы формулировалась как «Системная программа АНАЛИТИК». Игорь Васильевич решил, что такую совместную работу нужно представлять одним текстом.

Подходило время защиты, деканат говорит, что случай нестандартный: один текст на двух человек. Нужно решение кафедры. И предложили сходить на кафедру в Институт математики. Мы с Олей к Игорю Васильевичу, так мол и так, он сказал, что договорился с Л.В. Канторовичем, но нам надо прийти и рассказать о работе самим. Мы с Олей пошли в математику (Институт математики), с нами очень доброжелательно поговорили какие-то седовласые дяденьки, один из которых был Л.В. Канторович. Разрешили представить один текст.

После окончания университета я стала проситься в аспирантуру к Игорю Васильевичу. Он мягко отказывал: «Я еще не «кандидат». И вот вскоре была его защита кандидатской диссертации в Институте математике. Защита была на сцене в конференц-зале института. Игорь Васильевич сильно волновался: и просто так, и потому что многие математики в то время область деятельности программистов не относили к математической деятельности.

Вскоре я стала аспиранткой Игоря Васильевича. Андреем Петровичем Ершовым было создано Конструкторское Бюро Системного Программирования (КБСП), которое затем в Новосибирске переросло в Новосибирский филиал Института точной механики и вычислительной техники.

Какое-то время Игорь Васильевич по совместительству работал там начальником отдела. Моя работа была по-прежнему связана с выполнением аналитических преобразований на ЭВМ. В рамках совместных работ Вычислительного центра СО АН СССР и НФ ИТМ и ВТ выполнялась разработка системы аналитических преобразований для машины ЭЛЬБРУС. Игорь Васильевич был руководителем темы. Мы вели с Игорем Васильеви-

чем долгие обсуждения входного языка системы, подходов к реализации. И в качестве внутреннего языка мы снова остановились на схемах Канторовича. В нашей реализации макетной системы АУМ на БЭСМ-6 для них использовался мультикомандный вариант представления.

Сразу же как я стала аспиранткой Игоря Васильевича, он стал поручать мне руководство дипломными работами студентов.

Помню один короткий разговор: «Нина, Вы как обращаетесь к студентам, на «Вы» или на «ты»?» Сам Игорь Васильевич всегда относился к студентам крайне предупредительно и уважительно. Я удивилась и говорю «Когда как. Когда на «Вы», когда на «ты». Игорь Васильевич многозначительно помолчал, потом сказал: «Я всегда на «Вы».

При всем том, он был очень принципиален и не шел ни на какие компромиссы, которые могли бы повредить кафедре. Вот один из примеров.

Игорь Васильевич был руководителем спецсеминара «Системное программирование». Для большинства студентов кафедры программирования этот семинар был обязательным спецсеминаром. Необходимым условием получения зачета было его посещение. Допускалось за год максимум два пропуска. Однажды ко мне, как к секретарю кафедры, приходит плачущая студентка. Скоро защита, а она не может получить зачет по семинару, у нее три пропуска. Я пошла ходатайствовать за девушку, тем более, она была дочкой бывшей хорошей студентки Игоря Васильевича. Но ничего не помогло. Игорь Васильевич сказал «нет», зачета он не поставит.

В то же время, если имелась возможность опубликовать хорошую работу студента, изыскать возможность поездки и участия студента в конференции, Игорь Васильевич всегда старался помочь и всеми силами способствовал этому.

И слова «схемы Канторовича» всегда связаны у меня с образом Игоря Васильевича Поттосина, деликатного, строгого и доброго человека.

Н. А. Калинина, И. В. Поттосин

ИССЛЕДОВАНИЕ СОЦИАЛЬНОЙ ИСТОРИИ ОТЕЧЕСТВЕННОЙ ИНФОРМАТИКИ: СИБИРСКАЯ ШКОЛА ПРОГРАММИРОВАНИЯ

Сибирская школа программирования имеет широкую мировую известность. В данной работе рассматриваются работы, связанные с проектом АИСТ-0 и ранние работы по символьным и аналитическим преобразованиям.

ПРЕДИСЛОВИЕ

Эта статья готовилась нами к прошлогоднему сборнику. Игорь Васильевич просил статью переработать. Основное его замечание было содержательное: «Мы пишем историю — хотелось бы, чтобы эта статья содержала бы больше личных впечатлений и личных моментов». И этот «личный момент» задержал статью, а в декабре Игоря Васильевича не стало. И теперь мы уже вспоминаем с благодарностью об Игоре Васильевиче Поттосине. Эту статью мы практически оставили в варианте последней правки Игоря Васильевича.

1. ВВЕДЕНИЕ

«Любовь к отеческим гробам»
А.С.Пушкин

Сибирская школа программирования имеет широкую мировую известность. В данной работе рассматриваются работы по символьным и аналитическим преобразованиям, проводимые в СО АН СССР с 1965 по 1975 годах, и работы, связанные с проектом АИСТ-0. В 1966 г. выдающийся ученый Андрей Петрович Ершов в рамках отдела программирования и конструкторского бюро системного программирования начал работы по созданию автоматических информационных станций АИСТ. Первой частью этого проекта была разработка системы разделения времени АИСТ-0. Такие свойства системы, как разделение в процессах комплекса управления и об-

работки, иерархичность строения программного обеспечения, выделения ядра операционной системы, естественное сочетание режимов взаимодействия, обеспечили должную эффективность системы АИСТ-0.

2. ПРОЕКТ АИСТ-0

В 1966 г. Андреем Петровичем Ершовым были инициированы работы по автоматическим информационным станциям (проект АИСТ). Этот проект объединял широкий круг исследований по архитектуре вычислительных комплексов, их программному обеспечению и моделированию вычислительных систем. В рамках этого проекта была создана первая в стране развитая система разделения времени АИСТ-0. Система была реализована на многомашинном комплексе из отечественных ЭВМ. Такие свойства системы, как разделения в процессах комплекса управления и обработки, иерархичность строения программного обеспечения, выделение ядра операционной системы, продуманное сочетание различных режимов общения и обработки, способствовали хорошей эффективности и гибкости системы.

В состав системных программ АИСТ-0 входила и системная программа аналитических преобразований АНАЛИТИК.

3. РАБОТЫ ПО АНАЛИТИЧЕСКИМ ПРЕОБРАЗОВАНИЯМ

Работа по созданию систем с возможностью выполнения аналитических выкладок развивалась вокруг трех направлений, а именно: создание аналитических процессоров для универсальных систем программирования, создание систем, работающих в режиме разделения времени и выполняющих универсальные аналитические преобразования и разработку специализированных систем.

Необходимость в вычислительных задачах часто выполнять аналитическое дифференцирование привело к созданию препроцессоров, которые выполняют перевод с расширенного языка программирования, допускающего дополнительную операцию дифференцирования на язык описания вычислительных алгоритмов. Одним из таких дифпроцессоров был ДИФПРОЦЕССОР, предназначенный для работы вместе с системой АЛЬФА. ДИФФПРОЦЕССОР, помимо дифференцирования и подстановки, давал возможность выполнять приведение подобных, сокращение дробей и снятие лишних скобок в операциях одного порядка старшинства.

Система АНАЛИТИК разрабатывалась как системная программа для системы коллективного пользования АИСТ-0. Она предназначалась только для выполнения аналитических преобразований в режиме диалога. Основными объектами как входного, так и выходного языка являлись формулы и функции. Набор операторов языка позволял выполнять такие действия, как подстановку выражения вместо переменной, замену подвыражения на выражение (в том числе и функциональную замену), упрощение и аналитическое дифференцирование. Упрощение включало в себя приведение подобных, сокращение дроби, вынесение за скобки общего множителя и выполнение некоторых других действий. Кроме того, операторы языка позволяли создавать и использовать индивидуальный архив.

В области машинного выполнения аналитических преобразований специальное внимание уделялось проблеме выбора формы представления выражений, в которой преобразования выполняются наиболее эффективно.

В ряде работ исследовалась эффективность использования нескольких широко распространенных форм представления выражений, и были получены некоторые оценки по преимущественному выбору тех или иных представлений для определенных классов алгоритмов.

В системе АНАЛИТИК преобразования проводились над выражениями, представленными схемами Канторовича. Схемы были определены и рассмотрены Л.В. Канторовичем в 1957г. [4] и в последующем достаточно широко использовались многими исследователями. Можно отметить три витка использования схем для систем компьютерной алгебры. Первый виток охватил использование схем еще на ламповых ЭВМ. Вторым витком явилось широкое использование схем для проведения аналитических вычислений на ЭВМ среднего поколения. И наконец, на третьем витке схемы продолжают использоваться при построении СКА на современным ЭВМ.

Для ряда задач из теории дифференциальных уравнений были разработаны специальные процессоры КИНО и ПАССИВ. Процессор КИНО выполнял построение определяющих уравнений по заданной системе дифференциальных уравнений, а процессор ПАССИВ приводил уравнения к пассивной форме, т.е. проверял условия совместности.

В обоих процессорах проводились такие аналитические преобразования, как дифференцирование, раскрытие скобок и приведение подобных. Во всех программах аналитических преобразований особое внимание было обращено на внутреннюю форму представления выражений. Основным технологическим инструментом для разработки этих систем была система ЭПСИЛОН.

ЗАКЛЮЧЕНИЕ

Мы должны знать свою историю. Идеи и результаты ранней эпохи развития программирования нашли свое выражение и подтверждение во многих проектах сегодняшнего дня. Знание подходов и методов, прочно вошедших в практику современного программирования, является крепкой основой будущих исследований и практических разработок.

Работа поддержана грантом РГНФ 00-03-00277.

СПИСОК ЛИТЕРАТУРЫ

1. Ершов А.П., Вишневский Ю.Л., Кожухин Г.И., Макаров Г.П. Экспериментальная система коллективного пользования АИСТ-0 // Тр. 2-й Всесоюз. конф. по программированию: Заседание Н. — Новосибирск: ВЦ СОАН СССР, 1970. — С.3–14.
2. Бежанова М.М., Катков В.Л., Поттосин И.В. Работы по аналитическим преобразованиям в ВЦ СОАН СССР // Вычислительная математика и вычислительная техника. Вып. 3. — Харьков, 1972. — С.18–20.
3. Калинина Н.А. Системная программа АНАЛИТИК. Работы по аналитическим преобразованиям в ВЦ СОАН СССР // Там же. — С.33–37.
4. Канторович Л.В. Об одной математической символике, удобной для проведения вычислений на машине // Докл. АН СССР. — 1957. — Т. 113. — № 4.

Н. А. Черемных

АРХИВ АКАДЕМИКА А. П. ЕРШОВА

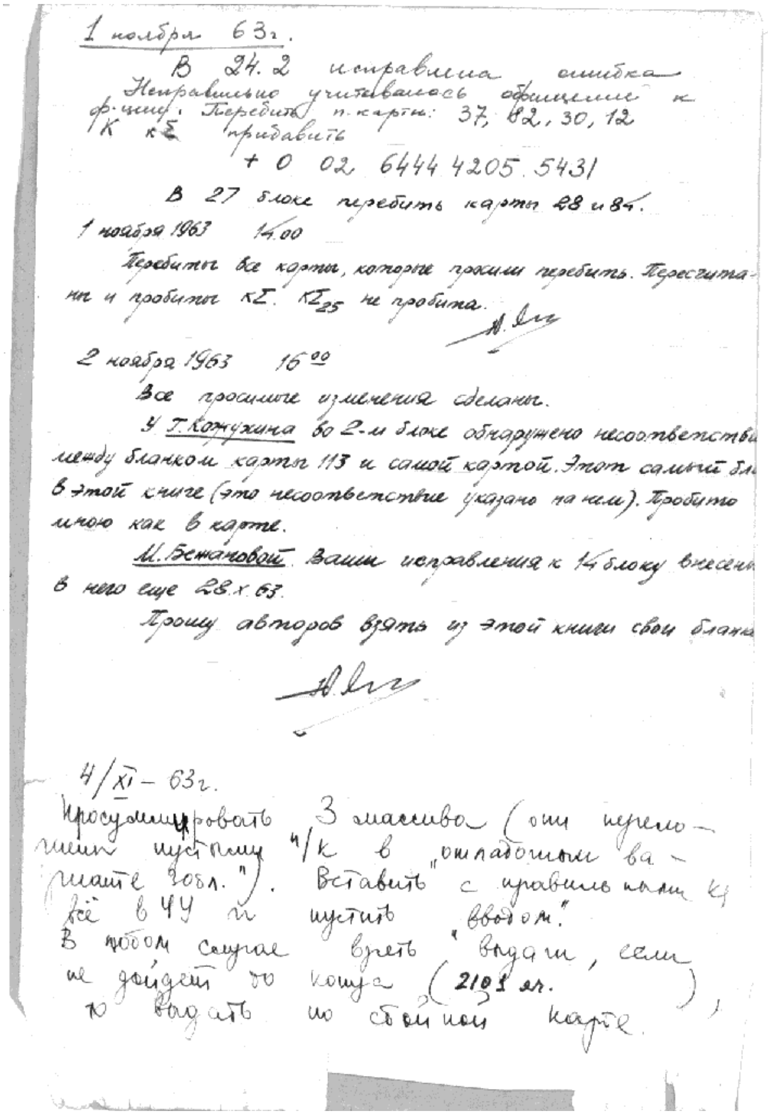
После безвременной кончины Андрея Петровича Ершова остался уникальный архив, состоящий из 500 с лишним толстых канцелярских папок, в которых хранятся документы, отражающие весь его жизненный путь, а вместе с ним — историю развития информатики в России. Каждая папка была собрана Андреем Петровичем либо по хронологическому признаку, либо по тематике.

В самых первых папках хранятся конспекты лекций выдающихся ученых, преподававших в МГУ во время его учебы, черновики курсовых и дипломной работы, материалы к первым научным публикациям.

После аспирантуры А.П.Ершов некоторое время работал в Институте точной механики и вычислительной техники, где создавались первые отечественные ЭВМ, затем — в Вычислительном центре АН СССР. В архиве можно найти служебные записки, написанные рукой Андрея Петровича, но от имени академиков А.А. Дородницына и С.Л. Соболева.

Далее собраны документы, касающиеся переезда в Новосибирск и создания Отдела программирования в Институте математики СО АН СССР.

Уникальны документы, связанные с проектом Альфа. Этот оптимизирующий транслятор с языка Альфа (отечественного варианта Алгола 60) для вычислительной машины М-20 стал первой крупной разработкой Отдела программирования. В архиве сохранились толстые канцелярские книги, в которых отражен весь процесс создания Альфа-транслятора. Они так и назывались: «Рождение α-транслятора», «Детство α-транслятора», «Отрочество α-транслятора». Записи в книгах велись практически ежедневно. Сначала в них отражался ход работы над проектом, фиксировались результаты обсуждений и формулировались проблемы (рис. 1), затем, когда началась отладка, записи в журнале стали играть еще более важную роль. Разработчики кратко суммировали результаты прогона программ, сообщали об ошибках, передавали информацию коллегам, сменявшим их в машинном зале.



1 ноября 63г.

В 24.2 исправлена ошибка

Неправильно учитывалось официнение к ф-ции. Перебить п. карты: 37, 82, 30, 12

К кΣ прибавить

+ 0 02 6444 4205 5431

В 27 блоке перебить карты 28 и 84.

1 ноября 1963 14.00

Перебиты все карты, которые просили перебить. Пересчитаны и пробиты кΣ. $к\Sigma_{25}$ не пробита.

2 ноября 1963 16.00

Все просимые изменения сделаны.

У Т. Кожухина во 2-м блоке обнаружено несоответстви[е] между бланком карты 113 и самой картой. Этот самый бл[анк] в этой книге (это несоответствие указано на нем). Пробито мною как в карте.

М. Бежановой. Ваши исправления к 14 блоку внесены в него еще 28.х.63.

Прошу авторов взять из этой книги свои блан[ки]

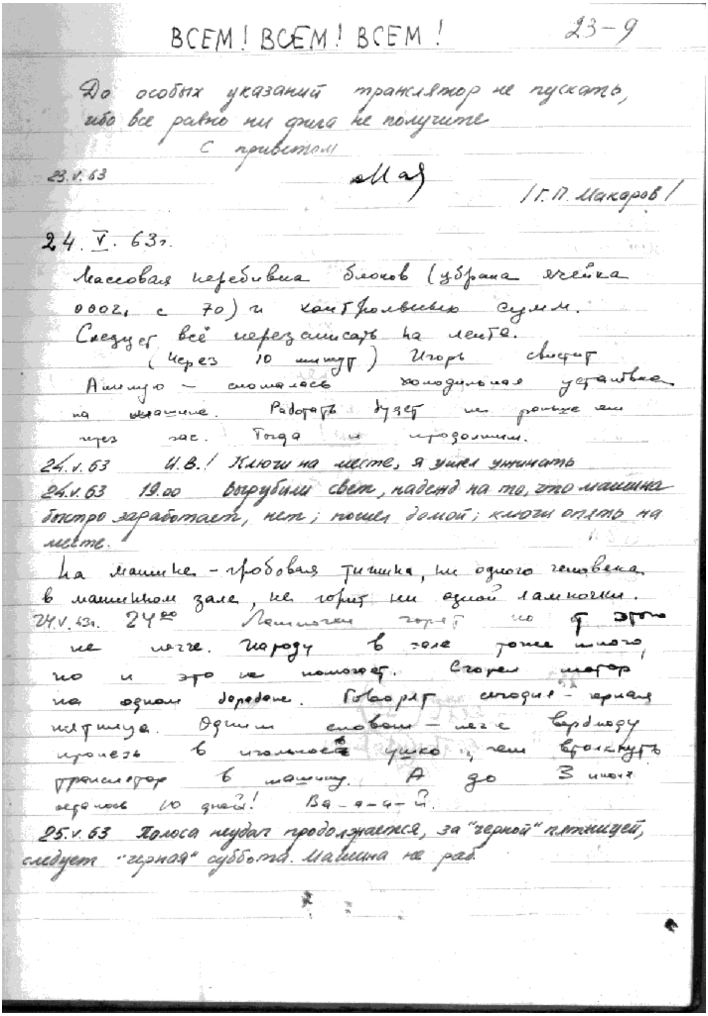
4/XI – 63г.

Просуммировать 3 массива (они переполнили пустоту ч/к в „отладочном варианте 20бл."). Вставить" с правильными к всё в ЧУ и пустить „вводом".

В любом случае взять „выдачи, если не дойдет до конца (2103 яч.), то выдать по сбойной карте.

Рис. 1

Нынешнему поколению программистов практически не знакомы проблемы, волновавшие их коллег в начале 60-х: нестабильно работающее оборудование или нехватка машинного времени, но в то время на их решение тратилась львиная доля времени и сил. Обо все этом подробно рассказывается в α-журналах. Приведем для примера одну страницу (рис. 2).



ВСЕМ! ВСЕМ! ВСЕМ! 23-9

До особых указаний транслятор не пускать, ибо все равно ни фига не получите

С приветом

23.V.63 /Г. П. Макаров/

24.V.63 г.

Массовая перебивка блоков (убрана ячейка 0002, с 70) и контрольных сумм. Следует всё переписать на ленте.

(через 10 минут) Игорь сказал Андрею — сломалась холодильная установка на машине. Работать будет не раньше чем через час. Тогда и продолжим.

24.V.63 И.В.! Ключи на месте, я ушел ужинать

24.V.63 19.00 вырубили свет, надеясь на то, что машина быстро заработает, нет; пошел домой; ключи опять на месте.

На машине — гробовая тишина, ни одного человека в машинном зале, не горит ни одной лампочки.

24.V.63 г. 24.00 Лампочки горят но от этого не легче. Народу в зале тоже много, но и это не поможет. Сгорел мотор на одном барабане. Говорят сегодня — черная пятница. Одним словом — легче верблюду пролезть в игольное ушко, чем впихнуть транслятор в машину. А до 3 июня осталось 10 дней! Ва-а-а-й.

25.V.63 Полоса неудач продолжается, за "черной" пятницей, следует "черная" суббота. Машина не раб.

Рис. 2

Начинается она обращением:

> «ВСЕМ! ВСЕМ! ВСЕМ!
> До особых указаний транслятор не пускать, ибо все равно ни фига не получите. 23.5.63. Г.П.Макаров[1]
> 24.5.63 г.
> Массовая перебивка блоков (убрана ячейка 0002. с 70) и контрольных сумм. Следует все перезаписать на ленте.[2]
> (через 10 минут) Игорь[3] свистит Аллилую – сломалась холодильная установка на машине. Работать будет не раньше, чем через час. Тогда и продолжим.[4]
> 24.5.63.
> И.В.! Ключи на месте, я ушел ужинать.
> 24.5.63. 19.00
> Вырубили свет, надежд на то, что машина быстро заработает, нет; пошел домой, ключи опять на месте.[5]
> На машине – гробовая тишина, ни одного человека в машинном зале, не горит ни одной лампочки.
> 24.5.63 г. 24.00. Лампочки горят, но от этого не легче. Народу в зале тоже много, но и это не помогает. Сгорел мотор на одном барабане. Говорят, сегодня – черная пятница. Одним словом, легче верблюду пролезть в игольное ушко, чем втолкнуть транслятор в машину. А до 3 июня осталось 10 дней! Ва–а-а-й.[6]

Сотрудники Отдела программирования в то время были очень молоды, стоически переносили все трудности и не теряли чувства юмора. Очередная запись в журнале: «31.5. 0 час. 15 мин. Вечер популярной песни». Видимо, машина стояла, домой уйти нельзя (вдруг начнет работать!), и программисты коротали время, придумывая новые слова на известную мелодию, а затем сочиняли басню (рис. 3–4).

[1] Геннадий Павлович Макаров в то время был главным инженером на машине М-20.
[2] Этот текст написан рукой И.В. Поттосина.
[3] По всей видимости, Игорь Васильевич Поттосин.
[4] Этот абзац принадлежит А.П. Ершову.
[5] Две записи подряд сделаны Г.П. Макаровым.
[6] Написано рукой И.В. Поттосина. Видимо, он до полуночи напрасно ждал, что машина заработает, и оставил в журнале «крик души».

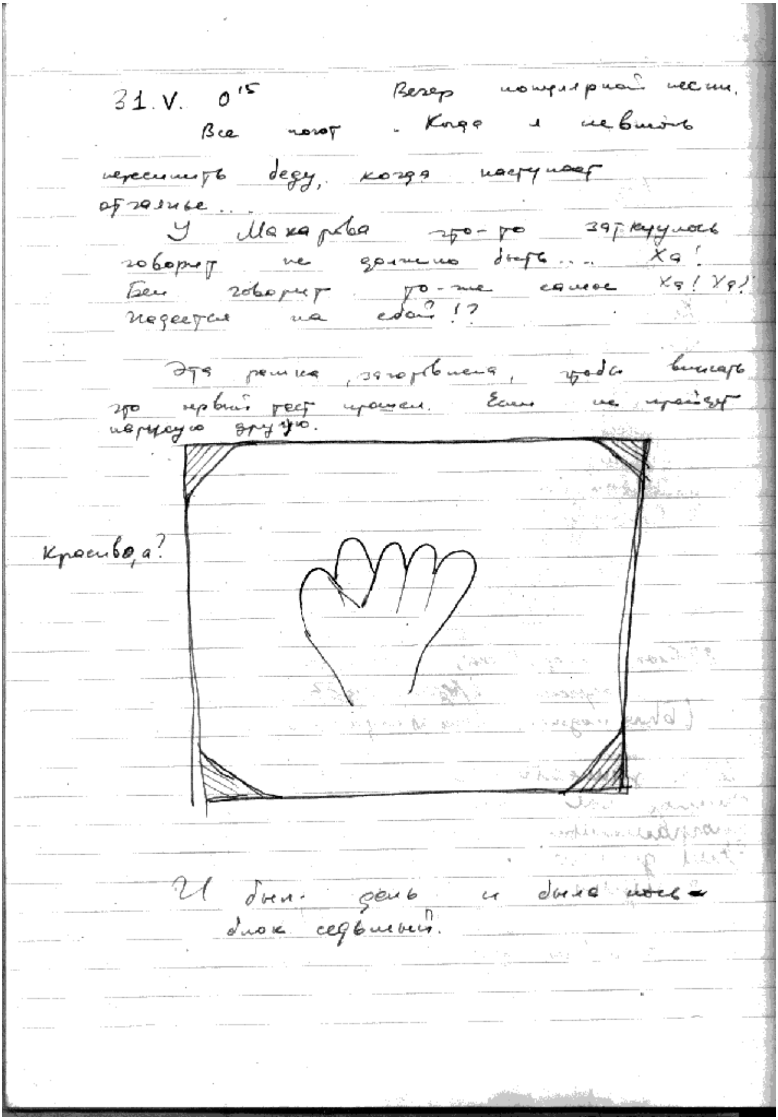
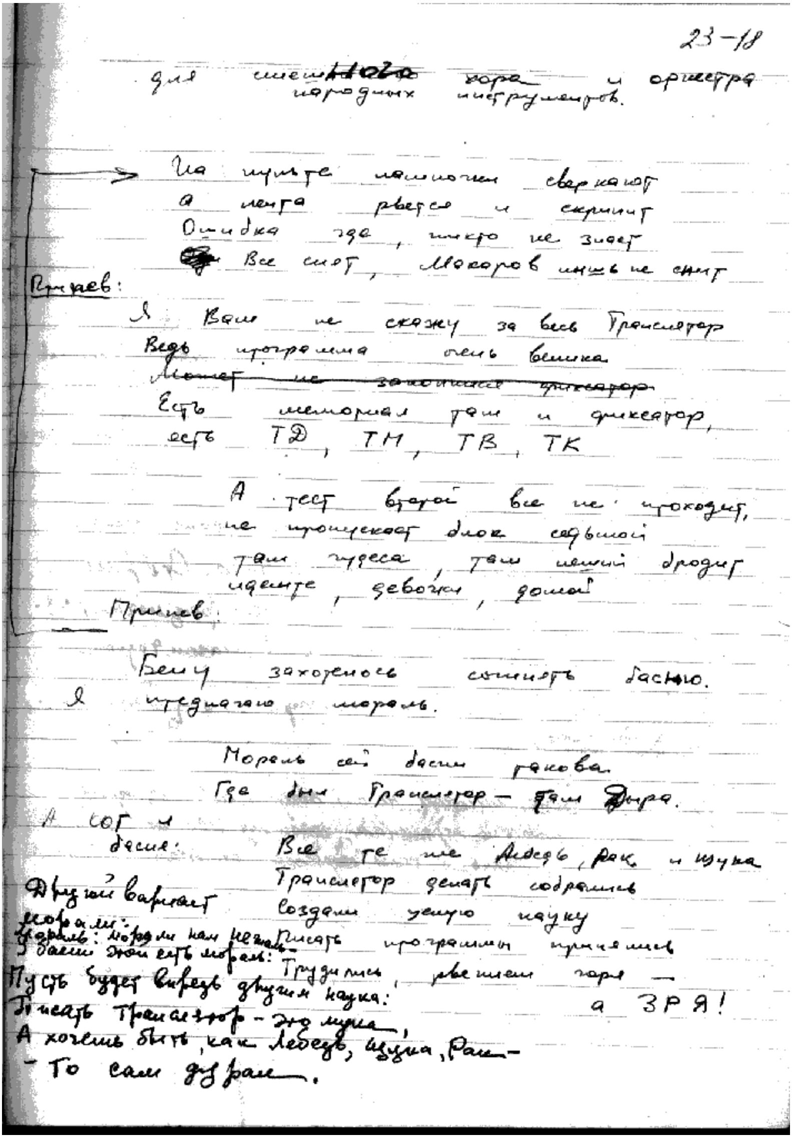

31.V. 0^15 Вечер популярной песни.
Все поют. Когда я не выйду
пересмотреть деду, когда наступает
отчаяние...
У Макарова что-то застукулось
говорит не должно быть... Ха!
Все говорят то-же самое Ха! Ха!
Надеется на себя!?
Эта решка, заговоренная, чтобы выкарабкать
что первый тест прошел. Если не пройдет
переписую другую.
Красиво, а?
И был день и была ночь –
блок седьмой.

Рис. 3


23-18

для смешанного хора и оркестра народных инструментов.

На пульте лампочки сверкают
А лента рвется и скрипит
Ошибка где, никто не знает
Все спят, Макаров лишь не спит

Ершов:

Я Вам не скажу за весь Транслятор
Ведь программа очень велика
~~Может не закончиться фиксатор~~
Есть мемориал там и фиксатор,
есть ТД, ТМ, ТВ, ТК

А тест второй все не проходит,
не пропускает блок седьмой
там чудеса, там леший бродит
идите, девочки, домой!

Пришев.

Всем захотелось сочинить басню.
Я предлагаю мораль.

Мораль сей басни такова.
Где был Транслятор — там дыра.

А вот и басня:

Все те же Лебедь, Рак и Щука
Транслятор делать собрались
Создали целую науку
Писать программы принялись
Трудились, рвались горя —
а ЗРЯ!

Другой вариант морали:
Мораль: морали нам не надо
У басни этой есть мораль:
Пусть будет вредь другим наука:
Писать Транслятор — это мука,
А хочешь быть, как Лебедь, Щука, Рак —
То сам дурак.

Рис. 4

Архив отражает огромную научно-организационную деятельность А.П. Ершова, который в разные годы жизни был председателем Научного Совета по комплексной проблеме «Кибернетика» АН СССР, руководил Комиссией по системному математическому обеспечению Координационного комитета по вычислительной технике АН СССР, Междуведомственной научно-технической комиссией по программному обеспечению ЭВМ при ГКНТ СССР и другими комитетами и комиссиями, во многом определявшими пути развития системного и теоретического программирования и всей вычислительной техники в нашей стране.

Все многочисленные зарубежные поездки А. П. Ершова отражены в отдельных папках. Поскольку он принимал самое активное участие в подготовке множества международных конференций и конгрессов, в архиве можно найти материалы его выездных дел, программы конференций и семинаров, переписку, касающуюся организации этих мероприятий, научные отчеты по командировкам, а кроме того — билеты, квитанции, записки для памяти — все эти материальные свидетельства давних событий, проходивших в самых разных точках земного шара.

Академик Ершов был редактором или членом редколлегии как русских журналов «Микропроцессорные средства и системы», «Кибернетика», «Программирование», так и международных — Acta Informatica, Information Processing Letters, Theoretical Computer Science. В архиве — обширная переписка с редакциями и читателями этих изданий.

Андрею Петровичу писали самые разные люди, многим он отвечал прямо или косвенно. Помимо редакционной переписки, обширный массив составляет научное эпистолярное наследие академика. Очевидно, нет ни одного известного в области информатики ученого, с которым бы А.П. Ершов не состоял в переписке. Сохранились материалы публичных выступлений Ершова, во время которых он вел дискуссии с аудиторией. Записки слушателей свидетельствуют о живом интересе публики к новому научному направлению — созданию искусственного интеллекта. Очень интересен процесс подготовки Международного симпозиума «Алгоритм в современной математике и ее приложениях», отраженный в материалах архива. Симпозиум в Ургенче был задуман как научное паломничество на родину великого средневекового математика Аль-Хорезми, которому современная наука обязана самим появлением понятия «алгоритм». Сохранилась переписка А.П. Ершова с участниками, среди которых было множество замечательных имен, варианты программы симпозиума, доклады, на нем представленные.

Подготовка и проведение крупнейших советских программистских мероприятий — а не было, наверное, за годы его активной научной деятельности, ни одного, в котором А.П. Ершов так или иначе не участвовал бы, — также представлены многочисленными интересными документами. Материалы 2-й Всесоюзной конференции по программированию (ВКП-2), которая проходила в Новосибирске в 1970 г., говорят об огромном интересе к данной проблеме самых широких слоев советского общества. Архив содержит анкеты более 800 участников конференции и сотни заявок, многие из которых устроители мероприятия не смогли удовлетворить.

Истории создания Конструкторского бюро системного программирования и Новосибирского филиала Института точной механики и вычислительной техники посвящено несколько папок, где можно найти переписку с соответствующими ведомствами, планы работ, персоналии и пр.

Значительная часть архива объединена проблемой информатизации народного образования в СССР. А.П. Ершов одним из первых в нашей стране осознал наступление эпохи компьютеров. Ему принадлежит известный лозунг «Программирование — вторая грамотность», и он приложил массу сил к тому, чтобы этот лозунг не остался на бумаге, а обучение основам программирования вошло в учебные планы наших школ.

А.П. Ершов был создателем первого учебника по информатике для средних школ и методического пособия для учителей. Все огромная работа по их созданию отражена в документах архива. Кроме того, здесь же можно найти материалы летних школ юных программистов, которые проводились в Академгородке начиная с 1977 года и задачи для олимпиад по программированию.

Отдельный интерес представляют документы, касающиеся повседневной жизни научного коллектива, возглавлявшегося А.П. Ершовым. Это служебные записки, выписки из решений Ученого совета ВЦ, планы работ (по годам и на пятилетки). В папках сохранились списки сотрудников, направлявшихся «на картошку», и объяснительные записки опоздавших на работу, письма в местный комитет с просьбой предоставить квартиру или место в детском саду для ребенка сотрудника отдела. В целом эти документы дают уникальную картину жизни советских ученых в 60–80-х годах прошлого столетия.

Многочисленные папки посвящены собственно научной работе. В архиве сохранились рукописи практически всех статей и монографий Ершова. Не только историки науки, но и активно действующие ученые смогут найти в них «информацию к размышлению» и, может быть, почерпнут в старых,

но не устаревших работах новые идеи или получат импульс к их появлению.

Бумажный архив, в том виде, как он собирался самим академиком Ершовым, хранится в толстых канцелярских папках и, в основном, систематизирован тематически. Внутри папки документы, как правило, расположены по хронологии.

Приступая к созданию электронного архива, мы решили следовать подходу А.П. Ершова и, размещая документы на сайте, группировать их по темам, отражающим основные вехи научного и жизненного пути А.П. Ершова, а внутри каждой темы выделяли группы и подгруппы документов. Таким образом, на первой странице архива лежит перечень основных тем, а спускаясь по дереву вниз, читатель в конце концов видит документ, который может состоять из одной или множества страниц. Кратким комментарием снабжена каждая тема, группа или подгруппа, а также сам документ.

Документы архива — это рукописные или печатные тексты статей, письма, написанные А.П. Ершовым или им полученные, телеграммы, служебные записки, билеты, квитанции, фотографии, заметки для памяти и пр. В комментариях к каждому документу указывается его автор, адресат, если это письмо или телеграмма, персоналии — люди, которые упоминаются в этом документе. Поскольку в архиве хранится множество документов практически на всех европейских языках, указывается оригинальный язык документа. Выбирая мышью пиктограммы страниц, можно увидеть на экране изображение самого документа. Стандартные кнопки навигации позволяют переходить от страницы к странице, возвращаться от документов к группам или главным темам. Следует отметить, что значительное количество документов архива — это третий или даже пятый экземпляр, напечатанный под копирку на плохой пишущей машинке, текст, написанный от руки или карандашом, пожелтевшие страницы старых газет и т.п. Читать такие документы с экрана достаточно трудно, поэтому, наряду с изображениями, приводятся их текстовые представления, иногда — на двух языках.

Кроме тематической организации архива, когда к одной теме относились документы, хранящиеся в нескольких разных папках, мы сохранили возможность как бы «пролистать» архивные папки одна за одной, рассматривая в них каждую страницу и не отвлекаясь на наш комментарий. Для этого внизу каждого экрана есть кнопка «Папки». Выбрав ее мышью, можно пройти по всем папкам, уже включенным в электронный архив.

В настоящее время отсканирована и размещена на сайте примерно половина всех документов. Работа продолжается. Познакомиться с электронным архивом можно на сайте `http://ershov.iis.nsk.su/russian/`

Н. В. Шилов, Е. К. Шилова

ИСТОРИЯ ЯЗЫКА REAL

КАК НАЙТИ МОНЕТКУ?

1. ТРУДНАЯ ЗАДАЧА

Математику уж затем учить надо,
что она ум в порядок приводит.
М.В. Ломоносов

Однажды, во время подготовки задач областной олимпиады по математике для школьников 6–8 классов, один из членов жюри предложил для обсуждения следующую головоломку.

> Дано 15 внешне одинаковых монет, среди которых одна — фальшивая, а остальные — настоящие. Все настоящие монеты имеют равный вес, а фальшивая монета имеет другой вес. Кроме того, есть одна гирька, равная по весу настоящей монете. Возможно ли найти фальшивую монету за (не более чем) 3 взвешивания на чашечных весах?

«Эта головоломка позволит сразу выделить лидера олимпиады», — аргументировал он своё предложение.

Возникла кратковременная пауза: все члены жюри задумались над решением предложенной задачи о фальшивой монетке. Молчание прервал другой член жюри: «Да-а, это интересная и трудная задача… Я пока не знаю, как её решить…” “По правде сказать, я тоже не знаю, как её решить”, — неожиданно “парировал” член жюри, предложивший задачу. “Ну, тогда и нечего обсуждать её”, — подвёл итог короткого обсуждения Председатель жюри.

Однако вердикт Председателя жюри ещё не означает окончания этой абсолютно правдивой истории. Дело в том, что среди членов этого жюри был один программист-теоретик. Он тоже не смог решить задачу сразу во время заседания, но решил разобраться с ней на досуге. Весь день после заседания он то и дело возвращался к анализу этой задачи. Увы, безрезуль-

татно! Ночью монетки, взвешивания, весы стали его настоящим кошмаром…

А наутро теоретик решил действовать. Он наметил два подхода к поиску решения головоломки. Первый подход можно условно назвать человеко-ориентированным, а второй — машинно-ориентированным. Сначала обсудим человеко-ориентированный подход, а в следующем разделе обсудим машинно-ориентированный подход.

Человеко-ориентированный подход к решению головоломки был задуман чрезвычайно просто. По совместительству теоретик работал преподавателем математического факультета в Новосибирском Государственном Университете. Поэтому на первом же занятии на следующий день со студентами он предложил эту головоломку студентам. Им был установлен и приз за первое решение: ксерокопия 100$ США.

Человеко-ориентированный подход быстро дал положительный результат, хотя теоретику не пришлось делать ксерокопию 100$, так как первой решила головоломку его жена, которая тут же отказалась от приза. А вот машинно-ориентированный подход к олимпиадной головоломке потребовал дополнительных размышлений.

Во-первых, эту головоломку естественно считать частным случаем следующей параметрической головоломки.

> Дано N внешне одинаковых монет, среди которых одна — фальшивая, а остальные — настоящие. Все настоящие монеты имеют равный вес, а фальшивая монета имеет другой вес. Кроме того, есть одна гирька, равная по весу настоящей монете. Возможно ли найти фальшивую монету за (не более чем) K взвешиваний на чашечных весах?

Во-вторых, для поиска подхода к решению параметрической головоломки стоит разобрать какой-либо простой частный случай, когда ответить на вопрос о возможности найти фальшивую монетку значительно проще, чем в олимпиадной головоломке. Давайте, например, разберём случай $M = 5$ и $K = 2$, когда надо найти одну фальшивую среди 5 монет за 2 взвешивания с помощью дополнительной гирьки, вес которой равен весу настоящей монеты.

В этом случае фальшивая монета может иметь любой номер из 1, 2, 3, 4 или 5, а вес её может быть как легче (л), так и тяжелее (т), чем у настоящей монеты. Значит, всего есть $5 \times 2 = 10$ различных вариантов (номер, вес)

фальшивой монеты. Ниже приведено 2 варианта фальшивой монеты и поиска этой монетки.

вариант	человек	весы	человек	весы	результат
(1, л)	: (1+2) ? (3+г)	<	(1) ? (2)	< :	1
(4, т)	: (1+2) ? (3+г)	=	("4) ? (г)	> :	4

Наблюдение, которое можно сделать, рассматривая эту таблицу, следующее: процесс поиска фальшивой монетки очень напоминает игру двух партнёров — один пытается найти фальшивую монету (назовём его Догадой), а другой — осуществляет взвешивание монет на весах (назовём его Весовщиком). Ходы Догады состоят в выборе монет для взвешивания, а ходы Весовщика — это результаты взвешиваний монет. Давайте назовём описанную выше игру игрой в монетки (но не на деньги!).

Раунд — это пара последовательных ходов партнеров в игре. Тогда параметрическую и олимпиадную головоломку можно переформулировать следующим образом:

- Существует ли в игре в N монеток выигрышная стратегия из K раундов?
- Существует ли в игре в 15 монеток выигрышная стратегия из 3 раундов?

2. НА ПОМОЩЬ ПРИХОДИТ КОМПЬЮТЕРНАЯ НАУКА

Компьютерная наука занимается изучением вычислительных машин
не более чем астрономия занимается изучением телескопов
Э. Дейкстра

Игровая модель головоломок очень понравилась программисту-теоретику. Но ещё необходимо было разобраться, как формально выразить неформальное понятие выигрышной стратегии.

Вот тут-то и пришла на помощь современная компьютерная наука. Говоря о вкладе компьютерной науки в сокровищницу знаний человечества, во-первых, необходимо назвать языки программирования, т.е. языки явного описания в каком порядке производить преобразования данных. Но, во-вторых, необходимо сказать о языках спецификаций, которые позволяют описывать свойства данных и вычислений без явного упоминания порядка преобразований и конкретных значений данных. Примеры языков программирования известны многим людям, начиная со школы: популярные

Бэйсик и Фортран, Си и Паскаль, Лисп и МЛ, Лого и Пролог. А вот примеры языков спецификаций известны только “широкому кругу узких специалистов”. Однако (будучи вхожим в этот круг) программист-теоретик использовал один из таких языков спецификаций для машинно-ориентированного решения параметрической головоломки, а именно: для формулировки понятия выигрышной стратегии была использована стратегия Пропозициональной Динамической Логики (ПДЛ).

Как и русский язык, язык ПДЛ состоит из слов, а из слов уже строятся предложения ПДЛ. Слово ПДЛ может быть действием или событием. Действия изменяют значения данных (например, позиции в игре), приводят к переходу от одних значений к другим (поэтому логика называется “динамической”). События не могут изменять значения данных, но для одних значений событие имеет место, а для других — нет. Предложения в языке ПДЛ часто называют спецификациями, высказываниями или пропозициями (поэтому и логика называется “пропозициональной”).

ПДЛ позволяет описывать формально, или (как это принято говорить в “широком кругу узких специалистов”) специфицировать, те позиции игры в монетки, где у Догады есть выигрышная стратегия в 1 раунд, в 2 раунда, в 3 раунда, и, вообще, для любого значения параметра K, где у Догады есть выигрышная стратегия в K раундов.

Проверка значений данных в модели, которые удовлетворяют спецификации, называется проверкой модели. Методы проверки моделей отличаются друг от друга тем, с какими моделями может работать тот или иной метод, какие языки спецификаций поддерживают, и, наконец, по своей эффективности. По-видимому, первыми, кто понял их прикладное значение, были американские исследователи Эдмунд Кларк и Алан Эмерсон и французский учёный Джозеф Сифакис. Произошло это около 20 лет назад в начале 1980-ых годов. Эти методы сразу стали применяться для анализа программного обеспечения и компьютерного оборудования. За прошедшие годы было разработано и реализовано на вычислительных машинах несколько десятков таких методов. Может быть, самая популярная реализация метода проверки моделей — это пакет SMV. Этот пакет был создан американским учёным Кеном Макмилланом. Сейчас свободно распространяются версии этого пакета для работы под Windows и Unix. Они доступны в Интернете.

Вот теперь уже всё готово для машинно-ориентированного решения олимпиадной головоломки: модель, спецификация и пакет проверки моделей SMV. Программисту-теоретику оставалось только скачать пакет SMV через Интернет, установить его на своём компьютере, запрограммировать в

терминах, понятных SMV, игру в 15 монеток и ПДЛ-спецификацию позиций, где у Догады есть выигрышная стратегия в три раунда.

3. РАСПРЕДЕЛЁННЫЕ СИСТЕМЫ

Здесь можно было бы закончить статью на мажорной ноте: олимпиадная головоломка решена! Да, решена, только программист-теоретик не использовал пакет SMV, а пошёл другим путём. Он использовал небольшую программу проверки моделей, написанную его коллегами для совместных исследований по проверке свойств протоколов на моделях распределённых систем. Вот об этих исследованиях и пойдёт речь в оставшейся части статьи…

Распределенная система — это несколько устройств, время от времени обменивающихся информационными сообщениями друг с другом. Так, например, распределённую систему образуют два компьютера, один из которых (клиент) посылает напарнику запросы на выполнения определённых действий, а второй (сервер) — информирует своего напарника о результатах этих действий; распределённую систему такого типа часто так и называют “системой клиент—сервер”. Глобальная распределенная система — это Интернет, объединяющий миллионы компьютеров-серверов и миллиарды компьютеров-клиентов.

Но не следует думать, что устройства распределённой системы обязательно должны быть вычислительными машинами. Распределённая система может включать как вычислительные машины, так и живых людей. Именно поэтому лучше говорить не об устройствах, а о взаимодействующих агентах. Например, в игре в монетки Догада и Весовщик — два агента, которые образуют распределённую систему типа клиент—сервер: Догада — это клиент, а Весовщик — это сервер; клиент запрашивает сервер произвести взвешивание, а сервер его производит и информирует клиента о результатах.

Распределённые системы всё чаще встречаются в современном мире, а их надёжность и свойства протокола становятся критическим моментом социальной, экономической и военной стабильности. Например, если протокол обмена сообщениями через Интернет между клиентами-покупателями и сервером е-магазина не обеспечивает секретности индивидуальных банковских pin-кодов, то такое положение дел чревато как экономическим ущербом для покупателей, так и банкротством е-магазина.

Осознавая значение надёжности распределённых систем, Международный Телекоммуникационный Союз ITU (International Telecommunication Union) в период 1976–2000 гг. разработал специальный стандарт Z100 для проектирования распределённых систем. Этот стандарт получил название SDL (Specification and Design Language). По-русски его называют поразному, например, языком формальных описаний. Нам кажется, что гораздо лучше перевести его по-другому: “specification” — “техническое задание”, “design” — “эскизное проектирование”; получается “язык технического задания и эскизного проектирования” или, кратко, “язык Тех-Про”.

Язык SDL имеет мощные средства описания структуры распределённых систем, индивидуальных агентов, возможных сообщений и потоков сообщений, посылаемых от агента к агенту. Этот язык имеет математически точные правила, которые позволяют (в принципе) реализовать все его конструкции на современных языках программирования. Эти правила называются операционной семантикой языка SDL. Но плата за выразительную силу языка — сложность операционной семантики для понимания и реализации. Достаточно сказать, что только описание правил семантики занимает более 500 страниц.

Ещё один недостаток языка SDL — это отсутствие средств описания свойств. Другими словами, нет стандартного языка спецификаций для систем и данных, описанных на языке SDL. В результате получается, что у нас есть возможность спроектировать на этом языке систему покупки товаров через Интернет, но нет возможности специфицировать свойство такое, что протокол обмена сообщениями между покупателями и е-магазином обеспечивает тайну индивидуальных банковских pin-кодов. А раз нет возможности специфицировать свойства, то у нас нет возможности при помощи вычислительных машин их проверять.

4. ОСОБЕННОСТИ НАЦИОНАЛЬНОЙ НАУКИ

Вернёмся к исследованиям по проверке свойств протоколов на моделях распределённых систем, в которых участвовал программист-теоретик вместе со своими коллегами по лаборатории Теоретического программирования в Институте систем информатики Сибирского Отделения Российской Академии Наук (ИСИ СО РАН) ещё с конца 1980-х годов. Ещё существовал Советский Союз, ещё выдвигались амбициозные национальные проекты. Один из этих проектов — создание национальной телекоммуникационной сети нового поколения.

Одним из основополагающих принципов этого проекта было использование языка SDL на всех этапах: от разработки требований до программной реализации. Такое массовое использование языка SDL предполагало разработку новых эффективных машинно-ориентированных методов исследования свойств распределённых систем. Группа научных сотрудников лаборатории Теоретического программирования ИСИ СО РАН, участвовавшая в этом национальном проекте, пришла к выводу, что самое реальное — проверять на моделях свойства протоколов для разрабатываемой национальной телекоммуникационной сети.

Жёсткая привязка всего телекоммуникационного проекта к языку SDL диктовала жёсткое условие: язык описания моделей распределённых систем должен быть настолько близок к языку SDL, чтобы быть понятным инженерам-разработчикам самой системы. Кроме этого языка описания моделей, должен быть "естественный" язык спецификации свойств этих моделей, который не требует дополнительно упрощать модели распределённых систем. И, наконец, и язык описания моделей, и язык спецификации свойств этих моделей должны быть поддержаны эффективными методиками проверки этих свойств в этих моделях.

После нескольких лет поисков, в 1992 г. научной общественности был представлен проект языка REAL. Этот язык состоит из двух равноправных частей: исполняемых спецификаций и логических спецификаций. Исполняемые спецификации языка REAL очень близки к языку SDL и служат для создания моделей распределённых систем. А логические спецификации языка REAL — это вариант Пропозициональной Динамической Логики (ПДЛ), обогащенный некоторыми дополнительными средствами, и служат они для спецификации свойств моделей. В 1992 г. язык ещё не имел формальной операционной семантики, пригодной для реализации на компьютере, но был снабжён неформальным описанием.

Отличительная особенность языка REAL — это широкие возможности работы со временем, каких нет в SDL. Модель времени, принятая в языке, позволяет так реалистично описывать течение времени у разных агентов в распределённых системах, что авторы языка единодушно выбрали название «Реал». Это русское название превратилось в английское «REAL», так как впервые язык был представлен на Международной конференции "Перспективы системной информатики", рабочий язык которой был английский.

Первый успех позволял двигаться дальше: на очереди стояла задача формальной операционной семантики языка REAL. Но пути возникли трудности совсем ненаучного характера: в 1992 г. уже не было Советского Союза, и прекратились скоординированные работы по проекту создания

национальной телекоммуникационной сети нового поколения. Одни научные и инженерные коллективы, ранее занятые в этом проекте, вообще прекратили своё существование, другие — сменили тему исследований, а некоторые — попытались продолжить начатые исследования по проекту с надеждой на свои силы и с верой в перспективность своего подхода. Именно так поступила группа авторов языка REAL.

В 1992-94 гг. участникам этой группы приходилось работать на энтузиазме: государственного заказа на исследования по проверке свойств моделей распределённых систем уже не было, негосударственных научных фондов для поддержки фундаментальных исследований ещё не было. Рассказывают, что в конце 1920-х годов участники Группы Изучения Реактивного Движения (ГИРД), создавшие первые советские ракеты, в шутку расшифровывали аббревиатуру ГИРД так: Группа Инженеров, Работающих Даром. Так вот, в 1992–94 гг. небольшой коллектив, занятый разработкой операционной семантики языка REAL92, можно было назвать ГУРД: Группа Учёных, Работающая Даром.

Несмотря на все трудности (научные и ненаучные), возникшие перед группой учёных, работавших даром, в 1994 г. первая версия операционной семантики языка REAL была представлена специалистам на международном рабочем совещании по моделям параллельных вычислений в Берлине. Отличительная особенность этой семантики — краткость. Достаточно сказать, что формальное и неформальное описание всей семантики занимает всего 25 страниц. (Сравните с сотнями страниц семантики SDL.) В то же время, эта семантика доступна для новичков и реализуема на компьютере.

Научные успехи 1994 г. совпали с началом финансирования исследований по языку REAL Международным Научным Фондом Дж. Сороса (ISF — International Scientific Foundation), Международной Ассоциацией развития сотрудничества с учёными новых независимых государств бывшего СССР (INTAS — INTernational ASsociation for promotion of cooperation with new independent states) и Российским Фондом Фундаментальных Исследований (РФФИ). В конце 1994 г. были получены небольшие и краткосрочные гранты от ISF и INTAS. Но кардинальные изменения произошли в конце 1995 г., когда исследования получили поддержку от INTAS и РФФИ в виде совместного гранта ИНТАС—РФФИ 95-0378 «Методы и средства верификации и анализа распределенных систем» (рук. В. А. Непомнящий) на период 1996–1999 гг.

5. П.Д.Л. = ПОЛНЫЙ. ДЕТСКИЙ. ЛЬГОТНЫЙ

Если на клетке слона написано "Буйвол",
не верь глазам своим.
Козьма Прутков "Мысли и афоризмы"

Итак, к середине 1990-х годов была разработана операционная семантика языка REAL для описания моделей и свойств распределённых систем. Теперь можно было ставить задачу проверки свойств распределённых систем в этих моделях.

Здесь можно было пойти на использование метода "грубой силы": запрограммировать какой-либо универсальный метод проверки моделей, а потом его применять ко всем моделям и всем свойствам автоматически, без вмешательства человека. Но такой подход — чересчур затратный для проверки простых свойств, которые чаще всего возникают на практике. Поэтому возникла идея разобраться с проверкой наиболее часто встречающихся свойств, а дополнительно реализовать какой-либо универсальный метод проверки моделей для остальных свойств.

Что касается универсального метода проверки моделей, то выбор пал на так называемый ускоренный метод проверки моделей, который был теоретически разработан группой зарубежных ученых в 1992 г. Программная реализация этого метода оказалась довольно-таки практичной и в настоящее время широко используется для проведения различных экспериментов в ИСИ СО РАН. В частности, именно эта программа проверки моделей была использована программистом-теоретиком для машинно-ориентированного решения олимпиадной головоломки.

Теперь время рассказать про проверку свойств распределённых систем, которые часто встречаются на практике. Здесь на помощь пришла идея классифицировать эти свойства. Еще в 1980-х годах американские учёные Мани Чанди и Джадев Мишра и израильские ученые Захар Манна и Амир Пнуели разработали основы такой классификации и исследовали приёмы математического доказательства свойств, попадающих в предложенную ими классификацию. Ими было замечено, что очень многие свойства распределённых систем, вызывающие реальный интерес, могут быть отнесены к четырем классам. Вот пример одного из этих классов:

после наступлений события P,
при любом сценарии работы распределённой системы
в некоторый момент времени наступит событие F.

Свойства, которые попали в этот класс, получили условное название свойств прогресса. (Позже мы приведём пример такого свойства, который прояснит это название.)

Для этих классов были разработаны простые шаблоны (схемы) для конструирования доказательств свойств распределённой системы. Шаблоны не зависят от распределённой системы, а только от свойства. Эти схемы позволяют конструировать эскизы доказательств, которые сводят доказательство сложных свойств из этих классов к доказательству большого числа простых свойств, легко выразимых в самых простых терминах Пропозициональной Динамической Логики (ПДЛ). Эскизы доказательств отличаются краткостью и простотой. Они легко могут быть сконструированы вручную одновременно с эскизным проектом распределённой системы, так как свойства всех четырёх классов имеют очень простую структуру. А вот доказательство простых свойств может оказаться непростым делом, так как нет никаких ограничений на события P и F, а они могут использовать очень ёмкие и сложные понятия.

Описанный подход к доказательству свойств распределённых систем был адаптирован для проверки свойств моделей распределённых систем, описанных на языке REAL. Главная идея состояла в том, что простые свойства, которые возникают после применения шаблонов доказательств, не надо доказывать, а надо проверять при помощи простого метода проверки моделей для самых простых формул ПДЛ. Так как этот подход использует классификацию на классы проблем и метод проверки моделей, то за ним закрепилось название “проблемно-ориентированный методики проверки моделей”. Эту методику можно считать основным результатом исследований 1996-1999 гг. по проверке свойств моделей распределённых систем, описанных на языке REAL. (Другие важные результаты, полученные в этот период, — это аниматор исполняемых спецификаций и конвертор SDL в исполняемые спецификации.)

Проблемно-ориентированная методика проверки свойств моделей была опробована на примерах реальных распределённых систем. Самый популярный из них — покупка пассажиром железнодорожного билета пригородного сообщения в кассе-автомате. Мы будем называть его системой «касса—пассажир». Неформальное свойство системы «касса—пассажир», которое мы хотим проверить, состоит в следующем: если пассажир и касса готовы к работе, то при соблюдении правил покупки касса выдаст билет пассажиру до нужной ему станции. Это свойство попадает в класс свойств прогресса и отвечает за благополучное («прогрессивное») развитие событий:

после наступлений события P =
"пассажир и касса готовы к работе",
при любом правильном сценарии работы
распределённой системы "касса-пассажир"
в некоторый момент времени наступит событие F =
"касса выдаст билет пассажиру до нужной станции".

Было рассмотрено несколько моделей этой системы и вариантов спецификации свойства на языке REAL. Причём, новые модели и варианты появлялись в результате исправлений в модели и уточнений спецификаций после применения к ним проблемно-ориентированной методики проверки. Типичные ошибки, которые возникали при последовательном уточнении и исправлении распределённой системы «касса-пассажир» и при формальной спецификации ее свойств, были двух сортов:

- отсутствовала «чистка мусора»;
- не соблюдались временные ограничения.

Пример чистки мусора: после обслуживания пассажира касса сама должна «почистить» необработанные сигналы клавиатуры (например, аннулировать случайное нажатие клавиши какой-либо станции, вернуть лишнюю монету из монетоприёмника и т.п.).

Пример временных ограничений: быстродействие узлов кассы должно обеспечить обработку команд пассажира быстрее, чем время реакции пассажира, а информация для пассажира должна быть доступна достаточное время для принятия решения и реакции (например, касса должна распознать номинал очередной монеты, вычислить остаток суммы и высветить его на индикаторе быстрее, чем пассажир успеет ещё раз взглянуть на индикатор и опустить ещё одну монету).

Остаётся только добавить, что настоящая российская касса-автомат поддерживает несколько больший сервис, чем касса в модельном примере. Во-первых, кроме монет реальная касса принимает к оплате банкноты и пластиковые карточки. Во-вторых, на её клавиатуре есть три «магические» клавиши «П», «Д» и «Л» — «ПДЛ». Но это не означает, что автомат был проверен при помощи Пропозициональной Динамической Логики (ПДЛ). Смысл этих трёх клавиш совсем другой. Он расшифрован тут же на передней панели автомата, где укреплена табличка с правилами пользования: «П», «Д» и «Л» — это 3 вида тарифов, принятые на российской железной дороге, — «Полный», «Детский» и «Льготный». Если на кассе-автомате написано «ПДЛ», не верь глазам своим.

А. А. Берс

ЭЛЕКТРОННАЯ ПОДГОТОВКА ИЗДАНИЙ

Системный анализ и проекты

Годы 1968–70 прошли у Л.Л. Змиевской, А.Ф. Рара и А.А. Берса главным образом «под звездой» Алгола-68, — его изучения, освоения и перевода, включая создание русской версии (что было новинкой), проходившей параллельно с разработкой языка рабочей группой 2.1 ИФИП, в которой активно работал А.П. Ершов.

Уже при издании «Сообщения об Алголе-68» в журнале «Кибернетика» в Киеве (в номерах: *6*, 69г. и *1*, 70г.) мы столкнулись с ужасной бедностью наличной полиграфической технической базы. Чтобы напечатать параллельными текстами английский оригинал и его русский перевод, каждый из которых был сильно структурированным текстом с богатым ассортиментом шрифтовых выделений (6 вариантов начертаний шрифта), пришлось буквально «стать на уши» и лично встревать в ход набора и корректуры.

С теми же проблемами мне пришлось встретиться и через пять лет при издании «Пересмотренного сообщения об Алголе-68», перевод которого я начал в 1973г.

ИФИП предоставила нам готовые фотоформы оригинала описания языка, что позволяло обеспечить *безошибочное* воспроизведение английской части, но для параллельного издания текстов (билингва с поабзацным соответствием) было необходимо применить те же гарнитуры и начертания шрифтов и в русской части, для чего требовалось шесть различных начертаний трех гарнитур.

Оказалось, что это можно было сделать только в цехе фотонабора Первой Образцовой типографии им. А.А. Жданова в Москве, причем работа по подготовке качественного издания оказалась высшей по категории сложности и потребовала очень много времени. Таким образом, перед нами предстал воочию *живой пример* трудоемкого *ручного* использования «системы автоматизированного фотонабора на ЭВМ».

Все это привело к тому, что, взявшись за издание Алгола-68, Первая Образцовая типография им. А.А. Жданова в Москве заказала ВЦ СО АН СССР исследования по автоматизации фотонабора, которыми с 1975 г. и занялись А.А. Берс и (тогда ещё студент) В.В. Медведев. Работа, естест-

венно, началась с системного анализа процессов подготовки изданий, определяющих строение таких систем.

САПФИР (**С**истема **А**втоматизированной **П**одготовки **Ф**отонаборных **И**зданий, обеспечивающая **Р**едактирование) открыл в моей жизни целую эпоху «электронной подготовки изданий», которая продолжилась до 1990 года.

Результаты этого системного анализа процессов издания и сформированных на его основе гипотез и проектных решений практически осели на полках института в виде внутренних отчетов. Однако, по моему мнению, именно этот системный анализ ряда связанных проблемных областей (включая и собственно системное программирование, как проблемную область) дает наиболее содержательное представление о нашей работе.

Сокращенное изложение материалов, хранящихся в архиве А.П. Ершова:

- Система САПФИР:
  Берс.А.А. Эскизный проект Системы автоматизированной подготовки фотонаборных изданий, обеспечивающей редактирование, САПФИР. — Новосибирск, отчет ВЦ СО АН СССР, 1976.
  Берс.А.А., Медведев.В.В. Технический проект Системы автоматизированной подготовки фотонаборных изданий, обеспечивающей редактирование, САПФИР. — Новосибирск, отчет ВЦ СО АН СССР, 1977.
- Проект РУБИН газеты «Правда»:
  Системный анализ производственных процессов по выпуску газеты «Правда», Совместный отчет ВЦ СО АН СССР и Новосибирского филиала ИТМиВТ АН СССР. Н-ск, 1977.
  Берс.А.А. Генеральная схема создания и развития информационно-вычислительной системы РУБИН газеты «Правда». — М. — Н-ск, Издательство «Правда», 1979.

в дальнейшем выделяется изменением шрифта.

Прежде чем читать далее, читателю следует сориентироваться в общем контексте состояния дел с ЭВМ того времени. В стране начался серийный выпуск ЕС ЭВМ, главным образом её младших моделей ЕС-1020 и ЕС-1030, а мини-ЭВМ серии СМ только планировались.

Практически везде машины использовались в режиме пакетной обработки задач. Устройства ввода-вывода — перфокарты и перфоленты, печать через телетайпы и АЦПУ с весьма ограниченным набором символов.

Следует иметь в виду, что оперативная память была на магнитных сердечниках и маленькой (32 — 128 Кбайт), сменные магнитные диски — 7.5 Мбайт, вся внешняя архивная память — на магнитных лентах. О системах коллективного доступа и диалога с машиной ещё только разговаривали, дисплеи видели воочию единицы.

Не вредно также помнить, что микропроцессор Intel-8080 был построен в 1974 г., первый персональный компьютер — в 1976 г., а IBM PC XT был выпущен в 1981 г.

ПРОЕКТ САПФИР

Исходные положения

В основу системы были заложены принципы разделения различных сторон работы по подготовке издания между несколькими типами пользователей и отделения собственно текста от его полиграфического исполнения.

Особо важным представлялось обеспечение комплексного характера полной работы над изданием для всех этапов работ с помощью ЭВМ и накопление в архиве системы не только подготовленных собственно текстов, но и отдельно сформированных способов полиграфического исполнения изданий.

Характеристическими чертами книги как объекта, которым должна быть подчинена работа системы подготовки изданий с помощью ЭВМ, являются, по нашему мнению, следующие.

1. Обязательное соответствие и *верность оригиналу* — это, по-видимому, не требует разъяснений.

2. ***Однородность*** *исполнения* ***одинаковых*** *элементов издания* — это требование проходит «красной нитью» через все руководящие материалы и методические пособия по редактированию и полиграфическому оформлению книжных изданий.

Так, например, все абзацные отступы должны иметь один и тот же размер во всей книге, все заголовки рубрик одной ступени должны быть набраны одним шрифтом и одинаково расположены. Для издания в целом — это относится к формату страниц, расположению колонтитулов и колонцифр, организации подписей к иллюстрациям и т.п.; те же требования однородности применяются для многотомных или серийных изданий.

3. *Книга* является не только продуктом *производства*, но и *произведением искусства* — это относится к каждой книге. Выбор формата и размеров полей полосы, подбор шрифта, размещение иллюстраций и подписей к ним, расположение и способ оформления примечаний, уравновешенность композиции спусков, организация указателей, оглавления, выходных данных и конечно же титульного разворота — все это является предметом заботы и работы не только художественного редактора и художника, но и технического редактора и всех других людей, выпускающих книгу.

Общие свойства системы подготовки изданий

Для того чтобы обеспечить реализацию этих черт книжных изданий, удобство пользования и эффективность работы, программная система подготовки изданий с помощью ЭВМ должна обладать следующими свойствами.

1. Обеспечение ***комплексного подхода к подготовке издания***, который начинается с ввода оригинала в память ЭВМ и завершается выдачей через фотонаборные автоматы полного комплекта форм всех полос книги. Только на этом пути можно обеспечить эффективное использование таких систем и надежность процессов переработки текста.

Обработка издания в ЭВМ должна включать в себя осуществление всей правки, как представленной издательством, так и вызванной неизбежными ошибками операторов-наборщиков. А также реализацию всех указаний технического и художественного редактора — относящихся и к особенностям текста, например, выделения, набор формул и т.п., и к его расположению на полосах, например, размещение иллюстраций, организация заголовков и т.п.

2. Возможность ***легкого внесения изменений*** в текст и его оформление в течение всего процесса подготовки издания на ЭВМ. Рассматриваемая вместе со способностью программных систем возможность осуществить безошибочную реализацию подстановок в текст на всем его протяжении даже при наличии достаточно сложных контекстных ограничений, управляющих такой подстановкой, — эта возможность обеспечивает ***накопление улучшений*** в тексте издания в процессе его подготовки в ЭВМ.

Вместе с комплексностью и возможностью обеспечить защиту частей текста от непреднамеренного вмешательства и повреждения, которую предоставляют современные системы программирования, это обеспечит качественно новый уровень проведения процессов подготовки изданий.

3. Необходимо осуществить ***отделение текста издания от его полиграфического оформления***, что обеспечит качественно новый уровень работ по сравнению с существующей технологией.

4. ***Накопление*** в системе ***общих приемов полиграфического оформления*** и ***средств осуществления*** элементов изданий и возможность использовать их простым указанием на их названия.

Эти приемы и правила, накопленные за века развития книгоиздания, разбросаны по различным методическим и учебным пособиям по полиграфии или же передаются «из уст в уста» как традиции среди полиграфических и издательских работников. Собрание этих приемов и системная организация их применения позволяет после включения их в библиотеку системы сделать применение весьма сложных способов организации текста доступным даже рядовому пользователю.

5. Правильное ***разделение работ***, осуществляемых системой между пользователем-человеком и ЭВМ с учетом того, что программы для ЭВМ не могут в ходе своей работы ни опираться на смысл обрабатываемого текста,

ни воспринимать художественных образов, задуманных оформителем или художественным редактором.

Все решения, принимаемые на основе такой информации, должны делаться человеком, сообщающим затем машине, что нужно сделать, поскольку ЭВМ лучше человека приспособлена для выполнения однообразной рутинной технической работы по реализации этих решений.

Правильный учет указанных факторов позволяет организовать выполнение работ в *два этапа* — однократное, разовое, принятие решения человеком и систематическое, монотонное осуществление принятого решения с помощью ЭВМ.

Дело в том, что исправления, вносимые в набор в процессе подготовки издания, делятся на два класса: типографская правка, вызванная расхождением между набором и указаниями поступившего из издательства оригинала, и издательская правка, связанная с изменениями против оригинала, вносимые в корректуры. При этом издательской (в том числе авторской) правкой считаются любые изменения текста, а также изменения в его полиграфическом исполнении (изменение шрифта, переверстка и т.п.). Нормативы устанавливали, что объем издательской правки не должен был превышать 15% (а позднее и 10%, и даже 5%) стоимости набора для несложных текстов, и 5% (1%) стоимости набора для текстов повышенной сложности.

Однако, соответствующие типовые графики не охватывали той части работы по преобразованию текста, которая проводится издательством от поступления авторской рукописи до сдачи издания в набор. А сюда входят: вычитка, научное и литературное редактирование текста, техническое и художественное редактирование и разметка издательского оригинала.

Из этого видно, что процессы, проводимые в типографии, чередуются с процессами, проводимыми издательством. Причем, являясь процессами внесения изменений в текст, эта работа, с точки зрения выполнения ее в автоматизированной системе с помощью ЭВМ, ничем не отличается от внесения изменений в текст при корректуре набора.

Существенным здесь является тот факт, что только после набора текст начинает находиться ***на вещественном носителе***, позволяющем аккумулировать в себе улучшения, вносимые в текст.

Применение ЭВМ дает возможность создать вещественный носитель текста прямым вводом текста авторской рукописи в память машины с возможностью последующего автоматизированного проведения преобразований текста, реализующих желаемые изменения. Это существенно повышает эффективность и надежность внесения возможных изменений и дополнений в текст.

При этом появляется возможность проведения ***всех работ по подготовке издания*** *внутри* автоматизированной системы, начиная от ввода оригинала в архив системы и кончая выдачей через фотонаборный автомат полного оригинала печатных форм полос издания.

Другими словами, при применении автоматизированных систем подготовки изданий, *чем раньше текст будет введен в память ЭВМ, тем эффективнее и надежнее будут проводиться процессы его обработки и тем быстрее и с меньшими усилиями будет получен искомый результат*.

Отсюда логически следует, что, в принципе, все процессы, связанные с набором и подготовкой оригинала печатных форм, могут быть вынесены за пределы типографий.

Однако следует заметить, что реализация этого принципа потребует значительной реорганизации существующих производственных отношений, что, конечно, значительно сложнее, чем построить автоматизированную систему подготовки фотонаборных изданий.

Предлагаемая в САПФИРе технология проведения процессов подготовки текстов не зависела от того, размеченный или не размеченный оригинал будет вводиться в машину и будут ли в процессе правки в текст вноситься изменения по сравнению с оригиналом. Другое дело, что было заранее понятно и объяснялось заказчику: применение этой системы только внутри типографии не позволит полностью использовать заложенные в ней преимущества.

Разумеется, системный анализ издательско-полиграфической проблемной области включал в себя обзор родственных систем, составленный на основе доступной литературы, которая включает как русские, так и зарубежные книги, научные статьи, программные руководства и технические проспекты по системам управления фотонабором и системам текстового редактирования на ЭВМ. Обзор не затрагивал работ, связанных с ручной подготовкой информации, управляющей фотонабором с помощью специализированных наборно-кодирующих устройств, по которым уже были доступны несколько обзорных статей.

(В.Г.Богомолов и др. Оборудование для переработки текстовой информации и фотонабора // Полиграфия 2, 1974; также номер журнала Datamation. — Vol.16, N 16. — 1970, целиком был посвящен перспективам развития фотонабора.).

Обзор

Следует отметить, что использование отдельных специализированных устройств подготовки перфоленты для управления фотонабором является наиболее распространенным у нас в настоящее время. В связи с этим многие советские статьи в основном посвящены кодированию информации. По-

этому самыми ходовыми терминами, встречающимися в обсуждении фотонабора, являются «полнокодовая» и «неполнокодовая» перфолента. Мы оставили полностью за пределами обзора вопросы кодирования информации внутри систем, поскольку они не являются главными.

Привлечение информации о программах переработки текста, разработанных для внутреннего употребления в системах программирования для ЭВМ, вызвано тем, что в таких программах наиболее глубоко проработаны наборы операций и способы адресации для текстового редактирования, хотя они и рассчитаны, как правило, на выдачу текста через бедные выводные устройства ЭВМ, совершенно неприемлемые с позиций полиграфического качества. Мы старались ограничиться только сравнением функциональных возможностей соответствующих систем, хотя иногда нам и придется ссылаться на некоторые средства реализации этих возможностей.

Режимы работы

С точки зрения режима работы все системы могут быть разделены на два класса: обеспечивающие диалоговой процесс общения и обеспечивающие пакетный режим обработки текста.

Принципиальная разница в подходе к обработке текста в этих двух режимах состоит в том, что при диалоговом общении команды пользователя выполняются немедленно, и поэтому следующие указания должны задаваться по отношению к уже измененному новому состоянию текста. В то же время при пакетной обработке все указания делаются по отношению к исходному тексту, независимо от того, какие изменения в нем этими указаниями производятся.

Этот режим работы соответствует привычному для подготовки книжных изданий процессу корректуры. Многие работы по преобразованию текста носят ярко выраженный групповой характер и, следовательно, хорошо подходят для пакетной обработки. В связи с этим в редактор «ТЕКСТ» ВЦ СО АН СССР была введена наряду с диалоговыми средствами, возможность задания пакета указаний на обработку, что оказалось весьма удобным на практике.

Отметим также, что при наличии дисплеев с ЭЛТ, диалоговый режим упрощает процесс корректуры за счет перехода к неявной адресации по тексту при помощи управления курсором на экране.

Структура текста

Большинство программных текстовых редакторов рассматривает текст как составленный из строк с ограниченным числом символов, однако в некоторых из них текст рассматривается как последовательность букв или слов. Более сложное структурирование текста предлагается системой (A.vanDam, D.E.Rice On-line text editing: a survey // Computing Surveys. — 1971. — Vol. 3, N 3), где введено понятие гипертекста, связывающего отдельные фрагменты текста явными связями. Предложенная в САПФИРе структура текста может

рассматриваться как реализация идеи гипертекста в применении к естественней рубрикации издания и выделению блочных текстов.

Адресация

Почти все системы обеспечивают нумерацию строк в процессе их ввода с последующей адресацией относительно этих номеров. В некоторых случаях, однако, нумеруются отдельные слова текста. Как правило, допускается относительная адресация указанием сдвига от некоторого номера строки.

Кроме того, почти все системы позволяют указать место в тексте заданием образца. Наиболее полно эта последняя возможность разработана в языке Снобол. Введенное здесь понятие образца позволяет не только найти вхождение в текст конкретной последовательности символов, но и описать значительно более сложные предикаты над обрабатываемым текстом. Многие из образцов Снобола не могут быть описаны простым перечислением подстрок, так как подстроки, которые им соответствуют, будут зависеть от контекста, и число их не ограничено.

Многие системы не ограничивались предоставлением пользователю какого-нибудь одного способа адресации, а позволяли ортогонально комбинировать несколько способов. Такой же подход был осуществлен и в САПФИРе, однако он существенно опирался на разделение понятий структуры текста и структуры носителя и независимости собственно текста и его полиграфического оформления.

Команды редактирования

Ассортимент команд текстового редактирования в настоящее время можно считать практически установившимся. Разнообразие в списках команд, наблюдаемое в существующих системах, в основном определяется принятыми синтаксическими средствами, структурой текста и способами адресации. Кроме того, многие системы вводят с целью сокращения объема управляющей информации, команды с комбинированными возможностями.

Основной репертуар включает в себя команды добавления, удаления и замены строк и символов, команды, позволяющие найти или изменить положение текущего указателя, и команды изменения нумерации (адресации) элементов текста. Как правило, наряду с командами, оперирующими со строками, существуют команды внутристрокового редактирования, позволяющие применить те же возможности к отдельным символам. Иногда добавляются команды, позволяющие перенести некоторый фрагмент текста в другое место или заставить некоторый фрагмент быть скопированным сразу в нескольких местах текста.

Защита и достоверность

Как правило, обеспечение защиты текста осуществляется использованием средств защиты операционной системы для файлов, содержащих эти тексты. В некоторых случаях (редактор TECO фирмы DEC) организация ра-

боты обеспечивает одновременное существование, по крайней мере, одной предыдущей редакции текста, что позволяет восстановить результаты работы в случае сбоя.

С целью автоматического исправления ошибок наборщика, в наиболее ответственных случаях применяется двойной ввод информации с распечаткой (или показом на экране) разницы вводимых текстов.

В связи с тем, что в рассматриваемых системах не проводится разделение текстовой и полиграфической информации, в них нет возможности раздельной защиты каждой из этих составляющих.

К средствам проверки достоверности текста следует отнести также возможность распечатки всех мест текста, содержащих указанный образец. Эта возможность может быть скомбинирована из предоставляемых средств во многих системах для мини-ЭВМ типа PDP-11.

Управление выводом

Почти все системы используют для управления расположением выходного текста специальные последовательности символов, включаемых в нужные места в основной текст. Эти последовательности, называемые форматными командами, или директивами, интерпретируются подпрограммами выдачи текста и переводятся ими либо в последовательности символов выходного текста, либо в команды управления выводными устройствами.

Разумеется, текстовые редакторы, рассчитанные на распечатку ЭВМ, имеют значительно более бедные возможности. Однако и они позволяют выключать текст в заданный размер, управлять отступами, делать выделения разрядкой и т.п. Возможна организация разбиения текста на страницы, их последовательная нумерация и надпечатка заголовков, однако распределение текста по страницам должно проводиться пользователем. В числе возможностей, ставших стандартными для большинства систем, есть средства табуляции, позволяющие организовывать выходной текст в виде таблиц и выводов.

Некоторые системы имеют средства для набора многострочных формул. Наиболее разработанное решение этого вопроса дано в работах B.W. Kernigham, L.L.Cherry и в редакторе TROFF J.F.Ossanna. Предложенный в САПФИРе способ набора блочных текстов укладкой является обобщением указанного подхода с учетом разделения текстовой и полиграфической информации.

Фотонаборное устройство и программное обеспечение фирмы «Интернэшнл График К°» дает возможности для одновременного фотонабора текста и иллюстраций (в том числе полутоновых). Такие же возможности анонсированы в проспекте «Система электронного набора и печати «ВидеоКомп» 70/800».

Системная организация

Большинство машинных текстовых редакторов использует возможности операционных систем для организации службы архива.

Системы управления фотонабором, как правило, обслуживаются минимашинами и не имеют достаточных емкостей внешней памяти. Вследствие этого, архив таких систем, как правило, накапливается на перфоленте, т. е. отсутствует как оперативно доступный.

Макротехника

Использование макровозможностей применяется в большинстве текстовых редакторов, разработанных для использования в системах программного обеспечения ЭВМ. При этом такие макросредства применяются не только для того, чтобы с их помощью вводить новые сложные комбинированные команды, но и используются, чтобы вводить сокращения для часто повторяющихся фрагментов текста, подобные же возможности применяются и в программах управления фотонабором для системы Linotron-505, хотя их применение здесь носит более примитивный характер.

На самом деле для пользователя существенным является лишь возможность свертывания определенных фрагментов и, что особенно важно, последующего вызова их или их частей по некоторому имени. При этом для него не существенно, будет ли это свертывание реализовано через макротехнику или посредством использования процедурных свойств языков. Заметим также, что макроподстановка дает лишь возможность статического учета ситуаций, в то время как через процедуры можно получить динамическую информацию о текущем положении дел. Максимальное использование макротехники продемонстрировано в текстовом процессоре TRACE.

В системе САПФИР макротехника широко используется при генерации конкретного экземпляра системы, а также как средство, предоставляемое системному полиграфисту при подготовке процедур реализации форматных директив (фордов) для библиотеки. Во всех остальных случаях для сохранения непрограммистского характера общения пользователей с системой применена прозрачная для него смешанная стратегия вызова наиболее подходящего в данном случае экземпляра требуемой процедуры, заранее запасенного в библиотеке.

В заключение обзора перечислим те свойства системы САПФИР, для которых не было найдено аналогов в доступной литературе.

Во-первых, это последовательное проведение разделения собственно текстовой и полиграфической информации.

Во-вторых, это комплексный подход к подготовке изданий, опирающийся на ввод в первую очередь исходного текста издания для хранения в архиве системы и накопления в нем всех последующих результатов подготовки.

В-третьих, подход к верстке текста с предварительным расчетом макета издания.

Системный анализ строения книжных текстов

Задача системного анализа текстов состояла в определении основных элементов и структур, из которых составлен текст; их внутренних характеристик; взаимных связей; информации, требуемой для их задания; а также информации, с помощью которой определяется их полиграфическое осуществление.

Кроме того, этот анализ должен был привести к разделению полной информации, требуемой для осуществления некоторого издания, на информацию, описывающую собственно текстовую сторону этого издания (называемую далее всюду просто *текстом*), и информацию, задающую способы и особенности полиграфического воплощения (везде далее просто *полиграфия*) этого издания.

Для обоснования как выделяемых элементов и структур текста, так и для отделения текста от полиграфии использовались «мысленные эксперименты» по трансформации оформления и прослеживании при этом, при каких условиях и в какой степени будет оставаться **неизменным смысл** соответствующего фрагмента текста.

Понятие «смысла (содержания) текста» не будет формализоваться в каком бы то ни было направлении. Однако, при изложении всегда будет предполагаться, что читатель в состоянии содержательно следовать за ходом рассуждения, чтобы в каждый момент принимать аргументацию, проводимую в данном проекте (или находить в ней ошибки).

Наименьшим элементом текста является символ. Мы относим к символам все видимые при печати знаки: буквы, цифры, знаки препинания, специальные значки, используемые в различных областях науки и техники, и т.п. Все символы образуют видимый алфавит.

Определение 1. ***Прототекстом*** называется линейная упорядоченная последовательность символов видимого алфавита.

В данный момент уместно сказать о том, каким образом (вообще говоря, давно изобретенным) предлагается представлять текст, Для этой цели наряду с вышеупомянутым видимым алфавитом предполагается использовать *невидимый дополнительный* алфавит, символы которого вставляются в нужных местах между символами прототекста таким образом, что получившийся текст удовлетворяет следующему

Определению 2. ***Текст*** есть линейная упорядоченная последовательность символов, принадлежащих *объединению видимого и невидимого* алфавитов.

Таким образом, чтобы завершить описание строения текста, нам необходимо будет перечислить состав символов невидимого алфавита.

Сегодня в гипертекстах это называется тегами (разметкой), причем невидимыми становятся все символы, заключенные в скобки невидимости, для которых используются «<» и «>».

Информация о полосах

В соответствии с принятой практикой мы будем рассматривать полосу как основную единицу полиграфии. Очевидно, что вариации размеров полосы (в широких пределах) никак не влияют на содержание. Следовательно, информация о размерах полосы целиком принадлежит к полиграфии.

Не влияют на смысл текста также нормы и сигнатура (которые зависят от раскладки полос на листе). Информация о них тоже относится к полиграфии.

Легко представить себе два издания одного и того же текста, отличающиеся тем, что в одном из них нет колонтитулов, а в другом они есть. Конечно, пользоваться вторым из них будет удобнее, но с достаточной для целей нашего анализа уверенностью мы можем отнести к полиграфическому оформлению и колонтитулы. Однако в случае «бегущих» колонтитулов их содержание будет зависеть от текста, попадающего на соответствующую полосу. Этот факт необходимо иметь в виду при рассмотрении связей как элементов внутри текста, которые мы рассмотрим позднее.

Это же следует учитывать и в связи с отнесением к полиграфии колонцифр, здесь тоже возникают (при ссылках типа «см. стр. ххх») связи как элементы текста.

Очевидно, что ничего не меняется в содержании и при многоколонном размещении текста на полосе, следовательно, это тоже полиграфия. Вопросы, относящиеся к параллельному набору текстов, например, одного и того же текста на двух или более языках, отложим до обсуждения текстовых связей, поскольку они будут аналогичными как для колонок, так и для параллельного расположения текстов на смежных полосах разворота.

Точно также, смысл остается независящим от выбора гарнитуры, начертания и кегля основного шрифта, который использован при издании (о выделениях см. ниже), и от принятого интерлиньяжа.

Мысленные эксперименты по варьированию оформления текста, результатом которых явились приведенные выше утверждения, не были сформулированы явно вследствие их тривиальности. Мы и в дальнейшем будем описывать только те эксперименты, которые требуют детального изложения, а во всех остальных случаях ограничиваться лишь формулировкой результатов.

Связный текст

Уже на первый взгляд сразу видно разницу в наборе связного (или линейного) текста и двумерного текста, смысл которого зависит от взаимного расположения его элементов на плоскости страницы. Строение двумерного текста, типичными представителями которого являются формулы и таблицы,

рассмотрим позднее, а сейчас разберемся в элементах и структуре линейного текста.

Поскольку все преобразования текста, не затрагивающие его смысла, должны обеспечивать неизменность порядка расположения входящих в этот текст символов, то инвариантность прототекста является *необходимым, но не достаточным* условием сохранения смысла при преобразованиях текста. Естественные тексты имеют, на самом деле, более богатую структуру, чем простая линейная последовательность символов.

Действительно, буквы группируются в слова, слова в предложения, предложения в абзацы, а далее начинается иерархия рубрик текста, которая в разных изданиях имеет разную высоту (глубину) и разные типы организации. Один из возможных примеров:

том — книга — часть — глава —
параграф — абзац — предложение — слово — буква

Рубрикация

Информация о рубрикации не принадлежит прототексту, но должна *принадлежать тексту*, так как ее восстановление по прототексту не всегда может быть проведено однозначно, а ее утрата может привести к искажению смысла. Так, например, при не учете границ слова нельзя различить «парафраз» и «пара фраз».

Аналогично, хотя в типографиях иногда для вгонки или выгонки строки и имеет место создание или уничтожение абзаца, но по своему назначению «каждый абзац должен начинать новую мысль, новую микротему и внутри абзаца не должно быть отклонений от нее» (см. Справочная книга корректора и редактора. — М.: Книга, 1974.)

Следовательно, деление текста на абзацы (так же как и на предложения) носит смысловой характер и соответствующая информация (не восстанавливаемая автоматически) должна явно сохраняться в составе текста.

Исходя из приведенного выше анализа рубрикаций, мы включаем в состав невидимого алфавита разделители рубрик различного уровня.

При этом предполагается, что разделители рубрик, стоящие в тексте, могут быть некоторым образом связаны с указаниями о том, как оформляются рубрики одного и того же уровня (спусковая полоса, способ набора заголовка рубрики, нумерация рубрик), но сама эта информация, которая может варьироваться от издания к изданию, в текст не входит. См. также обсуждение выделений в тексте в связи с оформлением заголовков.

Состав алфавитов

Вернемся к рассмотрению видимого и невидимого алфавитов. Вопросы, связанные с определением состава входящих в них символов, зависят от многих факторов, связанных с реализационными аспектами проектирования системы и с составом и характеристиками используемого оборудования. Здесь мы остановимся лишь на тех факторах, которые при этом приходится учитывать.

1. Большое разнообразие начертаний символов, используемых в книжных изданиях, заведомо превосходит количество разных символов, которые любой фотонаборный автомат (или другое печатающее устройство) может предоставить для одновременного доступа. Это приводит к тому, что все используемые символы видимого алфавита обычно разбиваются на некоторые группы, которые мы, в соответствии со сложившейся практикой, будем называть регистрами (иногда они называются магазинами, рамками и т.п.).

В связи с тем, что в любой данный момент доступны символы только одного регистра, возникает необходимость во введении в состав невидимого алфавита символов перехода от одного регистра к другому.

Обычно принято различать два типа переходов с регистра на регистр: запоминаемый переход, действующий на все последующие символы до тех пор, пока не встретится новый запоминаемый переход, и незапоминаемый переход, действие которого распространяется лишь на ограниченное число следующих за ним символов (обычно на один символ).

2. Ограничения на состав знаков разных регистров сильно зависят (и обычно являются разными) от типа устройства. Это делает необходимыми работы по перекодированию представлений символов при переходе от одного устройства к другому и даже при переходе от одной работы с текстом в системе к другой.

Так, например, на стандартной входной клавиатуре (типа пишущей машинки) прописные и строчные буквы находятся на разных регистрах, в то время как в магазинах фотонаборных автоматов — на одном и том же. Поэтому выбор внутреннего для системы состава алфавитов (как видимого, так и невидимого) в большой степени будет определяться необходимостью уменьшить работу по перекодировке при переходе от одного уровня в системе к другому.

3. Еще одним важным фактором, влияющим на состав, а главным образом, на упорядоченность алфавитов, будут операции типа сравнения в процедурах обработки текста. Так, например, для выполнения переносов необходимо легко и эффективно различать гласные и согласные буквы.

4. Еще одним вопросом, решение которого будет сильно зависеть от выбранных способов реализации и от особенностей процедур обработки текста — это проблема пробела. Суть проблемы в том, помещать ли пробел (а именно, переменный пробел) в состав видимого алфавита или же считать его разделителем слов, входящим в состав невидимого алфавита. Трудности здесь связаны с ограничениями на возможности переноса некоторых сочетаний слов, например, отрезок текста «тов. И.С. Кон» нельзя разбивать переносом.

Выделения

Многообразие способов шрифтовых и не шрифтовых выделений в тексте позволяет сразу предположить, что большая часть информации о них относится к полиграфии.

Действительно, легко представить себе, что в процессе редактирования или вследствие бедности производственных возможностей вместо выделения другой гарнитурой будет использован полужирный прямой, а курсив всюду будет заменен разрядкой. Не знавший первоначальных замыслов автора (если они, вообще, носили столь определенный характер) читатель и не заметит такой замены, поскольку на смысл использования выделений она не повлияет.

По нашему мнению, текстовой информацией о выделениях служит только сам факт их наличия и то, что внутри одного текста могут быть выделения разных сортов. Вся остальная информация относится к способу реализации выделений и, следовательно, к полиграфии.

Поэтому для учета выделений в тексте достаточно иметь в составе невидимого алфавита необходимое число сортов парных ограничителей — выделительных скобок, в которые заключаются в тексте выделяемые фрагменты. Впоследствии с каждым сортом этих ограничителей будет связано полиграфическое оформление, принятое для данного выделения в данном конкретном издании.

Заголовки

Интересным следствием такого подхода к выделениям является возможность связать соответственно некоторые специально выделенные сорта выделений с разделителями рубрик разного уровня, заключая в выделительные скобки связанный с данным уровнем заголовок соответствующей рубрики. При этом по цепочке:

уровень_рубрики — тип_выделения — оформление_выделения

сразу обеспечивается однородность оформления заголовков рубрик одного уровня и сохраняется легкость его изменения.

Связи в тексте

До сих пор мы рассматривали только структуру и элементы связного (линейного) текста, который не покрывает, конечно, всего многообразия структур текстов. Во-первых, как уже упоминалось, существуют существенно двумерные тексты, а во-вторых, имеются части текста, связанные с некоторым связным текстом, но лежащие вне него, хотя они и входят вместе с ним в один полный текст.

Вследствие сказанного мы будем различать следующие типы связей или ссылок:

- внутритекстовые ссылки, облегчающие установление смысловых связей между удаленными частями текста. Например, «см. стр. XXX», «см. раздел 2.1.1.», «как сказано в главе II»;
- ссылки из линейного текста на примечания к нему;
- связи с двумерными фрагментами текста: формулами, иллюстрациями и подписями к ним, таблицами и выводами;

- связи из текста на вспомогательные элементы издания, например, «бегущие» колонтитулы, колонцифры и т.п.;
- связи из одного текста в другой, например, при параллельном издании одного и того же текста на двух (или более) языках.

Как и в предыдущих случаях, связи необходимо относить ***к тексту***, а не к полиграфии, включать как элементы невидимого алфавита и различать связи разных типов. Тогда с соответствующим типом связи можно впоследствии сопоставить соответствующий способ его осуществления так же, как это будет делаться для рубрик и выделений.

При этом связь всегда указывает на требуемое место ***в тексте***, а полиграфическое оформление соответствующей ссылки, например требуемый номер страницы, система может подставлять автоматически.

Суммируя все вышеизложенное, получаем, что каждый связный текст состоит из прототекста, в нужные места которого вкраплены символы невидимого алфавита, с помощью которых отражается рубрикация текста, использование регистров, выделения и связи как внутри данного текста, так и на другие тексты, сцепленные с данным.

В тексте полностью отсутствует информация, определяющая его полиграфическое оформление, — она должна задаваться отдельно, тоже невидимыми символами, но другими.

Само собой разумеется, что текст, на который указывает связь из некоторого связного текста, сам может иметь структуру связного текста или же быть двумерным текстом, к определению структуры последнего мы теперь и перейдем.

Двумерные тексты

Для элементов двумерных текстов, которые мы, для краткости, будем называть **блоками**, необходимо задавать их взаимное расположение по отношению к другим таким же элементам, причем по обоим измерениям. При описании структуры блочных текстов необходимо учитывать следующие обстоятельства.

1. Практически почти все устройства ввода текста в ЭВМ, кроме так называемых «графических дисплеев», позволяют вводить с клавиатуры только линейный текст. Поэтому указания о двумерности расположения должны быть представлены в этом линейном тексте специальными (невидимыми) символами.

2. Поскольку различные фотонаборные автоматы обладают разными возможностями в отношении состава выполняемых ими при наборе полосы действий, то описание структуры блочных текстов должно быть независимым от способа ее осуществления. Другими словами, должна быть описана структура соотношений, статически определяющая взаимное расположение блоков, а не программа, динамически описывающая, что будет делать фотонаборный автомат при реализации текста.

3. С целью отделения собственно текста от его полиграфического оформления, взаимное расположение блоков должно задаваться относительным способом, не опирающимся на их абсолютные размеры, которые должны определяться системой автоматически для выбранных правил набора блоков и реализации соотношений между ними.

Отделение текста от полиграфии

Преимущества, которые дает разделение текста и его полиграфического исполнения в системе автоматизированной подготовки изданий, проявляются на всех уровнях процесса подготовки издания:

- на издательском уровне — возникает возможность легкого, без повторения всех работ по набору и выверке, текста, переиздания книги в другом полиграфическом варианте;
- на редакторском уровне — отсутствие опасности при смене оформления испортить текст.

Простота и экономичность получения вариантов освобождает редактора (а также технического и художественного редакторов) от «бремени *уже сделанной* работы», позволяя им ставить эксперименты для выбора наилучшего способа оформления текста, проверки их путем выдачи пробных страниц через фотонаборный автомат и возможности сравнить варианты перед принятием окончательного решения; на уровне оператора-наборщика — поскольку связь элементов, составляющих текст с их полиграфическим оформлением, устанавливается однажды заранее, то объем набираемой информации значительно уменьшается.

Справедливость этого положения тем больше, чем сложнее набираемый текст. Особенно большая экономия получается при наборе формул и таблиц, т.е. блочных текстов.

В общем случае — чем сложнее текст и оформление подготавливаемого издания, тем больше выгоды от применения автоматизированной системы для его обработки.

Заливка и укладка

Можно заметить, что способы организации набора линейных текстов и блочных (двумерных) текстов прямо противоположны. Размещение абзацев линейных текстов производится в пределах некоторого, заранее отведенного, места на полосе, и при этом совершенно несущественно, как распределятся части текста по элементам этой полосы (строкам).

Такой способ набора можно назвать *заливкой текста* в предназначенное для него место. Разбиение текста на строки производится системой автоматически, различается при наборе на разный формат и для читателя практически совершенно не влияет на смысл текста.

При блочном наборе распределение частей текста по блокам (в частности, определение того, какой фрагмент текста составит строку) проводится с самого начала. В противоположность предыдущему способу, здесь, однако,

заранее не определено, сколько места будет занимать блочный текст после его осуществления.

Напротив, именно эта информация вычисляется системой автоматически, исходя из распределения текста по элементарным блокам (строкам). Этот способ набора мы будем называть *укладкой* текста *по блокам*.

Любой текст издания можно представить как совокупность линейных и блочных его частей, упорядоченных определенным образом и связанных между собой. Отсюда легко следует, что управление расположением этого текста можно обеспечить, используя следующие средства:

- укладка (для набора блочных текстов);
- размещение элементов издания непосредственно в терминах фактических размеров полосы, например, постановка иллюстрации, указание величины спуска и т.п.;
- заливка линейных текстов в оставшееся от применения предыдущего способа место на полосе.

В совокупности эти три способа решают задачи верстки, используя макет полос как исходное пространство размещения.

Групповые действия

Большинство процессов обработки текста носят характер систематического применения однородных указаний во многих местах на протяжении всего или большой части текста. Система предоставляет всем пользователям общие возможности задания такой работы путем однократного описания соответствующих указаний и задания принципа нахождения соответствующих мест в тексте.

Общая схема организации таких групповых действий следующая:

- ***диапазоны*** (указывающие части текста, внутри которых проводятся преобразования),
- ***предикат*** (посредством которого обнаруживаются необходимые места, например, по образцу или через задание связи),
- ***указание*** (которое необходимо выполнить),
- ***возврат*** (на поиск следующего места).

Такие групповые действия обеспечивают контекстную замену, поиск всех мест текста, содержащих заданный образец, и снабжение одинаковых элементов текста однородным полиграфическим исполнением.

Применение системы САПФИР для подготовки изданий создавало для пользователей системы некоторое качественно новое рабочее окружение. Особенности этого рабочего окружения должны были способствовать существенному улучшению производительности труда и создавать дополнительные удобства пользователям, но, с другой стороны, это рабочее окружение предъявляло к пользователю новые дополнительные требования и

меняло в некоторых отношениях (по бедности техники) сложившийся стиль редакционно-издательской работы.

Рабочее окружение и пользователи

Одним из новых требований, вызываемых рабочим окружением, является использование адресации для указания мест в тексте, подлежащих редактированию (корректуре).

Другой существенной чертой рабочего окружения является различие между описываемым текстом в том виде, в каком он появится на полосах готового издания, и описывающим его текстом, который видит пользователь в процессе обработки издания на системе и который предъявляется ему в форме главной технической распечатки (ГТР).

Существует много причин, вызывающих различие между описываемым и описывавшим текстом, и разница в печатных возможностях устройств ЭВМ и фотонаборных автоматов, будучи существенной, не является определяющей в этом случае.

Основные причины указанного различия следующие:

- полиграфические издания используют шрифт с буквами различной ширины, отсутствующий на обычных воспроизводящих устройствах современных ЭВМ;
- подготовка изданий на разных языках и по всем отраслям человеческой деятельности приводит к применению очень большого числа используемых символов, принадлежащих разным графическим основам (символы, используемые в разных науках, все не европейские алфавиты и т.п.);
- ограничения стандартных устройств обмена в ЭВМ требуют предварительного описания двухмерных (блочных) текстов с помощью некоторого линейного языка;
- отделение собственно текста от его конкретного полиграфического оформления, для этого необходимо оставлять в составе собственно текста указания об абстрактных выделениях, уровнях рубрикации и связях, привязывая к ним самостоятельно задаваемые указания о способах оформления соответствующих элементов собственно текста.

Важной особенностью создаваемого САПФИРом рабочего окружения является безусловная верность выполнения редактирующих указаний пользователя, делающая ненужными процессы сверки выправленного текста. Кроме того, система обеспечивает неизменность частей текста, не затронутых при редактировании.

В процессе подготовки издания часто возникает необходимость в осуществлении серии однородных действий, применяемых к разным местам текста, определяемых некоторым явно сформулированным условием. Вводимое САПФИРом рабочее окружение позволяет осуществлять такую серию

действий, задав ее одним указанием. Исполнение этих действий во всех требуемых местах текста гарантируется системой.

Рабочее окружение, создаваемое САПФИРом, обеспечивает проверку всей информации, предъявляемой пользователем (как текстовой, так и управляющей), на непротиворечивость и безопасность применения к тексту, обеспечивая тем самым защиту обрабатываемого системой текста от непреднамеренной его порчи пользователем.

Текст и носитель

Как и всякий другой вид информации, текстовая информация всегда представлена на каком-нибудь материальном носителе. При этом способ записи в особенности строения носителя привносят дополнительное структурирование информации, накладывающееся на структуру собственно текста.

В процессе переработки текста в системе она многократно переписывается с одного вида носителя на другой. Сделать невидимыми для пользователя особенности представления информации на разных носителях, а также обеспечить независимость собственной структуры текста — одна из главных задач, возникающих при разработке системы.

Однако структуры, по крайней мере, двух носителей **существенно неустранимы** из поля зрения пользователя:

- во-первых, представление перерабатываемого текста в главной технической распечатке (ГТР), по которой пользователь будет обращаться к тексту при его подготовке;
- во-вторых, тот вид текста, который он получит по завершении всех процессов его переработки системой, т.е. комплект оригиналов печатных форм издания.

Знание структурных особенностей первой из упомянутых форм необходимо всем пользователям, а работа в терминах готовых полос может оказаться существенной для художественного (технического) редактора.

Структура текста. Основными элементами структуры текста в системе САПФИР являются: рубрикация, выделения, связи (например, внутритекстовые ссылки). Рубрикация обеспечивается разделителями и ведется от абзаца вверх (пункт, параграф, глава и т.п.) с последовательной нумерацией уровней рубрик. Такой способ представления рубрикации выбран, поскольку число уровней и стиль рубрикации в разных изданиях могут быть различными и, в отдельных случаях, могут быть вообще отнесены к полиграфическому оформлению, а не к собственно тексту.

Более мелкие, чем абзацы, части текста (предложения, слова) не обеспечиваются явно структурой собственно текста, поскольку строение текста

внутри абзаца совпадает с его же строением, обеспечиваемым структурой носителя.

В структуру собственно текста включаются только абстрактные выделения, обеспечиваемые нумерованными ограничителями, связи в тексте и из текста также обеспечиваются с помощью ограничителей, принадлежащих к «невидимой части» базового алфавита системы САПФИР.

Структура носителя. Основной, видимый носитель текста (распечатка) имеет структуру четырехмерного массива позиций.

Весь текст издания целиком рассматривается в архиве системы как один заказ. Заказ состоит из последовательно расположенных секций (первое измерение), составленных из последовательности страниц (второе измерение). Каждая страница является последовательностью строк (третье измерение), образованных следующими одна за другой позициями (четвертое измерение).

Подчеркнем, что понятия строки и страницы принадлежат только структуре носителя и не входят в понятие структуры собственно текста. Важно также понимать, что размещение текста по строкам и страницам ГТР никак не связано с будущим размещением этого текста на полосах издания.

Адресация. Самая главная проблема, которая должна быть решена при проектировании адресации, — в максимальной степени обеспечить неизменность адресов тех элементов (фрагментов) текста, которые не затрагиваются данной редактирующей операцией. При изменении текста может измениться размещение элементов структуры собственно текста по элементам структуры носителя и, кроме того, могут измениться «расстояния», выражаемые как в элементах структуры собственно текста, так и в элементах структуры носителя.

Именно из этих соображений структура носителя и способы адресации были выбраны в системе САПФИР таким образом, чтобы обеспечить пользователя отображением текущего состояния текста на основе постраничной замены в распечатке только тех ее страниц, которые были затронуты при редактировании.

Адресация может базироваться на структуре текста или структуре носителя и в каждом из этих случаев может быть прямой или относительной. Кроме того, возможна адресация по содержанию текста (по образцу). С некоторыми ограничениями указанные выше способы адресации свободно могут комбинироваться друг с другом. Ограничения, накладываемые на адресацию, объясняются, главным образом, необходимостью обеспечить ее эффективную реализацию.

Секции нумеруются в пределах заказа. Страницы нумеруются в пределах секции, поэтому полный номер страницы имеет вид: номер-секции, номер-страницы, однако при работе в пределах секции номер текущей секции может опускаться по умолчанию.

Строки последовательно нумеруются в пределах секции независимо от их расположения на той или иной странице. Кроме того, можно пользоваться относительным адресом строки на странице.

Наличие двух самостоятельных способов адресации для строк создает дополнительные удобства пользователю и позволяет сохранить стабильный номер строки, даже если в процессе редактирования она перемещается с одной страницы распечатки на другую.

В процессе отдельного сеанса редактирования система следит за тем, содержание каких страниц распечатки изменяется, с тем, чтобы выдать новую распечатку только этих страниц в конце сеанса.

Вследствие небольшого объема и легкости обозримости строки в целом существует только относительная адресация позиций в строке. Поскольку в пределах абзаца структурирование текста в распечатке совпадает со структурой носителя, абзац состоит из строк, а последние — из позиций, адресация по структуре собственно текста ведется на уровне абзацев и выше.

Другим способом адресации по тексту является адресация по образцу. Фактически система в этом случае вырабатывает адрес по структуре носителя того места текста, которое буквально совпадает с заданным образцом. Система может обеспечить также поиск и выдачу адресов всех вхождений заданного образца в некоторую часть текста.

Выбранная система адресации предоставляет пользователю богатые возможности, являясь более мощной, чем системы адресации, допускаемые аналогичными отечественными и зарубежными системами, оставаясь при этом эффективно реализуемой. За исключением поиска по образцу, все способы задания адресов и диапазонов статически приводятся (блоком адресного анализа) к фактически реализованному в системе адресу «секция, номер-строки, позиция».

Средства текстового редактирования.

В состав средств редактирования входит описание структуры распечаток, способы адресации и команды.

В связи с большой важностью обеспечения *серийных* работ при редактировании текста в общем случае строение команды текстового редактирования имеет следующий вид:

в ДИАПАЗОНЕ при УСЛОВИИ выполнить ДЕЙСТВИЕ,

где *ДИАПАЗОН* указывается парой граничных адресов, *УСЛОВИЕ* может быть задано образцом, а *ДЕЙСТВИЕ* выбирается из следующего списка команд текстового редактирования:

вычеркнуть *текст между двумя указанными позициями,*
вставить *указанный текст в указанную позицию,*
скопировать *указанный текст в указанные места,*
переставить *указанные фрагменты текста в указанном порядке,*
перенести *указанный фрагмент текста в указанное место,*

сменить *нумерацию строк текста,*
уплотнить *указанный фрагмент текста за счет неполных строк в абзаце,*
печатать *указанную часть текста в указанном стиле.*

Этот список команд может быть в дальнейшем пополнен, в частности, за счет команд, связанных с работой за дисплеем в диалоговом режиме.

В зависимости от конкретных потребностей не все компоненты структуры команды, названные выше, должны задаваться в каждом отдельном случае. Использование предоставляемых системой средств умолчания, помноженное на разнообразие задания адресации и условий, превращает этот небольшой список в мощный набор средств текстового редактирования, способный удовлетворить самого взыскательного пользователя.

К средствам редактирования относится также язык описания блочных (двумерных) текстов. Описание семантики этого языка более подробно изложено в работе *Берс А.А., Детушев В.А. Управление набором блочных текстов в системе САПФИР.*

В состав основных средств редактирования включены средства свертывания наборов команд, которые могут называться по их имени. Такие макросредства значительно увеличивают изобразительные возможности языка редактирования.

В состав средств редактирования включены форды, которые могут быть вставлены в текст соответствующими командами. С помощью фордов осуществляется вызов процедур управления расположением и форматом текста, находящихся в системной библиотеке фордов (СБФ). Вызов этих форматных процедур может производиться также неявно через связи, устанавливаемые в спецификации заказа между некоторым типом элементов структуры текста и соответствующей процедурой.

Вид текста в процессе подготовки

Организация базового алфавита. Исходя из того, что система будет перерабатывать тексты на многих языках, основной состав алфавита представляет собой объединенный полиграфический алфавит, рассчитанный на обслуживание 124 языков, письменность которых построена на основе латиницы и кириллицы. Этот алфавит, как известно, содержит 450 различных символов. С целью уменьшения числа символов проводится отделение акцентов от основных знаков с тем, чтобы основной регистр базового алфавита укладывался в 192 символа. Оставшиеся символы (64) образуют *невидимый* алфавит и используются для служебных целей и разметки.

Вследствие этого базовый алфавит включает в себя некоторое число регистров, один из которых занят акцентами, а другие могут содержать символы на других графических основах, математические символы и т.п.

Переход от одного регистра к другому осуществляется специально выделенным символом невидимого алфавита. Переход к регистру акцентов (своему для каждого основного регистра) управляется другим невидимым

специальным символом, область действия которого распространяется на один следующий за ним знак. Вводится специальный (невидимый) знак «лигатура», действующий аналогично акценту, но в пределах одного и того же регистра.

Введенный механизм регистрового деления алфавита позволяет обеспечить выбор любого символа с требуемой графемой. Отработка перехода с одной гарнитуры на другую, а также управление размерами знаков по кеглю и ширине обеспечивается использованием средств задания выделений в тексте. Принятая регистровая структура организации базового алфавита обладает достаточной для требуемых применений гибкостью и, в то же время, может быть эффективно реализована на машинах ЕС ЭВМ при помощи имеющихся там специальных команд перекодировки.

Как было сказано, распечатки организованы в пачки страниц, каждая из которых представляет собой соответствующую секцию заказа. Число строк на странице и длина строк определяются режимом работы системы.

В заголовке каждой страницы печатается информация, позволяющая идентифицировать текст, к которому принадлежит страница (например, обозначение заказа), место данной страницы в этом тексте (например, номер-секции — номер-страницы) и обозначение редакции (например, дату последнего изменения на ней). Кроме того, каждой секции предшествует титульная страница, которая может содержать кроме обозначения заказа и секции служебную информацию о текущем состоянии распечатки данной секции, например: сведения о редакции каждой из страниц секции, использованный режим управления форматом распечатки данной секции и т.п.

По завершении каждого отдельного сеанса редактирования конкретной секции система определяет, какие из страниц предыдущей распечатки требуют замены, и выдает для них новую распечатку. При этом в случае вставки новых частей текста система заводит новые промежуточные страницы с тем, чтобы по возможности уменьшить число перепечатываемых страниц. Вместе с комплексом заменяемых/добавляемых страниц всегда выдается новый титульный лист секции. Для поддержания распечатки, правильно отражающей текущее состояние текста, пользователь должен произвести замену/вставку полученных страниц в свой комплект распечатки в соответствии с указаниями нового титульного листа.

При необходимости реорганизация расположения текста в некоторой части или во всей распечатке может производиться как отдельный сеанс редактирования. Эта реорганизация может включать перенумерацию строк, должна применяться не менее чем к секции и во всех случаях делает предыдущую распечатку этой секции недействительной.

Оригинал-макет

Для того чтобы иметь возможность проверить результаты работы системы по подготовке изданий без дорогостоящей выдачи всего текста через

фотонаборный автомат, предусматривается возможность распечатки оригинал-макета.

Эта распечатка проводится после отработки фордов и верстки текста и отражает точное распределение текста по строкам и страницам после выключки и с учетом расположения иллюстраций и блочных текстов на полосе. Никакая нетекстовая информация (уже отработанная при верстке-выключке) в оригинал-макете не распечатывается.

Страницы оригинал-макета содержат в точности то же число строк, что и полосы готового издания.

Места, занятые иллюстрациями, оставляются в оригинал-макете незапечатанными, и внутри них, при необходимости, можно дать обозначение соответствующей иллюстрации. Оригинал-макет отражает аналогичным образом спуск полос, отбивку рубрик, размещение построчных примечаний и т.п. Система предоставляет также возможность заказать параллельно с оригинал-макетом выдачу отдельных полос издания через фотонаборный автомат.

Архив и библиотеки

Архив системы САПФИР предназначен для хранения обрабатываемых текстов, а также ранее отработанных текстов прошедших заказов для обеспечения возможности их переиздания. Предусматривается два уровня архива:

- оперативный архив, в котором хранятся заказы, находящиеся в работе,
- неоперативный архив прошлых текстов, хранимых с целью переиздания.

Каталоги обоих уровней постоянно присутствуют в системе. Тексты оперативного архива располагаются, как правило, на магнитных дисках, но, в случае недостатка на них места, могут временно размещаться и на магнитных лентах. Тексты в неоперативном архиве всегда хранятся на магнитных лентах. Перемещение текста с одного уровня архива на другой требует отдельной работы, инициируемой оператором системы.

Библиотеки системы делятся на два класса: *библиотеки сокращений* и *библиотеки форматных директив*. Они располагаются на магнитных дисках и всегда доступны обрабатывающим процессорам.

Организация работы

Процесс подготовки издания в системе САПФИР складывается из некоторой последовательности сеансов обработки текста. Начальный сеанс состоит во вводе текста и выдаче ГТР. Каждый последующий сеанс включает требуемую заказанную обработку текста из архива с выдачей заменяемых страниц ГТР.

Сеанс, естественно, разбивается на две части:

- человеческая работа, состоящая в чтении ГТР и подготовке пакета указаний для системы к набору;

- машинная работа, состоящая во вводе указаний, в их выполнения и выдаче результатов.

В результате сеанса в архив помещается измененное состояние обработанного текста. С целью страховки пользователя от его собственных ошибок предыдущая редакция текста тоже хранится в системе и может быть вызвана по особому указанию. Обычно в конце сеанса его результат становится текущей редакцией, а старая предыдущая редакция текста замещается старой текущей редакцией.

Работы и процессоры

В ходе сеанса, как правило, выполняется одна или несколько работ. Понятие работы рассматривается с точки зрения пользователя и обозначает некоторый логически замкнутый комплекс действий, используемый для обработки текста или для изменения состояния каких-либо компонент системы. Для каждого вида пользователя определен набор работ, которые может выполнять для него система.

Структура общения пользователей и обслуживающего персонала с системой САПФИР поддерживает производственный поток информации, обслуживающий основной технологический процесс подготовки издания, и справочный поток, снабжающий пользователей информацией о текущем состоянии системы.

Оба эти потока замыкаются через пользователей, начинаясь с подготовки данных операторами-наборщиками и завершаясь соответствующими распечатками.

Такая структура общения с системой выбрана вследствие того, что она инвариантна к выполнению работ в пакетном или диалоговом режиме связи пользователей с системой. Здесь рассматривается стартовый вариант системы, рассчитанный на пакетную работу. Дальнейшее расширение будет обеспечивать возможности диалога, что потребует пополнения этой структуры при ее сохранении добавочным программным обеспечением. При этом правила задания работ останутся для пользователей почти неизменными.

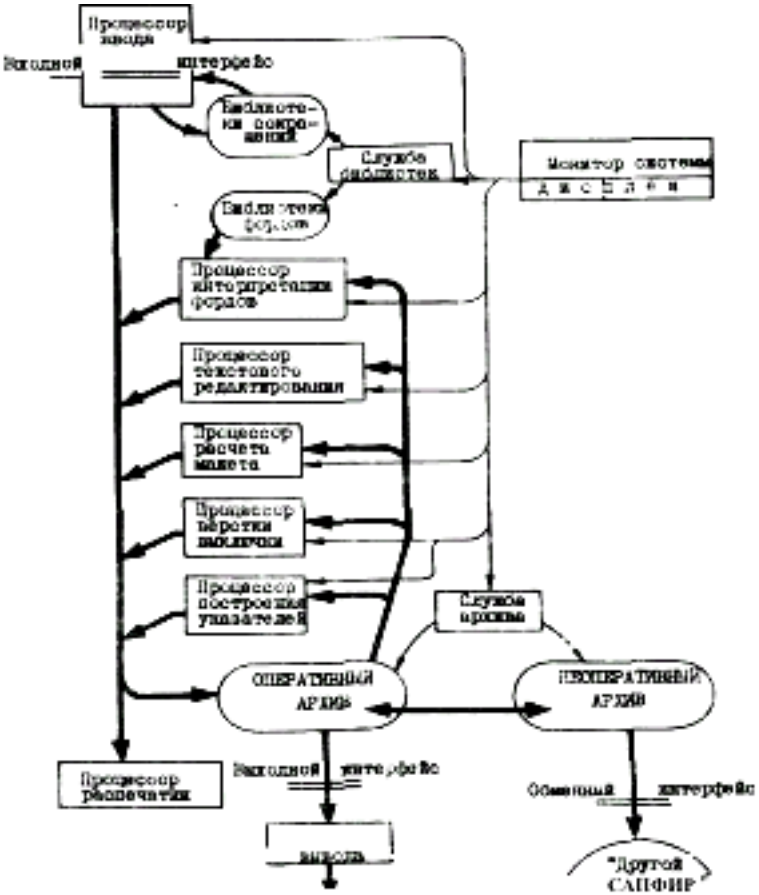
Основными исполнительными *структурными единицами* системы, обеспечивающими проведение работ, являются процессоры. При этом процессор может выполнять одну или несколько работ целиком или же одна работа может выполняться несколькими процессорами. Однако соотношение между работами и процессорами не допускает, чтобы часть какой-либо работы выполнялась частью некоторого процессора.

Процессоры системы делятся на:

- обрабатывающие процессоры,
- процессор текстового редактирования,
- процессор интерпретации фордов,
- процессор расчета макета,
- процессор верстки-выключки,
- процессор составления указателей,

- процессоры ввода и вывода, процессор распечатки,
- монитор системы,
- процессор службы архива и библиотек,
- процессор подготовки фордов.

Связи обрабатывающих процессоров показаны на рисунке.



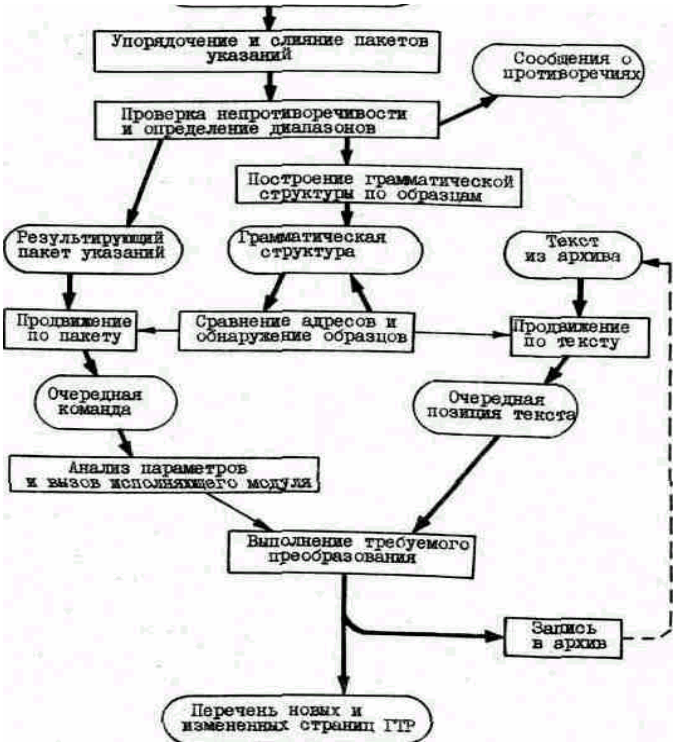
В системе предусмотрены два уровня архива (оперативный и неоперативный) и два типа библиотек (сокращений и фордов). Работа с архивами и библиотеками осуществляется специальным процессором, называемым процессором службы архива и библиотек.

Монитор системы является процессором, предназначенным для управления другими процессорами и организации прохождения работ через систему в той степени, в какой это не обеспечивается средствами операционной системы, под управлением которой работает САПФИР.

В мониторе сконцентрирована вся техническая информация о состоянии находящихся в системе заказов и подробностях запуска других процессоров.

Монитор взаимодействует с оператором ЭВМ и помогает ему в организации потока заданий при работе системы. Кроме того, при пополнении системы диалоговыми средствами именно на монитор придется организация диалогового режима.

Процессор и работы текстового редактирования. Строение процессора текстового редактирования учитывает, что указания для редактирования одного и того же текста могут поступать в виде нескольких пакетов команд. Когда реализуется параллельная корректура (приходящая от редактора, автора и т.д.), процессор производит слияние всех указаний и их упорядочение по возрастанию адресов их применения к исходному тексту.

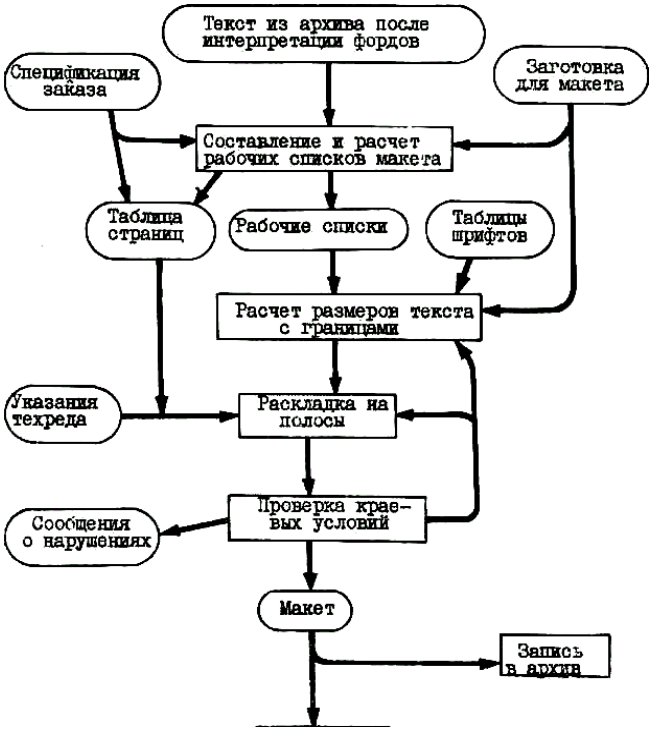


Пакетный режим осуществления корректирующих указаний предполагает, что вся адресация в них относится к одному и тому же состоянию текста, представленному ГТР.

Процессор проверяет полученные указания на их синтаксическую правильность, а также на то, чтобы для одного и того же места исходного текста не возникли противоречивые предписания по обработке.

В процессе анализа формируется финальный пакет указаний, который и будет применен к тексту, если не возникло противоречий. В противном случае выдаются сообщения, и пакет указаний возвращается пользователям на доработку. Исходный текст остается неизмененным. Одновременно с анализом непротиворечивости указаний выделяются образцы, если они имеются, и определяются диапазоны.

Процессор и работы макетирования. Исходной информацией для составления макета служат: текст после интерпретации в нем фордов, спецификация заказа и таблицы размеров шрифтов, используемых для выбранного в данном издании полиграфического оформления текста. Вся исходная информация берется из архива системы.



Макет как структура данных состоит из рабочего списка, представляющего собой таблицу, каждый элемент которой соответствует фрагментам текста, расположенным в порядке их следования, а именно: заголовкам рубрик, абзацам, блокам текста.

В том случае, если блок, такой как формула, расположен внутри текста абзаца, то в рабочем списке заводятся расположенные в их порядке элементы, соответствующие тексту начальной части абзаца, этому блоку и тексту завершающей части абзаца.

Для тех блоков текста, расположение которых определено нежестко (иллюстрации, примечания), заводится дополнительный список, связанный с основными двусторонними ссылками.

Каждый элемент основного и дополнительного списков содержит информацию о размерах соответствующего фрагмента текста, задаваемую максимальным и минимальным значениями, которую данный размер может принимать. Кроме того, в состав элемента переносятся все текстовые связи соответствующего фрагмента.

Таблица полос, получаемая в качестве результата, предназначается для ссылок на элементы рабочего списка, размещаемые на каждой данной полосе в результате макетирования. И, кроме того, для указания координат для расположения соответствующих фрагментов текста на полосе.

Процессор расчета макета строит основной и дополнительный рабочий списки, исходя из заготовки макета и текста полученных после интерпретации фордов. Для каждого элемента производится расчет размеров соответствующих фрагментов текста с использованием таблиц размеров шрифтов и информации из спецификации заказа.

Работа по дальнейшему расчету макета может быть факторизована по спусковым полосам. При расчете раскладки текста на полосы заполняется информация в элементах таблицы полос, и проверяются краевые условия, задаваемые технологическими правилами набора. Расчет части абзаца, помещаемой на полосу, производится с точностью до слова, и для каждой половины «разрезанного» абзаца вычисляются граничные значения его размеров.

Процессор планирует размещение иллюстраций и блоков на полосах по возможности ближе к тому месту текста, где на них имеются ссылки. При этом расчет числа строк в абзацах, для которых нужно сделать оборку, производится через значение занимаемой абзацем площади.

При обнаружении нарушения краевых условий делается попытка назначить фрагментам текста другие размеры в пределах рассчитанных границ. Этот процесс продолжается до тех пор, пока не будут устранены все неувязки. В случае если процессор исчерпал все заложенные в него возможности, выдается сообщение, запрашивающее указания от технического (художественного) редактора, текущее состояние макета запоминается, и процесс расчета прекращается вплоть до появления указаний. После завершения

расчета макета он может быть помещен в архив и/или передан процессору распечатки.

Заметим, что процессор расчета макета не производит фактического размещения текста на полосах в ходе итерационного процесса расчета, оперируя не с текстом, а только с макетом, объем которого несравненно меньше.

Процессоры и работы верстки. Верстка производится по макету, который, как и текст для верстки, берется из архива. Для каждой полосы последовательно производится расположение на полосе блоков и расположение текста внутри их. После определения размеров и формы места на полосе, предназначенного для заливки, производится вызов модуля заливки текста, который, при необходимости, в свою очередь, обращается к модулю реализации переносов.

Поскольку в процессе расчета макета была собрана глобальная информация о тексте и для каждого заливаемого фрагмента текста определены формат строки и число строк, в которое он должен быть залит, модель заливки опирается в ходе работы на вычисленное среднее значение переменного пробела, соответствующее указанным размерам фрагмента. При этом, при обнаружении неувязки, вызванной дискретностью слов при переносах, производится возврат по тексту до места, начиная с которого можно проводить заливку за счет выключающей способности данного отрезка текста, изменив среднее значение переменного пробела.

Процессор верстки-выключки организован таким образом, чтобы локализовать на компенсационной основе отклонения от ранее рассчитанного макета в пределах минимально возможного числа абзацев.

По завершении верстки происходит окончательное формирование полос, и текст каждой полосы, представленный в формате выходного интерфейса системы, записывается в архив. При необходимости может быть вызван процессор распечатки для выдачи точного, построчно совпадающего с окончательным текстом издания, оригинал-макета.

Интерфейсы системы. Для того чтобы сделать работу процессоров системы, не зависящей от специфических особенностей представления символов на устройствах ввода и вывода, а также для того, чтобы обеспечить возможность подключения в дальнейшем новых устройств, предусматриваются специальные средства отделения представлений символов в устройствах от представлений их в системе.

Для этой цели определяются ограничения на формат представления входной и выходной информации системы, называемые входным и выходным интерфейсами, соответственно.

Ограничение, определяющее входной интерфейс, заключается в том, что представление текста в его формате рассматривается как линейно упорядоченная последовательность символов базового алфавита (как видимых, так и невидимых).

Ограничение, накладываемое на формат представления информации в выходном интерфейсе, состоит в том, что текст представляется линейно упорядоченной последовательностью двухбайтовых полей, первым байтом которого является «представление» символа базового алфавита, а второй байт занят информацией о ширине символа, указанного первым байтом.

Те поля, второй байт которых указывает на нулевую ширину, служат для размещения в них представлений команд управления соответствующим устройством.

Ограничение, накладываемое на формат представления информации в файлах неоперативного архива, состоит в том, что текст заказов представляется на магнитной ленте в виде одного последовательно организованного файла с блокированными записями фиксированной длины. Первый блок файла должен содержать информацию о внутреннем строении текста, формат которой будет установлен при программировании. Эта информация должна заканчиваться специальным разделителем.

Сегодня (декабрь 2004 г.) эти материалы в первую очередь с очевидностью демонстрируют желание дать редакторам возможность работать почеловечески даже на тех аппаратных и технических средствах и пакетных режимах, которые были в наличии. Конечно, дальше опытной эксплуатации пакетная система редактирования на ЕС-1020 в операционной системе ДОС не пошла.

Однако уже при защите Эскизного проекта системы САПФИР на техническом совете Госкомиздата СССР акад. А.П. Ершова взял за жилетную пуговицу директор издательства ЦК КПСС «Правда» Борис Александрович Фельдман и сказал: «Я хочу, чтобы у нас тоже была такая система. И даже лучше». Так началась другая история — история проекта РУБИН.

ПРОЕКТ РУБИН

В конце 1976 г. комбинат «Правда», Вычислительный центр СО АН СССР и Новосибирский филиал ИТМиВТ АН СССР подписали «протокол о научно-техническом сотрудничестве», нацеленном на решение проблемы внедрения средств вычислительной техники для производственных процессов комбината.

Прежде всего, предусматривалось проведение системного анализа производственной деятельности комбината с позиций информатики и вычислительного дела. При этом имелось в виду, что результаты такого анализа должны будут стать основой для выработки технического задания на проведение проектных и опытно-конструкторских работ.

Начальный период. Первые контакты, общее ознакомление с комбинатом, выяснение круга вопросов, подлежащих изучению. После обсуждения интересующих «Правду» проблем было решено начать с общего обследования характера редакционно-издательских процессов, принятых в издательстве.

Эта работа выполнялась в качестве совместного социалистического обязательства ВЦ СО АН СССР и НФ ИТМиВТ АН СССР в честь 60-й годовщины Великой октябрьской социалистической революции, принятого по инициативе лаборатории экспериментальной информатики ВЦ СО АН СССР.

Подготовка. Выбор главного звена в проблематике, разработка методики системного анализа и ее согласование. Весной 1977 г. в издательство был отправлен нижеследующий документ.

ПРЕДВАРИТЕЛЬНЫЙ ЗАПРОС НА ИНФОРМАЦИЮ,
необходимую для составления плана
системного анализа комбината «Правда»

Нижеследующие вопросы относятся к системе работы с информацией, применяемой на «ПРАВДЕ» в настоящее время,

Предполагается, что ответы на них будут отражать реальное текущее положение дел, безотносительно к каким бы то ни было возможностям или ограничениям предстоящей машинной системы.

1. Источники информации, их особенности, классификация по значимости, объемы по источникам.
2. Способы поступления информации. Деление по типу носителя, объемы по каждому типу и другие характеристики.
3. Схема информационных потоков при поступлении информации. Точки входа. Предварительная реакция, критерии отбора и обработка. Как движется информационный материал. Где накапливается, где корректируется, уточняется, копируется, уничтожается, сортируется, регламентируется, визируется и т.п.?
4. Хранение информации. Формы хранения, типы и объемы хранилищ. Сроки хранения и от чего они зависят. Надежность хранения.
5. Отдельные важные факторы классификации информационных материалов. Секретность. Ее типы, способы и формы ее обеспечения. Иноязычная информация, — какие языки, особенности работы с ней. Другие факторы классификации, заслуживающие внимания.
6. Применение хранимой информации. Цели и смысл ее применения. Время жизни, форма и техника использования. Когда, кем и каким образом используются разные виды информации.

7. Пользователи информации. Их ранги и права. Частота обращения к архиву по типам пользователей. Цели обращения. Форма вопросов (запросов) и ответов — типовые примеры. Ориентировочное время обслуживания запросов.
8. Стиль взаимоотношений при передаче информации, степень формализации. Статистика использования по типам материалов, контроль пользования, протоколирование. Форма контроля и его уровни. История пользования, ее анализ (есть ли?), цели и выводы.
9. Стиль досье «Правды». Круг обязанностей службы досье. Н.Ф. Рогульская, деловая характеристика, примечательные профессиональные и личностные черты. Особенности характера и интеллекта. Особенности других сотрудников службы досье. (Ответы с согласия т. Рогульской)
10. Некоторый желаемый идеал в текущем контексте сложившихся отношений и техники работы с информацией (может быть несколько разных ответов).
11. Предвидимые и реальные нарушения в работе системы на данное время. Ранжирование по нежелательности. Последствия. Способы устранения. Терпимые сроки устранения. Меры, принимаемые по поводу нарушений. Реакция и пенальти. Несколько конкретных примеров.
12. Меры профилактики, применяемые для обеспечения информационной безопасности и системы работы в целом.
13. ЧП, которые бывали или могут быть. Чего не может (не должно) быть никогда? Что требует обязательной подстраховки?
14. Оценка существующей системы работы с разных точек зрения.
15. Степень соответствия возможностей системы работы реальным потребностям.
16. Какие виды и характеристики информации должны возрасти (уменьшиться), ускориться и т.п.?
17. Чем хороша существующая система работы? Чем она плоха?
18. Что хочется улучшить? Почему это трудно сделать?
19. Раздражающие моменты; терпимые, достаточно хорошие, совсем хорошие (комфортные).
20. Почему обязательно применение ЭВМ в системе «Правды»?
21. Отличия редакции «Правды» от других редакций. Связи с другими редакциями газет (по информации), связи с журнальным комплексом.

Ответы на поставленные вопросы предназначены только для исполнителей и позволят ориентировать предстоящий системный анализ на выяснение (установление) свойств будущей автоматизированной системы.

Сформулированные вопросы в значительной мере отражают наше текущее представление об источнике вопросов. Поэтому, если авторам ответов этот список вопросов покажется несбалансированным, то они могут дополнительно указать на те или иные факторы, не затронутые нами. Желательно, чтобы недостатки вопросов не повлияли на полноту и качество ответов.

Ответы желательно получить к концу марта сего года.

С уважением А.А. Берс, Г.Д. Чинин 07.02.77

На запрос был получен подробный и благожелательный ответ на 8 страницах и ряд дополнительных материалов. На ряде совместных семинаров ВЦ и НФ ИТМиВТ была выработана методика обследования. Далее выполнение работ проходило по следующим этапам.

Обследование. Работа выездной бригады совместно с сотрудниками комбината. 6 июня 1977 г. на территорию комбината «Правда» для детального изучения производственных процессов по выработанной методике высадился «десант СО АН СССР» в составе:

А.А. Берс, заведующий научно-исследовательской группой, ответственный исполнитель от ВЦ СО АН СССР;

Г.Д. Чинин, главный инженер, ответственный исполнитель от НФ ИТМиВТ АН СССР;

В.В. Грушецкий, младший научный сотрудник ВЦ СО АН СССР;

Ю.А. Первин, кандидат тех. наук, старший научный сотрудник ВЦ СО АН СССР;

П.К. Леонов, начальник лаборатории НФ ИТМиВТ АН СССР;

Ф.Р. Цанг, начальник лаборатории НФ ИТМиВТ АН СССР.

Каждый вооружен методикой и следующей инструкцией.

ПЛАН-ПРОГРАММА
информационного обследования редакции газеты «Правда»

1. Знакомство со структурой редакции и процедурами прохождения материалов в номер. Определение отделов и служб, подлежащих обследованию.
2. Для каждого из отделов:
   - 2.1. выяснение объема, характера, форм и сроков поступления входной информации;
   - 2.2. выяснение типов, объемов и сроков подготовки выходных материалов;
   - 2.3. определение информационных связей с другими отделами и объемов соответствующих информационных потоков;
   - 2.4. уяснение специфических особенностей подготовки информации;
   - 2.5. определение характера разделения работ внутри отдела.
3. Изучение характера работы и темпов обслуживания в справочно-библиографических отделах: внешнеполитической и внутренней жизни.
4. Знакомство с организацией справочно-информационных фондов и принципами их классификации.
5. Наблюдение хода работы в «часы пик».
6. Выяснение типов и объемов материалов и характера, объемов и срочности запросов.

7. Определение связей с другими отделами и размеров соответствующих информационных потоков.
8. Изучение движения материалов и процесса формирования очередных номеров газеты:
   8.1. объемы материалов от отделов;
   8.2. путь материала в номер: перепечатка, набор, проверка, визирование и т.п.;
   8.3. объем переделок по полосам в процессе подготовки номера.
9. Различие в материалах и объем переделки для двух выпусков одного и того же номера.
10. Организация и характер взаимодействия между редакцией и выпускающей службой в ходе подготовки номера.

6 июня 1977 г.

Зав, НИГ ВЦ СО АН СССР, А.А.Берс

Гл. инженер НФ ИТМиВТ, Г.Д. Чинин

Мы провели на комбинате пять интенсивно заполненных и очень интересных дней, встречаясь с высококлассными профессионалами, и затем вернулись в Академгородок для обработки материалов обследования, формирования основных гипотез и подготовки текста отчета. Естественно, что этот отчет готовился при большой поддержке и прямом участии сотрудников комбината «Правда», особая благодарность:

- генеральному директору комбината Борису Александровичу Фельдману — за постановку задачи и постоянный интерес к развитию работ;
- ответственному секретарю «Правды» Сергею Витальевичу Цукасову и начальнику Технического отдела издательства Владимиру Александровичу Тифенбаху — за чёткую ориентацию в делах комбината и редакции и предоставление режима наибольшего благоприятствования.

Далее воспроизведено основное содержание отчета о проведенном «Системном анализе производственных процессов по выпуску газеты «Правда». При этом надо, однако, сразу сделать оговорки в отношении широты охвата проблемы. Работа была начата по инициативе издательства в комбинате «Правда», и именно с него началось наше изучение комбината. Разработчики довольно быстро осознали, что комбинат в целом — это огромное всесоюзное объединение, издающее и печатающее практически все виды полиграфической продукции. Он втягивает в свой производственный цикл десятки типографий и корреспондентских отделений страны с разветвленной экспедиционной службой. В связи с этим — здесь потенциально виден очень широкий круг задач на компьютеризацию.

1. Общие вопросы

Для того чтобы сохранить принцип системности и не утратить ориентации в этом многообразии, необходимо было выделить стержень производственной деятельности, доступный для целостного охвата, наиболее актуальный для применения вычислительных машин и в то же время находящийся в центре производственной деятельности комбината.

Таким процессом нам представился выпуск газеты «Правда» и даже не собственно выпуск, имея в виду печатание тиража, рассылку матриц и передачу полос по фототелеграфу, а подготовка номера к выпуску, завершающаяся подписанием контрольного оттиска в печать.

Совершенно очевидно, что это более частный и достаточно четко выделенный процесс. В стороне остаются выпуск остальных газет, а также журналов и книг, цветная и графическая печать, экспедиция и подписка, административное управление работой комбината и другие более изолированные участки работы.

В то же время ощущение того, что вся многообразная деятельность комбината, так или иначе, подчинена или сопредельна этой главной задаче, пришло к нам очень быстро, и это чувство со временем только укреплялось.

Выпуск газеты с жестким суточным циклом, проблема отбора материала, требование абсолютной достоверности, необходимость постоянного ретроспективного анализа и обратной связи через письма читателей — все это делает выпуск процессом, наиболее требовательным к обеспечению эффективности, надежности и достоверности. А стало быть — первым кандидатом на изучение возможностей применения ЭВМ для электронной подготовки изданий.

Таким образом, редакция газеты «Правда» и ее производственные процессы стали главным объектом системного анализа. Сказанное, однако, не означает, что компоненты процесса, относящиеся к другим подразделениям комбината, остались «за бортом» Наоборот, там, где связь с выпуском газеты носит органический характер — прежде всего набор, верстка и правка, — соответствующие подразделения и процессы изучались со всей возможной тщательностью.

Основу системного анализа составлял взгляд на выпуск газеты как на процесс переработки информации. По нашему мнению, это делает анализ более объективным, а главное — более соответствующим характеру и целям дальнейшей работы. Указанный подход определил стиль дальнейшего изложения и употребляемую терминологию.

Материал организован следующим образом. Разделы со второго по четвертый — это систематическое и полное изложение результатов системного анализа. В пятом разделе сжато повторяются и поясняются основные выводы.

Целостность системного анализа требует лаконичности изложения, мы не тратили бумаги на обоснования там, где суждения опираются на точное

знание, и не подыскивали сомнительных доказательств интуитивным соображениям.

2. Редакция «Правды» как система обработки данных

Описание реально существующей организации посредством некоторой модельной системы всегда приводит к некоторому упрощению представления ее структуры и функций с риском опустить некоторые даже существенные детали.

Однако этот риск оправдан, поскольку без проработки такого обобщенного представления, легко можно очутиться в положении, когда «за деревьями леса не видно». Более того, эксплуатируя использованную метафору далее, можно утверждать, что если контуры «леса» очерчены правильно, то в них всегда можно будет найти место, чтобы «обратно посадить» какой-нибудь опущенный сорт «деревьев».

Это обобщенное представление позволяет явно выделить основные цели деятельности моделируемой организации, ее главные составные части и основные связи между ними.

Отметим сразу же, что задача описания существующих связей и движения информации между подразделениями редакции значительно упрощается в связи с тем, что этот процесс в большой степени регламентирован и упорядочен.

Существует утверждённый редколлегией «Порядок выпуска номеров», четкая система планирования газеты для разных временных интервалов и хорошо разделенные функции различных подразделении.

Основной целью работы редакции является, естественно, выпуск номеров газеты. Именно с точки зрения отношения к этим цели можно провести функциональное разбиение подразделений редакции.

Мы выделяем следующие типы подразделений.

1. Тематические отделы, основной функцией которых является подготовка материалов, помещаемых на газетные полосы, — это отделы:

- партийной жизни, пропаганды марксистско-ленинской теории, промышленно-экономический, сельскохозяйственный, литературы и искусства, культуры и быта, науки, школ и вузов;
- внутренней информации, военный, прессы, критики и библиографии, фельетонов, местных корреспондентов;
- международной информации, соцстран, европейских стран, американских стран, стран Азии и Африки;
- к ним же относятся группы специальных корреспондентов и политических обозревателей, а также, с некоторыми оговорками, и отдел иллюстраций.

2. Контрольно-справочные отделы, функцией которых является создание и поддержание информационных фондов материалов, предназначенных, как правило, не для помещения на полосу, а в качестве исходной информации для подготовки материалов тематическими отделами. На основе этих фон-

дов контрольно-справочные отделы обеспечивают поиск и подбор информации по запросам, а также проверку фактов в печатаемых «Правдой» материалах: отдел проверки, библиотека, справочно-библиографческий отдел международных отделов.

3. Отдел писем, стоящий особняком в силу специфичности, как обрабатываемой информации, так и способов работы с ней.

4. Службы выпуска, обеспечивающие интеграцию всех подготовленных материалов и аккумуляцию всей работы редакции на двенадцати полосах двух ежедневных выпусков газеты.

5. Редколлегия и секретариат, выполняющие функции планирования, руководства и контроля.

2. 1. Тематические отделы

Основная функция тематических отделов — подготовка материалов, помещаемых на полосы газеты. Объем обработки этих текстовых материалов определяется обычным порядком их подготовки, состоящим из следующих этапов:

- написания,
- перепечатки,
- вычитки,
- согласования в отделе,
- проверки фактов,
- перепечатки после правки,
- сдачи в набор,
- вычитки,
- проверки и правки после набора,
- правки и проверки (быть может, неоднократной) в процессе верстки.

Интегральная оценка объема перерабатываемой информации может быть произведена исходя из объема газеты.

Шесть полос одного номера имеют объем 170 Кбайт. Учитывая коэффициент зацепления информации в смежных выпусках, равный 1,2 (изменяется около 20% номера), получим общий объем, равный 204 Кбайт в день. По данным наборного цеха фактический объем набираемой информации составляет в среднем 220 Кбайт в день.

Поскольку перепечатка в отделах при этом не учитывается, то общий объем перерабатываемой тематическими отделами информации можно полагать вдвое большим, т. е. округленно — 0, 5 Мбайт в день.

В связи с подготовкой публикаций в тематических отделах необходим подбор материалов по темам и запросы этих материалов из информационных Фондов. Часть этих поисковых функций, естественно, выполняется внутри отдела, а другая часть формирует информационные потоки между тематическими и контрольно-справочными отделами (держателями информационных фондов)

Следующей функцией, выполняемой частично в тематическом отделе, а в основном контрольно-справочными отделами, является проверка фактического содержания публикуемых материалов, написания в них цитат, фамилий, географических названий и т.п.

Важной функцией тематических отделов, требующей информационной поддержки, является организация обратной связи от предыдущих публикаций с целью поддержания правильных пропорций в отношении охвата регионов страны, баланса критических и позитивных материалов и т.п. С этой целью необходимо иметь возможность оперативно отвечать на запросы: «Что, как и когда «Правда» публиковала по данному вопросу?»

Предыдущая функция тесно связана с проблемами перспективного и текущего планирования. Эта функция выполняется тематическим отделом совместно с секретариатом и редколлегией. Кроме того, в отделе необходимо отслеживать действенность газетных публикаций, своевременность получения ответов и т.п.

Подводя итоги, перечислим еще раз функции, которые должны обеспечиваться тематическим отделом:

- текстовая обработка,
- поиск и подбор материалов в информационных фондах,
- проверка фактов и написаний в публикациях,
- учет предыдущих публикаций «Правды» по данному вопросу,
- перспективное и текущее планирование,
- контроль действенности материалов.

2. 2. Контрольно-справочные отделы

Основными функциями контрольно-справочных отделов являются проверка публикуемых материалов и выполнение запросов тематических отделов по подбору информации на заданную тему.

Для того чтобы выполнять указанные функции, контрольно-справочные отделы ведут, пополняют и используют соответствующие информационные фонды. В связи с этим появляются функции по анализу поступающих материалов, их классификации и реферированию, а также поддержанию и изменению используемых классификаторов.

Общий объем поступающей по разным каналам (ТАСС, советские и иностранные газеты и журналы и т.п.) информации составляет около 3 Мбайт в день. Этот объем составлен из 1500—2000 отдельных материалов, для каждого из которых необходимо составить карточку-реферат. Полагая объем реферата равным в среднем 200 знакам, получим, что примерный объем ежедневно добавляемой справочной информации составляет около 300 Кбайт.

Эти цифры значительно (в 6–8 раз) варьируются в различных материалах, описывающих задания на разработку информационно-поисковых систем. Это, конечно, не означает изменение объема поступающей информа-

ции, просто в разных заданиях считается важным обеспечить обработку разных ее долей.

Фактически часть объема информационных фондов хранится и поддерживается не в контрольно-справочных, а в тематических отделах в соответствии с их профилем. Иногда такое разделение обуславливается разной степенью секретности материалов.

Отметим также, что классификаторы, на основе которых организуется структура информационных фондов, не являются постоянными. Разные рубрики в них появляются и исчезают с течением времени.

Если вернуться теперь назад, к основным функциям контрольно-справочных отделов: выдача информации по запросам и проверка фактов в публикуемых материалах, то можно увидеть, что, хотя обе эти функции опираются на содержание информационных фондов, характер работы с информацией в них существенно различается. Суть этих различий состоит не столько в том, что понятие факта чрезвычайно многообразно и, следовательно, весьма трудно формализуемо, сколько в совсем разной требуемой организации структуры фондов и характеристиках процесса поиска.

Как было сказано в ходе обследования в отделе проверки: «Фактом является все, что только можно проверить». Такая проверка проводится с использованием разнообразных справочников, картотек, словарей и т.п. Принципы идентификации факта в проверяемом тексте и поиска подтверждающих его материалов практически не сформулированы и существенно опираются на смысл текста.

Важность обязательности такой проверки объясняется, во-первых, официальным характером публикаций «Правды» и, во-вторых, например, тем, что в некоторых ответственных случаях поздно замеченная опечатка в одной букве приведёт к необходимости заново перепечатывать весь тираж выпуска газеты.

В связи с проведением корректуры и редактирования, возникающих вследствие проверки, контрольно-справочные отделы тоже выполняют функции обработки текстов. Отметим также, что необходимость в текстовой обработке возникает и при пополнении информационных фондов и составлении рефератов.

Суммируя, видим, что функциями, обеспечиваемыми контрольно-справочными отделами, являются: загрузка и сопровождение информационных фондов, составление реферативных карточек на входящие в них материалы, поиск и выдача информации по запросам, проверка фактов и написаний в публикациях, текстовая обработка информации.

2. 3. Отдел писем

Отдел писем является входом в информационную систему «Правды» в качестве одного из основных информационных потоков. Письма выделены в самостоятельный информационный поток в силу их специфических свойств, существенно выделяющих их среди других видов входной информации.

Среди важнейших специфических характеристик этого информационного потока следует отметить:

- дискретность, порционность информации; порция информации — письмо — служит одновременно как хранимой, так и обрабатываемой единицей информации;
- относительная равномерность поступления; эта равномерность ощутима не только на значительных промежутках времени — год, квартал, но и на оперативных отрезках — декада, неделя; даже суточные отклонения нельзя считать значительными;
- увеличенная степень ответственности редакции за каждый информационный объект — письмо; отмечается поступление каждого письма и, вообще говоря, путь прохождения и обработки каждого письма также должен контролироваться;
- особенности обработки письма; сотрудники отдела имеют дело с нестандартно оформленной и почти всегда рукописной информацией; при первичной обработке письма важно не формализуемое в таких условиях реферирование письма;
- трудоемкости обработки информации; эта особенность потока писем непосредственно вытекает из отмеченной выше ответственности редакции за письма и специфики их обработки, а также значительного объема обрабатываемой информации;
- динамичность базы данных; ежедневно система принимает для хранения и обработки значительный объем информации — примерно 1500 писем; одновременно из системы выводится (уничтожается) примерно такое же количество устаревшей информации;
- относительно несложное формальное описание единицы информации — письма; но важно отметить, что относительная простота идентификации письма сочетается с практически нереализуемым сегодняшними средствами формальным описанием семантики письма;
- отсутствие пересечений потока писем с другими информационными потоками; выходная информация отдела писем связана с другими отделами и службами; входной же поток полностью автономен.

Вместе с тем отдел писем обладает и теми общими чертами, которые характерны для большинства других отделов:

- значительный объем информации (отдел писем «Правды» регистрирует ежегодно 500 000 писем и по статистике последних лет это количество имеет тенденцию к увеличению);
- оперативность; поступающая в виде писем информация должна быть в тот же день зарегистрирована в отделе;
- ограничения (контролируемые отделом) на время обработки писем, направленных в другие отделы, и подготовку ответов;
- достаточно отчетливое (динамическое) разделение на обрабатываемую, или контролируемую, информацию и хранимую информацию; по

отделу писем установлены нормы времени на хранение *архивных* материалов;

- характерные для многих отделов проблемы хранения больших архивов; и сложность складирования больших бумажных объемов, доступ к которым должен быть легко реализуем в течение всего периода хранения;
- статистический анализ информации и динамическая рубрикация; в отделе писем эти характеристики особенно актуальны потому, что они непосредственно связаны с оперативной производительной загрузкой персонала отдела.

2. 4. Редколлегия и секретариат

Если отвлечься от задач оперативного контроля выпуска газет и чисто административного управления, то рабочие процедуры редколлегии и секретариата регламентированы существенно меньше по сравнению с производственными подразделениями, при этом из постоянных функций редколлегии и секретариата можно выделить следующие:

- координация работ редакции,
- перспективное и текущее планирование,
- контроль действенности публикаций газеты.

Говоря о взаимодействии редколлегии и секретариата с планируемой информационно-вычислительной системой комбината «Правда», можно считать их источником как регулярных, так и разовых информационно-поисковых предписаний.

Перечень регулярных запросов должен быть зафиксирован при проектировании, но нужно обязательно сохранить при этом и способность к формулированию разовых запросов и расширению их спектра.

Очень важным отдельным объектом проектирования для осуществления функций редколлегии и секретариата является подсистема ретроспективного анализа публикаций «Правды».

3. Требуемые системы обслуживания

В настоящем разделе рассматриваются свойства и характеристики, которыми должны обладать программные средства для того, чтобы обслуживать функции, выполняемые подразделениями редакции. Эти функции были выделены и описаны в предыдущем разделе.

Многие из этих средств достаточно хорошо проработаны к настоящему времени в системном программировании. В этом случае кратко характеризуются возможности в отношении интересующих нас функций, и дается ссылка на соответствующую литературу.

Когда это можно сделать, указывается система, пригодная для использования, или система, которая может послужить прототипом для адаптации ее к условиям работы в информационно-вычислительной системе (ИВС) «Правды». В тех случаях, когда реализация соответствующих функций на современном уровне информатики проработана слабо, мы пытаемся сфор-

мулировать основные направления, в которых необходимо инициировать соответствующее исследование.

3. 1. Система обработки текстов

Системы текстовой обработки прошли к настоящему времени путь от систем автоматизации набора до систем комплексной подготовки печатных изданий. Основные возможности систем текстовой обработки можно поделить на три группы:

- возможности правки текста как для редакционных целей, так и в связи с корректурой,
- возможности снабжения текста дополнительной структурой, осуществляемые с помощью нанесения на него требуемой разметки,
- возможности придания тексту необходимого полиграфического вида.

Возможности правки текста и его разметки могут осуществляться в настоящее время как в режиме диалогового взаимодействия с программной системой, что удобнее всего делать, используя экранные пульты, так и в пакетном режиме, достоинством которого является возможность сохранения привычного стиля правки по печатному тексту.

Внесение в текст разметки может использоваться как средство «развязывания» собственно текста и его полиграфического исполнения, также для обеспечения защиты и ограничения доступа к отдельным частям текста и для снабжения текста некоторой дополнительной информацией, существенной в процессе его обработки, но «невидимой» при его окончательном воспроизведении.

К такой информации могут относиться сведения об источниках фактов, ссылки на «хозяина» материалов и т.п.

В качестве основы текстовой обрабатывающей системы, которую можно использовать для ИВС «Правды» можно указать разрабатываемую в ВЦ СО АН СССР систему САПФИР, в которой заложен принцип отделения собственно текста от его полиграфического исполнения.

Эта система предусматривает реализацию всех перечисленных выше возможностей, но для включения в ИВС должна быть пополнена средствами составления макета и проведения газетной верстки с помощью графических и текстовых экранных терминалов, а также применением расширенных средств разметки в целях защиты утвержденных частей текста в ходе многократных правок при выпуске номера газеты.

3.2. Справочная система проверки фактов

Такой системы еще нет нигде. Вместо этого, поскольку проверку проводить надо, для ее осуществления пытаются использовать информационно-поисковые системы. Вследствие большого объема необходимого для этого информационного фонда, основные характеристики поиска при использовании информационно-поисковой системы (ИПС) для проверки фактов получаются плохими.

ИПС может предоставить для проверки какого-либо факта лишь целиком документ, который этот факт подтверждает. В результате коэффициент шума при поиске оказывается большим, так как пользователю предъявляется много избыточной для него информации, среди которой он должен сам найти нужную.

С другой стороны, вследствие необходимости обеспечения полноты поиска по отношению к заданному факту, время ответа системы резко возрастает с увеличением объема ее информационного фонда, а формулирование необходимого поискового предписания оказывается весьма громоздким и длительным.

Поэтому приемлемым в настоящее время путем применения ИПС для целей проверки фактов представляется факторизация информационных фондов, т.е. разбиение их на тематически сильно связанные отдельные части. Внешние связи между такими частями могут оказываться достаточно слабыми, таким образом, чтобы объем каждой из этих частей по порядку был бы равен 104–105 единицам поиска. Кроме того, для такой системы необходимо использовать специально подобранный язык диалогового формулирования поискового предписания. Эти меры позволили бы временно попользовать какую-либо из существующих поисковых систем в качестве прототипа для адаптации в качестве системы проверки фактов в ИВС «Правды». Это дало бы необходимый резерв времени для научных исследований, направленных на разработку принципов построения тезауруса фактов и языка запросов подлинной справочной системы проверки фактов.

3.3. Информационно-поисковая система внешней информации

Если в соответствии с предыдущим считать, что проверка фактов в публикациях будет осуществляться специально предназначенной для этого системой, то работы, которые должны будут обеспечивать в рамках ИВС «Правды» ИПС внешней информации, сводятся к следующим этапам:

- загрузке и поддержанию информационных фондов,
- заполнению и поддержанию реферативной части (индекса) системы,
- обеспечению диалогового формирования поискового предписания,

т.е. обычным функциям, на которые рассчитаны ИПС. Некоторым дополнительным свойством, которое надо принять в расчет при подборе прототипа для адаптации, является необходимость обеспечения средств модификации классификатора.

Для ЕС ЭВМ имеются две информационно-поисковые системы «ОКА» (прототип — IMS) и «КАМА» (CICS), которые могут рассматриваться в качестве возможных прототипов для адаптации в ИВС «Правда». Кроме того, в связи с большим объемом требуемых информационных фондов необходимо провести исследования возможности их факторизации.

ИПС напечатанных в газете материалов

Как уже говорилось, ежедневный объем двух выпусков «Правды» составляет 204 Кбайт, т. е. 72 Мбайт в год. В связи с очень большой потребно-

стью в создания ИПС, обеспечивающей быстрый доступ к предыдущим публикациям «Правды» и их анализ, а также сравнительно небольшим объемом информационного фонда, имеет смысл построить эту ИПС в качестве действующей модели ИПС внешней информации.

Такая модельная система обеспечит:

- удовлетворение первоочередных нужд редакции,
- разработку и уточнение принципов построения классификатора материалов и способов его модернизации,
- создание базы для обучения сотрудников редакции и накопления опыта эксплуатации,
- проверку принципов построения и удобства использования языка формирования поисковых предписаний.

3. 4. Система отдела писем

С точки зрения функций информационного и программного обеспечения, подсистема хранения и обработки писем должна строиться как ИПС с развитыми средствами защиты, самоизмерения, обеспечения надежности и динамики. Этап технического проектирования подсистемы позволит выбрать одну из существующих СУБД («Ока», «Кама», «Набоб», «Банк», «Синбад» и др.) в качестве прототипа и определить степень соответствия этого прототипа условиям эксплуатации его в «Правде». Одновременно с этим будут сформулированы требования к дополнительным, специфическим средствам ИПС хранения и обработки писем.

3.5. Системы планирования и контроля

Программные средства, реализующие указанные функции, входят как подсистемы практически в каждую из автоматизированных систем управления, широкая программа создания которых развернута в настоящее время в стране. Поэтому основной задачей здесь будет подбор прототипа и его адаптация к требованиям ИВС «Правды».

3. 6. Операционные системы ЦКП

Реальное применение большинства из рассмотренных выше функций потребует использования обеспечивающих их систем в режиме коллективного доступа с достаточно малым временем реакции. Для этого необходимо обеспечить работу этих систем в рамках и под управлением операционной системы центра коллективного пользования (ЦКП).

Математическое обеспечение типового ЦКП для ЭВМ отечественного производства (в том числе ЕС ЭВМ) разрабатывается в настоящее время в ВЦ СО АН СССР. Одним из основных принципов принятого подхода к архитектуре ЦКП является сочетание централизованных вычислительных мощностей с сетью периферийных центров обработки (ПЦО), предназначенных для локальной работы с информацией. Как будет видно из дальнейшего, такое строение вычислительной сети оказывается весьма подходящим для предлагаемых структуры и организации разработки ИВС «Правды».

4. Результаты системного анализа

Здесь формулируются предложения о целях, функциях, составе программного обеспечения и требуемой конфигурации аппаратных средств будущей ИВС «Правды». Эти предложения представлены совокупностью четырех взаимосвязанных гипотез:

- о целях,
- о функциях,
- об оборудовании,
- об организации разработок.

Предлагая некоторый принципиальный путь организации ИВС, эти гипотезы носят сугубо предварительный характер.

Они выдвигаются в настоящее время, для того чтобы:

- зафиксировать на возможно более ранней стадии общее видение контуров системы в целом;
- послужить средством закрепления тех впечатлений, которые возникли в ходе обследования и предшествовавших ему контактов с издательством «Правда» и которые впоследствии уже трудно будет восстановить, они будут «забиты» многочисленными деталями, возникающими в ходе проработки;
- вызвать (спровоцировать) реакцию пользователей и послужить основой для обсуждения, материалы которого смогут стать исходными для подготовки научно-технического задания на систему.

4.1. Гипотеза о целях

Главной целью создания ИВС «Правды» является обеспечение всех работ, проводимых в связи с выпуском номеров газеты — основной цели работы редакции.

Создаваемая система должна быть надежна и удобна в эксплуатации, в максимальной степени соответствовать сложившимся методам работы пользователей и выработанному за многие годы в редакции порядку работы.

С другой стороны, система должна в наименьшей степени зависеть от содержания работ и специфических «правдистских» процедур с тем, чтобы ее впоследствии можно было перенести для обеспечения работы редакции других газет (и журналов), обслуживаемых издательством «Правда».

Поскольку основным содержанием редакционно-издательских процессов является переработка текстов, то предлагается (см. гипотезу о функциях), чтобы центральной составляющей ИВС, на которую замыкаются и через которую осуществляются все связи и информационные потоки от других компонент ИВС «Правды», была бы система текстовой обработки.

Чтобы обеспечить требуемое малое время реакции системы при осуществлении ее функций в режиме коллективного доступа и для согласования системы с принципами организации операционной системы ЦКП, предлагается (см. гипотезу об оборудовании), чтобы аппаратное обеспечение пред-

ставляло собой сеть малых машин, соединенных с центральным мощным вычислительным комплексом.

Предполагается (см. гипотезу об организации разработок) такой порядок проведения работ и приобретения оборудования, чтобы обеспечить пользователей системы сначала небольшим, но полезным составом функций, расширяемым и усложняемым впоследствии, по мере поступления оборудования и прогресса в разработке программных систем.

4. 2. Гипотеза о функциях

Перечислим ещё раз функциональные системы, входящие в состав ИВС «Правды».

5. Система обработки текстов, включающая средства
   - ♦ редактирования и правки,
   - ♦ структурной разметки,
   - ♦ полиграфического оформления и верстки.
6. Информационно-поисковые системы
   - ♦ внешней информации,
   - ♦ публикаций газеты,
   - ♦ обработки писем.
7. Справочная система проверки фактов.
8. Система планирования и контроля.
9. Операционная система ЦКП, обеспечивающая работу всех составляющих ИВС и ее целостность как программного комплекса.

Однако для того чтобы ИВС «Правды» была бы функционально целостной системой, а не просто набором своих компонент, необходимо выделение его главного функционального компонента, через который бы замыкались все ее информационные потоки.

В связи с тем, что основная деятельность редакции выражается в форме подготовки текстовых материалов, такой центральной компонентой может быть только система текстовой обработки.

Дело в том, что основным недостатком «ручного» способа подготовки текстов является необходимость многократной сверки и вычитки текстов в ходе их редактирования. В то же время, при использовании для этой цели ЭВМ, однажды введенный в систему текст материализуется и, следовательно, неизменяем с такой же степенью надежности, как и текст, отлитый в строки на линотипе. С другой стороны, текст в памяти ЭВМ еще полностью свободен от любых особенностей своего полиграфического воплощения (шрифта, формата набора и т.п.).

Кроме того, находящийся в системе текст может быть снабжен (невидимыми по отношению к его окончательному воплощению) ссылками на источники информации, указаниями о необходимой защите и другой справочной информацией, которую необходимо учитывать в процессе его подготовки, согласования и визирования.

Система текстовой обработки позволит непосредственно выводить подготовленные тексты и использовать их при сборке (верстке) номера. Система обеспечит полную надежность и неизменность текстов в процессе их обращения внутри нее.

В связи с этим необходимо осуществлять ввод материалов в систему на возможно более ранней стадии их подготовки и использовать средства текстового редактирования для их правки с самого начала, что значительно повысит удобства работы для большинства сотрудников редакции.

Отметим также, что в состав работ, обеспечиваемых информационно-поисковыми и планирующими системами, крупной составной частью входит переработка информации в виде текстов и формирование их выходного вида. Текстовое редактирование является также существенной компонентой подготовки информации к включению ее в состав информационных фондов и подготовки карточек-рефератов, образующих поисковый аппарат ИПС.

Средства разметки системы текстовой обработки позволяют организовать непрерывную связь между функциями, осуществляемыми тематическими, контрольно-справочными отделами и службой выпуска.

Например, еще в ходе подготовки материала автором (в тематическом отделе), он может заранее выделить те места публикации, которые в случае необходимости при верстке могут быть сокращены в первую (вторую и т.д.) очередь. Более того, он может задать в тексте разные способы выражения отдельных фраз, что поможет сохранить хороший стиль изложения в случае сокращения. Система верстки может автоматически использовать эту разметку, избавляя работников службы выпуска от возможности сделать ошибку, обусловленную недостаточным проникновением в смысл написанного.

Встроенное в систему текстовой обработки средство обнаружения всех вхождений в текст некоторого образца позволяет обеспечить автоматическую проверку некоторого (не очень большого) множества написаний для ответственных названий, фамилий и т.п.

С другой стороны, средства разметки могут быть использованы для того, чтобы выделить в текстах утверждения, требующие проверки перед публикацией. Текст с такой пометкой может не пропускаться системой верстки на полосу до тех пор, пока эта разметка не будет снята, что сможет быть сделано только специально уполномоченными на то ответственными работниками. Те же средства разметки смогут обеспечить принятый порядок согласования и визирования публикуемых материалов.

Другими словами, использование системы обработки текстов как центральной компоненты ИВС «Правды» обеспечит однородность принципов работы с текстами, замкнутость этой работы внутри системы, т.е. в конечном итоге, удобство общения пользователей с системой и их полное освобождение от необходимости знать, как соответствующая работа делается внутри нее.

4. 3. Гипотеза о реализации отдела писем

Сложность проектируемой подсистемы и возможности разделения проекта на этапы, имеющие самостоятельное методическое, организационное и экономическое значение, делает целесообразным выделение нескольких очередей реализации.

Первая очередь автоматизированной подсистемы хранения и обработки писем предполагает разработку системы управления базами данных со всем комплексом управляющих программ. Из перечня проблем первой очереди исключаются лишь неисследованные проблемы использования естественного языка общения пользователя с системой в новой предметной области.

В таких рамках первой очереди подсистемы остаются проблемы разной степени проработки. В частности, неисследованными являются математическое и техническое обеспечение микрофишной техники в условиях подсистемы писем. С другой стороны, не вызывает сомнений ценность системы, которая берет на себя все функции хранения, обработки, контроля и защиты писем с сохранением бумажного архива отдела. Поэтому и в рамках первой очереди проекта имеет смысл выделить два этапа.

На первом этапе реализуется система, в которой библиотека писем — часть архива, хранящая тексты писем, — остается на бумажных носителях. При этом можно не менять существующую организацию хранения и, следовательно, временно сохранить годовой срок хранения писем в архиве. Это не скажется на проектируемых размерах базы данных — хранилище паспортов и аннотаций на машинных носителях.

Поэтому даже на начальном этапе первой очереди подсистемы банк данных будет рассчитан на требуемый объем 1000 Мбайт, что обеспечит хранение паспортов писем и аннотаций в течение 5 лет.

За время реализации первого этапа будут детально решены методологические вопросы использования аппаратов микрофиширования и визуализации микрофиш, подготовлена техническая база, исследовано и отлажено математическое обеспечение микрофиширования и визуализации.

Вторая очередь подсистемы. Ко второй очереди подсистемы писем отнесены языковые проблемы системного программирования. Представляет известный научный интерес использование естественного языка общения с машиной в системах подобного размера. Однако еще большее значение имеет скачок в повышении качества управления отделом писем — скачок, который будет обеспечен возможностью оперативного обращения к системе с нерегулярными запросами.

Вторая очередь подсистемы определяется не временно́й этапностью работ, а содержанием исследований. Не исключается и такая организация разработок, при которой языковые исследования в подсистеме писем могут быть завершены не позднее, а одновременно с разработками первой очереди.

К задачам второй очереди можно будет отнести организацию поиска писем по содержательным признакам на основе анализа текста аннотации или ее ключевых слов. В реализации первой очереди достаточно ограничиться поиском по кодам признаков, отмечаемых оператором в паспорте письма при его формировании. Такое ограничение позволит построить нетрудоемкую ИПС, удовлетворяющую сегодняшним требованиям отдела писем.

4.4. Гипотеза об организации разработок

Период создания больших программных систем измеряется годами, причем в настоящее время хорошо известно, что этот период не поддается существенному уменьшению за счет вовлечения в работу большого числа исполнителей, в силу внутренней сложности программных систем самих по себе.

С другой стороны, представляется важным, чтобы система как можно раньше начала служить пользователям, выполняя сначала хотя бы ограниченное множество основных своих функций. Для некоторых из них допустимо также осуществление и запуск в действие временно работающих средств, позволяющих выгадать время на разработку окончательного решения.

Представляется возможным осуществить такую поэтапную процедуру создания и для рассматриваемой информационно-вычислительной системы (ИВС) газеты «Правды».

Для этого первым этапом должна быть установка в отделах редакции сначала в автономном режиме мини-ЭВМ, которые впоследствии будут играть роль ПЦО (первичного центра обработки, см. гипотезу об оборудовании в 4.5).

Эти мини-ЭВМ должны быть снабжены программным обеспечением, выполняющим функции текстового редактирования и структурной разметки из системы обработки текстов. С их помощью начнется подготовка, редактирование и правка всех подготавливаемых для публикации материалов в тематических отделах. При этом подготовленный и отработанный в отделе материал передается в набор в виде перфоленты, выдаваемой ЭВМ по команде по завершении его обработки.

Сам набор и верстка на первом этапе продолжают проводиться по существующей технологии (при помощи управляемых перфолентой линотипов и на талерах).

Последующая правка набранных материалов может проводиться тоже при помощи мини-ЭВМ отдела, которая выдает в этом случае перфоленту для изготовления новых строк для заборки.

На этих же мини-ЭВМ устанавливаются подсобные информационные фонды тематических отделов (мини-досье), и на их материале ведется отработка диалоговых процессов формирования поисковых предписаний для ИПС (выработка языка запросов, разработка классификаторов и т.д.).

Мини-ЭВМ контрольно-справочных отделов используются как для обработки входов в будущие ИПС, так и для подготовки (с помощью средств текстового редактирования) содержания элементов информационных фондов ИПС (карточки-рефераты, загружаемые материалы и т.п.).

В частности, поскольку подготовленные для публикации материалы газеты «Правда» уже оказываются на машинном носителе, то они используются в этой работе без дополнительной подготовки к вводу.

На этой стадии обмен информацией между мини-ЭВМ происходит посредством физического переноса машинных носителей (гибких магнитных дисков или кассет с лентой).

Несколько более мощные мини-ЭВМ службы выпуска, снабженные алфавитными и графическими дисплеями, используются для отработки средств верстки номера по макету и, в частности, для целей пересчета макета, чтобы помочь найти простой и быстрый путь, когда нужно сделать переверстку.

Следует отметить, что хотя объем программного обеспечения, который нужно выполнить для первого этапа, составляет лишь малую часть общего объема программного хозяйства ИВС, значение этого первого этапа в ходе внедрения системы исключительно велико.

Дело в том, что для пользователей именно в этот период и возникнет ИВС «Правды», которая существенно затронет способы их работы, стиль подготовки материала и дисциплину взаимодействия отделов редакции.

Проведение подготовки текстов на ЭВМ потребует перехода от листа бумаги и ручки к необходимости и умению использовать клавиатуры пультов машины и освоение методики правки за экранным пультом.

Представляется, что такое изменение характера работы будет психологически непростым и потребует определенного времени, чтобы привыкнуть к нему, убедиться в удобстве и облегчении выполнения работы с помощью ЭВМ и научиться эксплуатировать предоставляемые ими возможности.

В переходный период возможно разделение работы по правке между сотрудником отдела (автором), ведущим правку «по-старому» пером на гранках и оператором («машинисткой»), вводящим ее в ЭВМ. Однако при этом сохранится многоступенчатость работы с текстом и необходимость вычитки текста после исправления, т.е. окажутся неиспользованными именно те возможности и преимущества, которые может обеспечить применение ЭВМ.

Всё дальнейшее развитие системы уже не будет качественно изменять характер работы для пользователей — просто время от времени они будут узнавать, что множество выполняемых для них услуг системы увеличилось, или замечать, что скорость обслуживания возросла.

Второй этап связан с запуском центральной машины системы под управлением ОС ЕС и постановкой на неё существующих в составе программного обеспечения ЕС ЭВМ информационно-поисковых систем «ОКА» и/или «КАМА». В это время начинается использование ИПС публикаций «Правды» и одновременно использование её как действующей модели для обработки

большой ИПС внешней информации. Мини-ЭВМ контрольно-справочных отделов работают как удаленные абонентские пункты. Одновременно отрабатываются системы отдела писем и планирования.

На третьем этапе производится интеграция всех ЭВМ в единую систему, работающую под управлением ОС ЦКП. В этот момент прекращается перенос машинных носителей для обмена информацией между ЭВМ, так как все они оказываются связанными в единую сеть.

В состав программного обеспечения вводится полная система текстовой обработки. Происходит (невидимое пользователю) перераспределение работ по текстовому редактированию между центральной ЭВМ и периферийными машинами.

Начинает использоваться автоматизированная система верстки номера с выходом на фотонаборный автомат и фотополимерные печатные формы.

Здесь мы остановимся в изложении данной гипотезы, поскольку дальнейшее развитие системы может идти за счет замены средств, использованных временно, специально разработанными компонентами с улучшенными характеристиками, т.е. обычным для существования программных систем путем.

Перечислим положительные и отрицательные стороны предлагаемого порядка поэтапной разработки и внедрения системы.

Достоинства:

- возможность постепенной закупки и установки оборудования,
- постепенность обучения пользователей и перехода от старых форм работы к новым,
- возможность учета опыта эксплуатации в ходе разработки,
- тесное взаимодействие разработчиков и потребителей в ходе создания системы.

Недостатки:

- перманентность процесса обучения, вызываемая последовательным вводом в строй новых функций,
- потенциальная опасность возникновения психологического сопротивления пользователей, вызванного тем, что временно используемые средства обеспечения некоторых функций могут оказаться менее удобными в работе, чем их последующие специально разработанные варианты,
- тесная связь разработчиков с потребителем и необходимость постоянного сопровождения системы разработчиком в течение длительного времени.

Общим основанием наличия этих недостатков является тот известный факт, что всякая сложная программная система «живет лишь постольку, поскольку она изменяется»

Скоро сказка сказывается, да не скоро дело делается.
Под лежачий камень вода не течет.
Пословицы

4.5. Гипотезы об оборудовании

Как уже говорилось, все программные составляющие ИВС должны работать под управлением операционной системы в рамках центра коллективного пользования. Известно, что при централизованной организации возникают большие трудности в обеспечении приемлемого времени реакции системы в диалоговом режиме при большом числе пользователей и больших объемах информационных фондов.

Так, в полностью централизованной системе ИБМ для газеты «Нью-Йорк Таймс» время ответа достигало 15 минут. Другой крайностью в организации структуры системы является распределение всех ее функций между большим числом малых машин.

В этом случае существенно усложняются вопросы организации и поддержания больших информационных фондов и значительно усложняются программные компоненты, обеспечивающие связь и взаимодействие между отдельными машинами системы, что потребует разработки нового типа операционной системы.

На самом деле, в число работ, которые должны обеспечиваться ИВС, входят как работы, осуществляющие локальную переработку небольших порций информации, так и работы, связанные с информационными фондами в целом и требующие для своего осуществления больших вычислительных мощностей.

Можно объединить достоинства обоих рассмотренных выше подходов к архитектуре системы без присущих им недостатков, организовав ее в виде системы периферийных центров обработки (ПЦО), соединенных с мощным центральным вычислителем и большой общей внешней памятью.

Такой ПЦО, т.е. мини-ЭВМ, может обслуживать один–два тематических отдела, выполняя для них функции:

- ввода и правки текстовых материалов,
- осуществления запросов к локальному информационному фонду,
- текущего планирования работы отдела,
- контроля действенности публикаций,
- связи для запросов и справок к центральному информационному фонду,
- передачи подготовленных отделом материалов в систему, обеспечивающую службу выпуска (для верстки),
- обеспечения оперативной связи с контрольно-справочными отделами, службой выпуска и другими подразделениями редакции в ходе подготовки номера.

С другой стороны, такие крупные системы, как ИПС внешней информации и система проверки факсов, а также сводная верстка номера будут использовать, главным образом, вычислительные ресурсы центральной части системы.

Наиболее вероятным кандидатом для центральной вычислительной машины является одна из старших моделей ЕС ЭВМ. При этом центральная установка может состоять, для обеспечения надежной и бесперебойной работы, из нескольких таких процессоров с подключенной к ним достаточно большой системой внешней памяти на магнитных дисках и магнитных лентах.

К сожалению, следует заметить, что пригодных для полиграфического применения периферийных устройств и малых машин, которые были бы подходящими для использования в качестве ПЦО, наша промышленность не выпускает. Поэтому для выбора экранных пультов, графических дисплеев, устройств распечатки и мини-ЭВМ приходится обращаться к зарубежным образцам.

По-видимому, требуемой очень высокой надежности функционирования системы и совершенно обязательных легкости и комфорта общения с ней пользователей можно добиться сочетанием разных машин: применяя отечественную аппаратуру для основных вычислительных мощностей (центральные процессоры, каналы, оперативная и внешняя память и т.д.) и новейшие зарубежные устройства (управляемые микропроцессорами и поэтому весьма гибкие и перенастраиваемые) как периферийные и терминальные.

> ***Сократ:*** *«Как получается, что ты продаешь больше панцирей, чем другие, хотя делаешь их не более прочными и не более роскошными?»*
>
> ***Оружейник:*** *«Потому, что я делаю их более пропорциональными».*
>
> ***Сократ:*** *«Но ведь бывают непропорциональные фигуры. Как можешь ты делать пропорциональные панцири для непропорциональных фигур?»*
>
> ***Оружейник:*** *«А я их подгоняю. Панцирь по мерке и есть самый пропорциональный».*
>
> *Платон Диалоги. Т.1*

Грубо говоря, представленный системный анализ состоял из двух этапов: изучение работы комбината и формирование гипотезы об ИВС для газеты «Правда». При этом главным впечатлением на первом этапе было восхищение высокой степенью организации производственных процессов, большой четкостью в разделении функций и ответственностью, строгим регламентом работ, очень высокой индивидуальной квалификацией сотрудников.

Комбинат успешно и эффективно решал свою задачу — выпуск и распространение ряда главных ежедневных всесоюзных газет, особенно если

учесть высокое профессиональное качество газеты, достоверность официального издания и большие дополнительные трудности, связанные с долготной протяженностью территории Советского Союза при западном положении комбината.

Ощущение того, что при этом совершенстве и слаженности просто негде «воткнуться» с такими радикальными новшествами как ЭВМ и безбумажная обработка текстов, неоднократно приходило к разработчикам анализа и явно обнаруживалось у сотрудников «Правды».

При более глубоком знакомстве, однако, обе стороны сошлись на том, что эти ощущения не только отражают, и то лишь частично, существующее положение дел, но и формируют психологический барьер, скрывающий серьезные проблемы завтрашнего дня.

В заключительной части отчета А.П. Ершовым написано:

Постараемся просуммировать факторы, толкающие на поиск новых форм организации производственных процессов в газетном деле так, как они стали для нас видными в ходе анализа.

1. Эффективность и слаженность коллектива «Правды» носят в значительной степени уникальный характер, отражающий ее особое положение среди средств массовой информации СССР и достигаемый отбором сотрудников с уровнем квалификации выше среднедостижимого.

2. Для многих подразделений «Правды» характерен предельный режим работы, изнуряющий сотрудников и не оставляющий места «запасу прочности» или способности к изменениям. Этот предельный режим подтверждается признаками энтропийной эрозии в досье некоторых отделов и информационных фондах, а также наличием некоторого числа в буквальном смысле незаменимых работников, отсутствие которых немедленно сказывается на показателях работы.

3. В то же время созревает необходимость создания значительного производственного ресурса газеты, вызванного, главным образом, ходом времени, общим научно-техническим прогрессом, количественным и качественным ростом общественно-политических и экономических форм жизни государства и всего мира в целом.

4. Более конкретно эти веления времени выдвигают следующие задачи (без претензии на упорядочение этих задач по значимости):

- увеличение объема газеты,
- увеличение числа и разнообразия выпусков газеты,
- повышение степени оперативности ряда рубрик,
- увеличение объема ретроспективной информации в связи с ходом времени,
- резкое увеличение объема и усложнение текущей информации, относящейся к международной жизни,

- улучшение деятельности газеты, связанной с письмами читателей,
- повышение качества и эффективности газеты как источника установочной и аналитической информации,
- поддержание складывающегося международного уровня полиграфического качества газеты.

5. Немаловажное значение имеет улучшение экономических показателей выпуска газеты.

6. Наконец, новые формы работы «Правды» должны стать тиражируемыми, быть образцом или прототипом повышения уровня работы других всесоюзных и республиканских редакций.

Естественно, что вычислительная техника далеко не всегда является единственным средством решения этих задач, однако вопросы ее применения играют ведущую роль, так как затрагивают основной компонент всей деятельности «Правды» — работу с текстом.

Если, следуя этому положению, выразить несколько утрированно главную идею авторов анализа, то она состоит в том, что мы с помощью ЭВМ стремимся исключить из производственных процессов по выпуску газеты перо и карандаш как орудие труда газетчика и ношение бумаги как форму информационной связи между сотрудниками и подразделениями.

Тем самым мы в некотором смысле выдвигаем на первый план не столько установку комплекса некоторого оборудования или создание конкретной информационно-поисковой системы, сколько реализацию некоторой общей формы работы сотрудников в виде машинной обработки текстов, для какой бы цели эта обработка ни применялась. Главное в этом подходе — это материализация текста, ликвидирующая «перевалочные работы» и перепроверки при его транспортировке.

Другая важная идея, выдвигаемая разработчиками анализа, — это принцип постепенности и движения от периферии к центру в разработке ИВС «Правда».

Установка терминалов на столы редакционных и издательских работников — уже это достаточно серьезное потрясение сложившихся основ профессиональной работы и трудовых навыков. Во всем остальном система должна быть очень пластична и приспособляема к конкретным нуждам производственных отделов. Она должна быть удобна в работе не когда-нибудь в будущем, а немедленно; не для каких-то гипотетических и специально обученных людей, а для тех, кто работает на комбинате сейчас.

Внедрение системы достаточно долго не должно нарушать сложившихся организационных форм работы и взаимодействия подразделений. Этой цели отвечает предложение начинать работу с развертывания отдельских подсистем. Залогом же последующей интеграции будет служить закладываемая с самого начала информационная совместимость подсистем и унификация методов обработки текстов.

Третье существенное положение состоит в выделении среди отдельских подсистем модельной подсистемы, которая, при всей ее ограниченности,

будет с технической точки зрения прототипом по всем серьезным компонентам разработки (организация хранения и поиска информации, выбор базовой системы, защита и восстановление после сбоев, язык информационно-поисковых предписаний, программное обеспечение терминального оборудования и т.п.). Возможным кандидатом при разработке такой модели могла бы быть, например, подсистема ретроспективного анализа публикаций «Правды».

Коснемся, наконец, соотношения предлагаемого подхода с другими известными системами компьютеризации газетной работы. Не затрагивая всех аспектов сопоставления, упомянем лишь о системах «Нью-Йорк Таймс» и об эволюции взгляда на системы в наших глазах, начиная с первых контактов с «Правдой».

Система ИБМ, предоставляя однородный сервис строго централизованного комплекса и, несмотря на высокий уровень разработки специального программного обеспечения, оказалась неудобной и неэффективной в работе. С нашей точки зрения объективным недостатком подхода была громоздкость базового программного обеспечения, умноженная на перецентрализацию информационных фондов и работы с ними, и все это в сочетании с недостаточным учетом специфики отделов редакции.

Система фирмы Харрис — полностью децентрализованная — представляет собой другую технологическую крайность. Она, может быть, и пригодна для «Нью-Йорк Таймс», но, по-видимому, неприемлема в условиях «Правды». Отсутствие взаимодействия между отделами, полностью отвечающими за заполнение выделенного «окна», совершенно другие условия композиции и верстки номера (многополосность, большое количество рекламы, значительное число полос с однородной информацией типа объявлений или курсов акций).

Все эти специфические условия, допускающие интенсивную децентрализацию, не позволяют идти по пути буквального восприятия организации системы фирмы Харрис.

В первоначальной постановке подача материала о потребностях «Правды», возможно под влиянием переговоров с ИБМ в 1975 г., также отражала тенденцию централизованной обработки информации, поступающей в «Правду».

В то же время наш системный анализ показал следующее.

- Отдел в «Правде» является реально действующей операционной единицей и имеет четко очерченный круг вопросов, относящихся к его компетенции.
- Входная информация, поступающая в «Правду», как правило, имеет конкретного адресата. Централизованная коммутация входных информационных потоков практически отсутствует или носит чисто технический характер. Внутренняя циркуляция информации также обходится в основном без центрального диспетчирования.

- В настоящее время проблема механизированной обработки входной информации (если исключить процедуры регистрации) не актуальна. Каждый входной материал *прочитывается,* т.е. опосредствуется *неформализуемой* интеллектуальной работой сотрудников отделов. Удельный вес информации, поступающей сразу на электронных носителях (телетайп, телеграммы) пока сравнительно невелик.

Все эти наблюдения позволили сделать принципиальное заключение о том, что проблема помещения на машинные носители и последующей автоматизированной обработки относится не столько к входному информационному потоку, сколько к текстовой продукции отделов «Правда». Именно этим, в первую очередь, объясняется сдвиг в сторону отдельских подсистем.

В то же время объединяющее начало чувствуется в «Правде», наверное, сильнее, чем в большинстве других газет. В течение ближайших лет ста объединение будет по-прежнему реализовываться, прежде всего, на уровне человеческих отношений.

Однако уже сейчас создание глобальной системы автоматизации набора и верстки газеты, тесно завязывающей службу выпуска с производственными отделами и печатным цехом, является актуальной технической задачей.

Со временем появятся «межотдельские» задачи, связанные с перекрестной обработкой разных баз данных. Необходимо также предвидеть и возможность автоматического пополнения информационных фондов «Правды».

Все эти факторы требуют, наряду с отдельскими подсистемами, сопровождающего развития централизованных систем.

Высказанные соображения поясняют сущность смешанного подхода к организации информационно-вычислительных работ в «Правде» так, как он описан в отчете.

По условиям работы проделанный анализ должен стать основой для выработки технического задания на проектирование ИВС «Правда». Надо, однако, заметить, что какой бы решающий характер ни носило сотрудничество авторов «Системного анализа» с правдистами, данный отчет носит характер одностороннего документа. На этом этапе работы авторы анализа еще не могут взять на себя ответственность навязывать свои выводы в качестве единственно правильных.

Компонент изучения в анализе занимал не меньшую роль, чем выработка рекомендаций, и представление нашего видения «Правды» составляет неотъемлемую часть. Критическая реакция заинтересованной стороны на положения и гипотезы сделанного анализа не менее важна, чем проделанная работа. Есть надежда, что эта попытка отразить «Правду» в зеркале информатики послужит стимулом для такой реакции и создаст предпосылки к выработке достоверного технического задания, удовлетворяющего заказчика и выполнимого для исполнителя.

СИСТЕМА РУБИН ГАЗЕТЫ «ПРАВДА»

Так все и получилось. «Системный анализ производственных процессов по выпуску газеты «Правда» был признан, причем обсуждение предложенных в нем принципов и гипотез активно происходило на разных уровнях, как научных, так и «наверху».

В результате этой почти годовой работы «Проект РУБИН» (Редактирование, Управление, База Информации, Набор) был включен в план важнейших научно-исследовательских работ страны, как совместно осуществляемый Издательством ЦК КПСС «Правда», ВЦ СО АН СССР и МГУ.

В результате обсуждения положений «Системного анализа» совместно с издательством и редакцией газеты «Правда» были выработаны основные принципы построения и порядок разработки и внедрения системы РУБИН.

Они были зафиксированы в документе «Генеральная схема создания и развития информационно-вычислительной системы РУБИН газеты «Правда», утвержденном в феврале 1979 г. главной редакцией газеты в качестве основополагающего документа для разработки.

1. Генеральная схема

Генеральная схема определяет Информационно-Вычислительную Систему РУБИН как основанную на электронной вычислительной технике совокупность аппаратных, программных и организационно-методических средств, органически включаемых в процессы редакционно-издательской подготовки и выпуска газеты и имеющих своим назначением обеспечение комплексного справочно-информационного обслуживания редакции, автоматизированной обработки текстов при подготовке материалов, загрузки и поддержания информационных фондов, планирования и выпуска газеты.

- Система должна обеспечивать высокую надежность выполнения всего комплекса основных и вспомогательных операций по выпуску газеты в режиме тесного взаимодействия с сотрудниками редакции.
- Система РУБИН строится на сочетании центрального вычислительного комплекса с активными, т.е. способными к автономному функционированию периферийными терминалами. При этом реализация и внедрение системы начинается с ввода в строй ее периферийной части.
- Разработка системы РУБИН рассчитана на использование вычислительной техники стран СЭВ, допуская в целях обеспечения передового технического уровня применение периферийной импортной техники в особых случаях.
- Ввод системы в эксплуатацию проводится по этапам, предусматривающим постепенный переход к новым формам работы и с полным

учетом всего сложившегося опыта выпуска газеты и максимальным соответствием структуре редакции «Правды».

2. Основные исходные положения

Главным видом работы при создании очередного номера газеты признавалась подготовка составляющих его текстовых материалов. Характерную основную черту этой работы составляет интенсивная многократная переработка материалов.

Содержательная сторона работы с текстом практически может проводиться пока только человеком и требует, следовательно, включения в систему средств обеспечения диалога для большого числа пользователей.

Для высококачественной подготовки материалов, их проверки, а также планирования номеров система должна предоставлять справочно-информационное обслуживание, опирающееся на большие информационные фонды с малым временем доступа к ним.

Для реализации указанных целей при разработке информационно-вычислительной системы Генеральной схемой предусмотрены следующие действия:

- организация в системе двух отдельных потоков информации — информационно-справочного и текстового, замыкающихся соответственно на справочный фонд и архив текстов; причем по отдельным указаниям возможна передача материалов из справочного фонда в архив текстов и обратно;
- разделение справочного фонда и архива текстов на центральные (для контрольно-справочных отделов и секретариата) и локальные фонды (ведущиеся в тематических отделах для их собственного использования);
- хранение всех текстовых материалов (кроме оригиналов писем) в памяти ЭВМ и их автоматическая обработка по указанию пользователей посредством программных и аппаратных средств;
- использование специальных диалоговых средств формирования; поисковых образов и рефератов (аннотаций) материалов при их включении в справочные фонды, а также помощи в формулировании поисковых предписаний при запросах материалов из справочных фондов и при формировании и оперативном изменении макетов полос в ходе выпуска номера;
- обеспечение автоматического набора материалов, находящихся в архиве текстов, с независимым заданием его полиграфического оформления;
- включение в систему средств, обеспечивающих постоянную адаптацию правил общения с системой.

3. Этапы разработки и внедрения системы РУБИН

Цель выделения этапов создания системы состоит в сокращении периода времени от начала ее проектирования до начала использования, а также в равномерном распределении затрат на приобретение технических средств.

Возможность поэтапного ввода системы в эксплуатацию обеспечивается выбором активного периферийного оборудования и порядком ее запуска от периферии к центру.

Этапы работы предусматривают использование задела в программном обеспечении периферийных мини-ЭВМ и совмещение опытно-промышленной эксплуатации периферийных подсистем с разработкой программного обеспечения центрального вычислительного комплекса.

Предоставляемые в начале эксплуатации системы сравнительно ограниченные функциональные возможности пополняются на каждом последующем этапе.

Каждый этап разработки системы определяет соответствующий уровень предоставляемых функциональных возможностей и средств.

Устанавливаются следующие этапы создания системы, определяемые, главным образом, типом и характеристиками вводимого в эксплуатацию оборудования и временем, необходимым для разработки программного обеспечения.

Первый этап — 1-й уровень средств

Начальный состав функциональных возможностей ИВС предоставляется двум отделам (одному тематическому и одному справочному) и секретариату редакции для опытного практического использования наряду со старыми методами работы.

Вводятся средства эксплуатации локальных фондов, приема информации по телетайпу ТАСС, ведения перспективных и недельных планов, текстового редактирования. При этом для материалов, подготавливаемых средствами системы, обеспечиваются автоматические набор, расчет и набор строк заборки при правке.

Главными целями этого *макетно-демонстрационного* этапа являются демонстрация в реальных условиях преимуществ и удобств работы с системой, а также обеспечение ее коррекции к следующему этапу.

Второй этап — 2-й уровень средств

Создание и пуск в опытно-промышленную эксплуатацию локальных справочных фондов в отделах и секретариате редакции. Практическая проверка и использование средств планирования номеров газеты с возможностями анализа ее содержания по аннотированному оглавлению за ограниченный предшествующий период времени.

Освоение средств первичной обработки и загрузки информационных материалов в справочные фонды контрольно-справочных отделов.

Опытное использование возможностей расчета макета номера в службе выпуска параллельно с существующими методами. Переход машинописного бюро на первичный ввод текстовых материалов (и больших по объему допечаток к ним) на машинные носители информации.

Основными целями этого *определяющего* этапа являются внедрение в практику работы редакции новых средств для подготовки текстовых материалов в номер при помощи ЭВМ, а также разработка программного обеспечения для центрального вычислительного комплекса системы.

Третий этап — 3-й уровень средств

Запуск в эксплуатацию подсистемы ретроспективного анализа содержания газеты в режиме обслуживания контрольно-справочных отделов, секретариата и редколлегии.

Начало формирования центральных справочных фондов. Ввод в строй средств создания и ведения каталогов, индексов локальных и центральных информационно-справочных подсистем.

Начало эксплуатации подсистемы анализа почты газеты на базе локальных фондов отдела писем.

Переход службы выпуска на формирование полос номера через автоматизированный расчет набора и макета с помощью ЭВМ. Создание возможностей перехода к фотонабору.

Главной целью этого *центрального* этапа является включение в работу основных вычислительных мощностей системы.

Четвертый этап — 4-й уровень средств

Объединение центра и периферии в единую информационную сеть. Ввод в эксплуатацию прямой связи оборудования в отделах с центральными справочными фондами и архивом текстов.

Завершение перевода службы выпуска на использование автоматизированных средств и выход на фотонабор.

Запуск информационно-справочной подсистемы внешней информации и средств ее прямого подключения к линиям телесвязи.

Ввод в строй подсистемы обработки писем.

На этом *интегрирующем* этапе завершается начальный период создания ИВС в целом и происходит переход к ее регулярной эксплуатации и развитию.

4. Распределение средств системы по этапам

Общая схема структуры системы РУБИН газеты «Правда» предполагает, что реализация функциональных возможностей системы осуществляется следующими средствами, распределенными по четырем основным группам.

- Средства **Редактирования** обеспечивают эффективную и надежную обработку текстовых материалов при их многократной правке и замыкание всей технической стороны работы с текстами внутри ЭВМ, а также сервисное обслуживание работы с фондами.

- Средства **Управления** обеспечивают долгосрочное и оперативное планирование, подготовку очередных номеров, загрузки подразделений редакции и т.п.
- Средства **Базы Информации** обеспечивают справочно-информационное обслуживание при подготовке публикуемых материалов и проверке содержащихся в них фактов, создание и поддержание центральных и локальных информационных фондов и текстовых архивов, анализ почты газеты и ретроспективный анализ содержания газеты.
- Средства **Набора** обеспечивают непосредственное воспроизведение хранящихся в системе материалов, верстку и выпуск номеров газеты и т.п.

Ниже описывается распределение средств ИВС по группам и уровни средств, образуемые на каждом из этапов создания системы.

База информации составлена совокупностью локальных и центральных справочных фондов и архивов текстовых материалов. Для загрузки и поддержания этих фондов предусматриваются специальные средства, кроме того, для поиска и доступа к хранящимся в основных фондах материалам используются специальные вспомогательные внутрисистемные фонды — индексы и средства управления ими.

Наконец, средства запроса обеспечивают диалоговое формирование пользовательских предписаний для поиска информации в требуемых аспектах.

Вся совокупность информационно-справочных средств организуется, чтобы образовать локальные справочные фонды и центральные справочно-информационные фонды, обслуживающие подсистему ретроспективного анализа содержания газеты, информационно-справочную подсистему внешней информации, подсистему анализа почты газеты, подсистему обработки писем.

Для локальных справочных фондов на 1-м уровне будет обеспечиваться ведение информационных фондов, оперативный объем которых ограничивается техническими возможностями памяти периферийных ЭВМ — примерно 80-100 машинописных страниц. Возможность использования с одного терминала до 100 независимых локальных информационных фондов при поочередном доступе к ним. Независимое неавтоматическое реферирование исходных материалов, без их хранения или с внешним хранением в библиотеке, и использование каталогов с фиксированной рубрикацией. Средства для периодической перестройки каталогов как целого при неавтоматизированном формировании поисковых образов в стандартной форме, составление поисковых предписаний в регулярной форме или на ограниченном формальном языке. Защиту методом разделения фондов и паролями пользователей.

На 2-м уровне будут сняты некоторые ограничения и добавлены возможности за счет: связи с текстовыми редакторами и локальными архивами тек-

стов, введения средств оперативного изменения состава рубрик в каталогах и индексах в некоторых небольших пределах и средств образования логических связей другими локальными справочными фондами.

На 3-м уровне будут сняты ограничения по динамическому изменению каталогов и добавлены средства для:

- диалогового составления поисковых образов и формирования поисковых предписаний при обращении к центральным информационным фондам;
- составления рефератов методом диалогового сокращения введенных в память текстов при небольшом их объеме;
- защиты по совокупности допусков материала и пользователя;
- возможности описания формата общения с использованием макросредств и других средств верстки и сокращений.

На 4-м уровне за счет подключения к центральным фондам будут сняты ограничения на общение и появится возможность обмена информацией между локальными и центральными информационно-справочными фондами, а также возможность оперативной связи с терминала со службой выпуска через ЭВМ.

Для центральных справочных фондов на 1-м уровне будет обеспечена возможность приема непосредственно в периферийную ЭВМ сообщений, приходящих по телетайпным каналам ТАСС, с их последующим просмотром референтом и оперативным направлением для дальнейшего использования (в информационно-справочные фонды или картотеку, в номер, на сброс и т.п.).

На 2-м уровне станут доступными основные средства загрузки и поддержания фондов, включающие диалоговое составление поисковых образов и оперативную подготовку текстов рефератов и, тем самым, сможет начаться их постепенное накопление во внешней памяти ЭВМ.

На 3-м уровне начнется эксплуатация системы ретроспективного анализа содержания газеты, функционирующей на центральной ЭВМ. Обеспечивающие ее работу средства будут состоять из диалогового формирования поисковых образов и поисковых предписаний, динамического неавтоматического управления каталогом — введения и удаления его рубрик, составления рефератов методом диалогового сокращения для текстов, предварительно введенных в память ЭВМ, а также независимого составления рефератов для внешних материалов, возможности поддержания архива опубликованных в газете материалов.

Объем информационных фондов системы ретроспективного анализа содержания газеты будет ограничиваться физическими темпами накопления материалов в них. Доступ к этим фондам будет на 3-м этапе ограничен несколькими терминалами, подключенными непосредственно к центральной ЭВМ системы.

На 4-м уровне предоставляется связь с периферийными ЭВМ и, следовательно, с локальными справочными фондами и службой выпуска. Будут введены: подсистемы обработки писем, связь с информационно-справочной системой внешней информации, автоматическое корректирование каталогов, управляемое частотой обращений, возможности адаптации языков общения пользователей с системой, единый для всей системы порядок защиты доступа к фондам, возможности автоматизированного диалогового составления рефератов.

Предполагается использование подсистемы *ретроспективного анализа* содержания газеты в качестве действующей модели подсистемы внешней информации, поскольку объем последней существенно больше. Вследствие этого средства, перечисленные выше для подсистемы ретроспективного анализа содержания газеты, будут с определенным временным сдвигом доступны и для информационно-справочной системы внешней информации. При этом имеется в виду, что и для нее возможности составления рефератов диалоговым сокращением и автореферированием будут играть значительно большую роль.

На 1-м и 2-м уровнях средством обеспечения основных возможностей, предоставляемых для управления, являются локальные справочные фонды секретариата и редколлегии. Их содержание будет отвечать выполняемым с их помощью функциям: обеспечения проверки исполнения системы редакционных планов, ведения динамической модели газеты, планирования очередных номеров газеты, контроля реагирования на опубликованные материалы, учета текущей загрузки отделов, дежурств и т.п.

На 3-м уровне средства управления будут пополнены функциями, осуществляемыми на центральной ЭВМ: система ретроспективного анализа содержания газеты, средства анализа почты газеты, специализированные средства управления и планирования текущей работы, централизованная система контроля за реакцией на публикации газеты.

На 4-м уровне средства управления будут интегрированы с остальными возможностями ИВС, со снятием ограничений на доступ и объемы используемых информационных фондов.

Базисные средства текстового редактирования 1-го уровня будут обслуживать общий объем оперативно доступных текстов до двух учетно-издательских листов, включать средства обеспечения строково-абзацно-страничной структуры представления текстов, комплекта команд для строкового и внутристрокового редактирования, средств обеспечения адресации к тексту по структуре его носителя через распечатку или отображение на дисплее, средств сквозного поиска вхождений подстроки в строки текста, средств копирования и перестановки фрагментов текста.

На 2-м уровне к средствам редактирования будут добавлены:

- возможность адресации по структуре текста — к словам, фразам, рубрикам разного ранга; доступ к текстам материалов из локального справочного фонда — только одного одновременно;
- возможность включения в текст «внетекстовых», т.е. невоспроизводимых при наборе, вставок для создания вариантов внутриредакционных комментариев и справок, а также для полиграфической разметки текстов;
- средства выделения на экране фрагментов текста, подлежащих проверке (так называемое «зажигание», блокирующее прохождение текста с «зажженными» фрагментами через программы набора вплоть до «гашения» этих фрагментов после их проверки ответственными за это лицами);
- средства сцепления нескольких текстов в один, с суммарным объемом оперативно доступных текстов, не превышающим четырех учетно-издательских листов.

Дополнительные редактирующие возможности 3-го уровня будут обеспечивать адресацию по образцам, более богатый состав команд текстового редактирования, а также автоматическую проверку написания отдельных вхождений образцов в тексте для небольшого словаря образцов.

На 4-м уровне все перечисленные функциональные возможности интегрируются вместе со средствами верстки и выпуска и, кроме того, пополняются доступом ко всем справочным фондам и архивам текстов, регулируемым средствами защиты, составлением указателей на вхождение в текст заданных образцов и автоматическим расчетом таблиц и других видов блочных текстов.

При этом будут сняты ограничения на объемы оперативно доступных текстов и фондов. Кроме того, для обеспечения малого времени ответа системы функции редактирования будут перераспределены (невидимо для пользователей) между центральной и периферийными ЭВМ.

Средства набора и выпуска на 1-м уровне обеспечивают возможности для автоматизации процессов расчета и организации выключки материалов в столбцы и набора подготовленных текстовых материалов через управляемые перфолентой автоматические линотипы *НА-140* или аналогичные. Распечатывание оригинал-макетов набранных столбцов ведется при помощи ЭВМ. Использование телетайпной информации осуществляется ее вводом с линии в периферийную ЭВМ и выдачи оригинала-макета автоматически сформированных столбцов для оперативной правки на экране дисплея непосредственно в номер, определения размеров материалов в строках набора, а также составления и выдачи перфоленты заборки строк при правке.

На 2-м уровне будут добавлены средства для:

- расчета макета с выдачей текущего состояния на графопостроитель;
- автоматического расчета и выдачи бланков заказов для набора на крупнокегельных машинах заголовков газеты;

- учета реализации внесенных верстальщиком исправлений по сигналам с талера;
- автоматического выбора авторских вариантов текста по заданному объему сокращения величины материала при выпуске;
- проверки верстаемых материалов на отсутствие неснятых вопросов («зажженных» мест текста);
- доступа к средствам текстового редактирования для обработки последних оперативных материалов, поступающих с телетайпа.

На 3-м уровне добавятся возможности:

- управления макетом на графическом дисплее с одновременным показом составляющих текстов на текстовом дисплее по вызову и автоматической корректировкой макета при правке текстового материала;
- дублирования текущего состояния номера (макета и текстов) для руководящих работников редакции в 3-4 места;
- обеспечения оперативной связи руководства редакции и группы выпуска посредством терминалов системы РУБИН.

На 4-м уровне станет возможной прямая связь с центральными текстовыми архивами, контрольно-справочными отделами и центральными справочными фондами.

Таково было мое видение развития системы весной 1979 г., выраженное в «Генеральной схеме», утвержденной Главной редакцией «Правды» в феврале 1979 г., — оптимистичное по функциональному наполнению и благожелательности сотрудников, и консервативно-пессимистичное в отношении прогресса технических средств.

Разработки по проекту РУБИН начали осуществляться сразу в нескольких направлениях как по периферии, так и по центральному вычислительному комплексу, второе направление работ возглавил в Москве проф. Э.З. Любимский. Его лаборатория сосредоточилась на адаптации и доработках информационных систем для ЕС ЭВМ 1060х2, которую установили на комбинате.

Успешно было защищено техническое задание на систему, и начался этап технического проекта, регулярно собирался на заседания Научно-технический совет Проекта, началась подготовка заполнения информационных баз данных в ЦВК.

Однако непосредственный выход в редакцию по-прежнему определялся тем, какая аппаратура будет стоять на рабочих местах сотрудников. Ничего подходящего не производилось не только у нас, но и за рубежом. Кроме того, напомним, что для РУБИНа допускалось использовать только отечественную вычислительную технику, в крайнем случае — из стран СЭВ.

Основные трудности заключались в том, что предоставляемое шрифтовое обеспечение было слабым, а для зарубежных средств — только с латинским шрифтом.

После попыток разместить заказы в отечественной промышленности и долгих поисков партнера по разработке, в проекте РУБИН появился в середине 1980 г. новый адрес — г. Блоне под Варшавой. Там располагается завод точного машиностроения «МЕРА-Блоне», делавший в рамках СЭВ печатающие устройства для всех ЕС ЭВМ и матричные принтеры для минимашин. Именно он стал нашим партнером по созданию рабочих мест для периферии РУБИНа.

Однако это уже другая история — история проекта МРАМОР.

Л. В. Городняя

ПОЧТИ 30 ЛЕТ СПУСТЯ

ВВЕДЕНИЕ

В этом году при разборе своего научного архива мне удалось найти многие рукописи давних лет. Оказалось небезынтересно взглянуть на них с позиций сегодняшнего дня и сопоставить тогдашнюю оценку проблем и перспектив технологии программирования с современными воззрениями.

Известно, что при повторном издании своей знаменитой книги Фр. Брукс отметил поразительную устойчивость основной проблематики. Что же в этом видели и до сих пор видим мы?

Особенно интересен в этом плане обзор подходов к реализации языков программирования по материалам конференции 1975 года, подготовленный для журнала «Программирование» в начале 1976 года, отредактированный А.П. Ершовым и прорецензированный А.А. Берсом.

Проблематика системного программирования теперь редко признается особо важной, хотя в профессиональных кругах она бесспорно остается в числе ключевых и критичных. Трудоемкость решения многих проблем не вызывает энтузиазма и провоцирует на поиск новых панацей. При чтении старых обзоров возникают слегка скептические комментарии, но в основном Брукс, по-видимому, прав: основные трудности программирования носят столь глубоко содержательный характер, что пути преодоления этих трудностей пока ускользнули от нас.

Но вернемся мысленно почти на тридцать лет назад. Конференция «Методы реализации алгоритмических языков» вполне заслуживала название «Сумма технологий». Каждый подход был представлен в разных вариантах, можно было сравнить и оценить методы, языки и системы с разных точек зрения. Подготовленный в феврале 1976 года обзор отражает лишь часть круга идей, «носившихся в воздухе» тех времен.

С 10 по 13 сентября 1975 г. в городе Новосибирске состоялся симпозиум по методам реализации алгоритмических языков, организованный Вычислительным центром СО АН СССР. Работой оргкомитета и симпозиума руководил член-корреспондент АН СССР А.П. Ершов. 59 участников симпозиума, представители 23 организаций из 10 городов СССР, и 11 гостей из 7 стран провели серию заседаний, завершившуюся оживленной дискуссией

о системах построения трансляторов (СПТ). (Труды симпозиума, публикуемые в издании ВЦ СО АН СССР, были доступны к пересылке наложенным платежом по гарантийным письмам.)

Многоплановость большинства докладов, а также стремление к компактности определили стиль настоящего обзора: тематически систематизированное изложение основных положений со ссылками на доклады, вошедшие в сборник трудов (их список помещен в конце обзора). Названия тех докладов, которые не вошли в этот сборник, приводятся непосредственно в тексте. Выделены следующие тематические разделы.

1. Методика. Методология.
2. Общие схемы реализации языков.
3. Выбор языка реализации.
4. Грамматический анализ.
5. Практичность систем программирования.
   5.1. Независимость от машины.
   5.2. Оптимизация в программировании.
6. Структура данных.
7. Расширяемость. Макротехника.
8. Диагностика ошибок.
9. Генерация тестов.

1. МЕТОДИКА. МЕТОДОЛОГИЯ

Полезность программных (идейных) документов в больших проектах и четкого разграничения научно-исследовательской и опытно-конструкторской работ [4], целесообразность выделения так называемого Внутреннего языка, обеспечивающего стабильный уровень языково-независимого анализа программ и их оптимизации [7] и унифицирующего класс алгоритмических языков, показывает анализ развития проекта БЕТА [4].

Сформулированы некоторые требования к методике построения больших программ: «Методика должна быть императивной, целенаправленной, всепоглощающей и развивающейся», и предложена «операционная сеть» как модель программной поддержки такой методики [11].

Для многих проектов характерно стремление описывать трансляторы, модифицируя описания синтаксиса исходного языка [1, 3, 10–13, 17, 30].

Отмечается, что лишь небольшая часть проекта транслятора описывается в синтаксисе языка и даже хорошая структура получения транслятора по

описанию языка только перекладывает проблемы с разработчика транслятора на создателя языка [32].

2. ОБЩИЕ СХЕМЫ РЕАЛИЗАЦИИ ЯЗЫКОВ

1. Программирование на специальном языке транслятора с исходного языка на язык команд абстрактной машины предлагается в многоязыковой транслирующей системе T-SEMOL [15]. Описаны типичные схемы трансляции, выполняемые в системе T-SEMOL. Язык команд абстрактной машины похож на систему команд ЭВМ типа М-20.

2. Использование одного специального языка в качестве как языка программирования транслятора, так и промежуточного языка в двухступенчатой схеме трансляции [С.С. Камынин. Э.З. Любимский. Алгоритмический машино-ориентированный язык как средство разработки машинно-независимого обеспечения].

3. Задание множества процедур, выполняющих частные действия, необходимых для трансляции с исходного языка, и описание вывода транслятора в исчислении трансляции, по которым система МАСОН строит на промежуточном языке высокого уровня транслятор с исходного языка [14].

4. Описание задачи множеством отношений и множеством используемых в отношениях процедур в системе программирования ПРИЗ, по которому строится вычислительная модель. Метод применим к построению проблемно-ориентированных расширений языка [21].

5. Представление транслятора в виде последовательности преобразований текста исходного языка в текст эталонного языка на автоматно-ориентированном R-метаязыке [10].

6. Синтаксически управляемая трансляция на основе программы, транслирующей текст по метаописанию языка, и правил вывода объектного кода [12].

7. Получение транслятора с исходного языка на базовый по описанию синтаксиса исходного языка в виде формул над словами и семантики языка в виде эквивалентных текстов на базовом языке в системе DEPOT [1].

8. Интегрированное описание синтаксиса и семантики языка на расширении Паскаля, которое система построения трансляторов [3] преобразует в написанный на Паскале транслятор с исходного языка на Паскаль. Паскаль обеспечивает переносимость, практичность и изучаемость полученного транслятора [3].

9. Тщательное отделение языково-зависимых и машинно-зависимых черт транслятора в системе БЕТА [6]. Новый язык погружается в систему описанием языково-зависимых черт транслятора [5]. Выход на новую машину описывается конкретизацией машинно-зависимых черт [6]. Принципиальная возможность языково-независимого анализа и преобразований программ [7, 9] определяет практичность системы [7, 8].

10. Динамическое взаимодействие языково-зависимой и машинно-зависимой фаз трансляции [С.С.Камынин, Э.З.Любимский].

11. Получение транслятора с исходного языка по описанию синтаксиса языка модифицированными БНФ, семантики языка в терминах атрибутов Кнута и прагматики языка на специальном языке PRA в системе COPS [30]. Описание машинно-зависимых черт на языке описания прагматики обеспечивает выход системы на разные машины.

12. Описание на специальном языке CDL, похожем по своим функциям на систему построения трансляторов и интерпретирующем данные как макрогенератор синтаксиса и семантики языка, действий, обеспечивающих анализ текста и работу с таблицами. Система на базе языка CDL обеспечивает оптимизацию всего процесса проектирования и внедрения нового языка [32].

13. Представление языка в виде макрорасширения базового языка позволяет реализовывать новый язык с помощью макрогенератора [28].

3. ВЫБОР ЯЗЫКА РЕАЛИЗАЦИИ

Отмечено две тенденции в выборе языков реализации [1]. Универсальные языки типа Алгол-68 используются для описания фазы синтеза объектного кода [31] и для обеспечения независимости описания от машины [23]. Алгол-68 — любимый пробный камень многих разработок [2, 5, 16, 20, 23, 31, 32]. Но для задачи построения транслятора универсальный алгоритмический язык слишком богат и имеет бесполезные конструкции. Транслятор с такого языка велик, эффективность объектного кода низка [31]Основные действия при трансляции: анализ, обслуживание таблиц и ввод-вывод, их легче описывать на удобном специальном языке, легком для программирования и изучений [32].

CDL — язык реализации трансляторов, по целям и функциям похож на СПТ, основан на типе грамматик, удобном для описания языков [32].

МИДЛ — гибридный язык промежуточного уровня, создан с целью описания оптимизатора для языка весьма высокого уровня Сетл [25].

Язык L2 предназначен для решения задач, возникающих при построении больших и содержательных программ [27].

Ярмо — машинно-ориентированный язык высокого уровня, спроектирован для реализации матобеспечения [26].

Semol — язык представления трансляторов [15].

Система МАСОН основывается на трех специализированных языках: метаязык для описания процессов трансляции, промежуточный язык, обеспечивающий независимость объектного кода от машины, и язык заданий, описывающий вывод формулы в исчислении трансляций [14].

4. ГРАММАТИЧЕСКИЙ АНАЛИЗ

Использование формализованного синтаксиса для управления процессом трансляции позволяет свести проблему синтаксического анализа к проблеме построения синтаксического дерева, а проблему трансляции — к проблеме обхода дерева [13].

Изучаются соотношения между классами грамматик [29, В.Н.Редько. Проблемы построения многоязыковых процессоров] и преобразований КС-грамматики для построения автомата, анализирующего порождаемый КС-грамматикой язык [17].

Введение новой синтаксической концепции «позиция» [4, 5] позволяет регуляризировать КС-грамматику, перейти от синтаксической структуры к грамматической [5], содержащей описание понятий с помощью атрибутов. Синтаксическая структура не зависит от набора понятий. Отмечается удобство метода атрибутов Кнута для описаний семантических действий [2].

Атрибуты синтезируют информацию, по которой проверяются некоторые условия согласованности [2, 5].

Метод описания семантики с помощью атрибутов используется в ряде систем [3, 5, 16, 30], хотя он не прост [16].

5. ПРАКТИЧНОСТЬ СИСТЕМ ПРОГРАММИРОВАНИЯ.

Большинство проектов ориентируется на обеспечение производства больших программ.

Эта ориентация определяет заботу о независимости проектов от машин, операционных систем и алгоритмических языков [3]. Раздельная трансляция [32, 9, 16], трансляция на промежуточный язык [14, 16, 3], оптимизации программ [7, 8, 9, 16, 32] отражают стремление к практичности систем.

5.1. Независимость от машин.

Необходимость быстрого распространения программ на новые машины побуждает разработчиков алгоритмических языков и трансляторов обеспечивать независимость от машин.

Прежде всего это отражено в стремлении к отсутствию машинно-зависимых черт в структуре данных [26, 32], в поиске машинно-независимых схем обработки [15, 16] и описания [23] программ.

Выделение прагматики в определении исходного языка делают возможной генерацию на любом объектном коде. Прагматика — способ преобразования атрибутного дерева разборов для получения результирующего кода [30].

Вариантом такого подхода является выделение промежуточного языка как дополнительного этапа трансляции [16, 14]. Промежуточный язык позволяет решить и проблему раздельной трансляции частей программы [16].

Другое решение проблемы предложено в системе БЕТА. Внутренний язык этой системы — язык высокого уровня, не имеющий внешнего представления. Этот язык пригоден для решения машинно-независимых проблем, возникающих при трансляции [4, 6, 7, 8, 9]. Но генерация рабочей программы происходит машинно-зависимо [6].

5.2. Оптимизация в программировании.

Подробно исследованы проблемы анализа программ и оптимизирующих преобразований в проекте БЕТА [9, 7, 8].

Выбор внутреннего языка как уровня анализа и преобразований программы придает этому исследованию независимость и от алгоритмических языков, и от вычислительных машин. Оптимизация объектного кода — одна из целей создания внутреннего языка [1, 6].

Выделение анализа программы в отдельную фазу, упорядочение множества типов реализуемых объектов и множества типов информации, учитываемой при решении вопроса о применимости набора преобразований, исключают дублирование поиска нужных сведений в представлении программы [9].

Выбор последовательности применения глобальных оптимизирующих преобразований, учитывающих далекие информационные и логические связи и операторы, возникающие при трансляции и при сборке программы их модулей, может быть произведен и системой, и пользователем. Преобразования, не всегда сохраняющие эквивалентность программ, например, чистка зон [7], производятся лишь по указанию пользователя.

Более широкий подход к этой проблеме предлагается в системе CDL: оптимизация всего процесса проектирования и реализации программы [32]. Инструменты такой оптимизации: редактор, обеспечивающий взаимодействие частей транслятора, библиотека с возможностью претрансляции и раздельной трансляции, механизм отладки, возможность получать текущий файл транслятора и оптимизатор, повышающий эффективность результирующего транслятора. Признано, что первая цель оптимизации — «умиротворить» программиста, который или откажется от недостаточно удобной и эффективной системы, или будет заботиться об оптимальности исходной программы в ущерб ее выполнимости, если ему не докажут, что автоматическая оптимизация программы ничуть не хуже [32].

Программист должен заботиться лишь о правильности, удобочитаемости, переносимости и адаптируемости программы. Эффективность должна гарантировать система [32].

Разработчики систем, основанных на языках высокого уровня, уделяют особое внимание автоматической или автоматизированной оптимизации объектного кода.

Для языка Сетл создан большой оптимизатор, значение которого подчеркивается созданием специального языка Мидл для реализации оптимизатора [25].

В реализации модифицированного языка Сетл большие надежды возлагаются на диалог между пользователем, который многое знает о своей программе, но не умеет ее оптимизировать, и системным оптимизатором, который умеет применять факты о программе, но не знает их [22].

Предлагается алгоритм генерации — автомат с двумя стеками, минимизирующий количество копий объектов во время выполнения и управление памятью и производящий машинно-зависимую оптимизацию [16].

6. СТРУКТУРА ДАННЫХ

Создатели языка L2, языка для обработки структур данных, приняли за основу алгебраический подход, дающий формализм для построения строгой базы языка. Основной тип данных — «останов» и работа с указателями — основа методики доступа к элементам составного. Сложные преобразования структур данных представляются в виде операторов применения соотношений [27].

В языке Ярмо представлены средства описания структур памяти, состоящей из последовательности ячеек, с возможность организации эффек-

тивного доступа к элементам структуры подстановкой шаблонов операций чтения и записи [26].

Система ПОПЛАН использует стек, организованный в виде списка небольших порций, в надежде на более эффективное распределение памяти [24].

Системы на базе языка Сетл представляют отображения с помощью расширяемых расстановочных полей [22, 25].

Язык Мидл содержит аппарат описания доступа к чужим переменным [25].

При реализации Алгола-68 к типичным видам памяти «стек» и «куча» добавлен вид памяти «пузырь», характеризуемый сравнительно простым способом освобождения памяти. Описана реализация семафоров на однопроцессорной машине [19].

Для передачи информации от конструкции к внутренним и объемлющим конструкциям предлагаются специальные логические переменные, называемые предсказателями. С их помощью оценивается пространственно-временная стоимость вариантов рабочей программы с целью выбора самого дешевого варианта при повторном просмотре текста [31].

В R-метаязыке для формального задания семантики языков программирования используются абстрактные виды памяти: счетчиковая, вагонная, регистровая, табличная. Этот набор считается пополняемым, хотя язык описания новых видов памяти не предлагается. Определен лишь внешний вид операторов доступа к памяти [10]. Автоматно-ориентированный R-метаязык предназначен для задания отображений исходного, определяемого класса объектов в другие, эталонные, известные [10].

7. РАСШИРЯЕМОСТЬ. МАКРОТЕХНИКА

7.1. Расширяемость

Расширяемость — желанная черта современных систем программирования. Для ее реализации в систему включают макроязык [14] или язык, обладающий возможностями макроаппарата [6, 22, 25, 27].

В системе на базе языка L2 расширяемость используется для ввода новых операций и типов данных. Выполняют расширение переводящие подпрограммы, макрогенератор, а на уровне входного языка — операторы загрузки синтаксиса и семантики [27].

7.2. Макротехника

«Часть математического обеспечения, разработанная для того, чтобы позволить пользователю добавлять новые средства его собственной конструкции к существующей части математического обеспечения» П. Браун называет макрогенератором [28].

Макрогенератор используется в системе на базе языка L2 [27] как инструмент для расширения системы, в системе УТОПИСТ [21] — как генератор вычислительной модели, в реализации транслятора с Алгола-68 [31] — как генератор рабочей программы.

В системе БЕТА макросы используются как запросы к обстановке машинно-зависимых рабочих программ [6].

Разработчики языка Ярмо предлагают макроподстановку как способ описания доступа к данным [26].

Макрорасширяемы языки Сетл [22, 25], Балм [22], Мидл и Литтл [25]. Макроаппарат обеспечивает локализацию машино-зависимых черт программ на этих языках, что существенно облегчает перенос алгоритмов на новые машины.

Система, основанная на языке реализации трансляторов CDL [32], интерпретирует описание исходного языка с точки зрения макрогенерации. Есть версия, транслирующая на множестве макросов, подставляемых в зависимости от конкретной машины, и тем самым улучшающая в два раза пространственно-временные характеристики объектного кода.

Язык Метамакр [28] — попытка уточнить понятие макрогенерации и применить полученное уточнение для макрорасширения языков программирования.

8. ДИАГНОСТИКА ОШИБОК

В системе, обрабатывающей описание грамматики, имеет смысл проверка корректности описания [2].

Семантический контроль целесообразен, лишь когда программист убежден, что его программа почти правильна [32]. Имеет смысл проверять отсутствие побочного эффекта и ошибки типа "неопределенное значение" [32].

Применяется метод маркеров в трансляторах для продолжения анализа после обнаружения синтаксически — правильной программы. [С.П. Крицкий. Методика создания транслятора, устойчивого к синтаксическим ошибкам]

9. ГЕНЕРАЦИЯ ТЕСТОВ

Система на основе R-метаязыка [10] содержит дедуктор, используемый в режиме порождения контрольных тестов.

Возможность генерации минимального набора правильных программ, использующих все правила грамматики языка, отмечается в описании СПТ [3].

* * *

Труды всесоюзного симпозиума по методам реализации новых алгоритмических языков: Сб. науч. тр. / Под ред. В.Л. Каткова. — Новосибирск: ВЦ СО АН СССР, 1975. — Т. 1–2.

Содержание

13. Я. Крал. Почти нисходящий анализ обобщенных грамматик (на английском языке). Прага. ЧССР.
14. В. Л. Темов. Металингвистическая система общего назначения (МАСОН). Ленинград.
15. М. Г. Гонца. Подход к автоматизации построения многоязыковых транслирующих систем. Кишенев.
16. Р. Бранкар, Дж. Р. Кардинал, Дж. Леви, Дж. Р. Делескай, М. Ван Бегин. Простой транслирующий автомат, позволяющий генерировать оптимальный код. Брюссель. Бельгия.
17. Р. Штробель. Автоматические преобразования КС-грамматик. Берлин. ГДР.
18. Г. Е. Цейтлин, Е. Л. Ющенко. Некоторые вопросы теории параметрических моделей языков и параллельный синтаксический анализ. Киев.
19. Г. С. Цейтин. Реализация параллельного исполнения и гибких имен в АЛГОЛе-68. Ленинград.
20. И. О. Кернер. Один подъязык АЛГОЛа-68 и его реализация. Росток. ГДР.
21. Э. Х. Тыугу. Система программирования с автоматическим синтезом алгоритмов. Таллин.
22. Д. Я. Левин. Экспериментальная реализация языка СЕТЛ. Новосибирск.
23. М. Р. Левинсон. Метареализация АЛГОЛа-68. Москва.
24. Ю. М. Баяковский, Н. И. Вьюнова, В. А. Галантенко, А. Б. Ходуев. Некоторые особенности реализации языка РОР-2. Москва.
25. Е. Дик, М. Шимасаки, Дж. Шварц. МИДЛ: гибридный язык промежуточного уровня. Нью-Йорк. США.
26. Б. Г. Чеблаков. Представление структур данных в машинно-ориентированном языке высокого уровня. Новосибирск.
27. С. С. Гороховский, Ю. В. Капитонова, А. А. Летичевский. L2 — язык для обработки структур данных и его реализация. Киев.
28. В. Ш. Кауфман. О макрорасширении языков программирования. Москва.
29. А. Н. Маслов. Использование расширения индексных грамматик для синтаксического анализа. Москва.
30. Я. Боровец. Прагматика в системе построения компиляторов. Варшава. Польша.
31. А. Н. Терехов, Г. С. Цейтин. Язык синтез объектной программы с учетом последующего контекста. Ленинград.
32. К. Костер. CDL — язык реализации трансляторов (на английском языке). Западный Берлин.

И. Н. Скопин

МОДЕЛИ ЖИЗНЕННОГО ЦИКЛА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Рассматривается моделирование жизненного цикла программного обеспечения как основа технологичной разработки программ. Представлены разные подходы к моделированию жизненного цикла, отражающие различные представления о назначении такого моделирования. Описываются особенности объектно-ориентированного моделирования жизненного цикла, в том числе и учет непрерывно поступающих требований к разрабатываемому проекту.

ВВЕДЕНИЕ

Понятие жизненного цикла программного обеспечения появилось, когда программистское сообщество осознало необходимость перехода от кустарных ремесленнических методов разработки программ к технологичному промышленному их производству. Как обычно происходит в подобных ситуациях, программисты попытались перенести опыт других индустриальных производств в свою сферу. В частности, было заимствовано понятие жизненного цикла.

Аналогия жизненного цикла программного обеспечения с техническими системами имеет более глубокие корни, чем это может показаться на первый взгляд. Программы не подвержены физическому износу, но в ходе их эксплуатации обнаруживаются ошибки (неисправности), требующие исправления. Ошибки возникают также от изменения условий использования программы. Последнее же является принципиальным свойством программного обеспечения, иначе оно теряет свой смысл. Поэтому правомерно говорить *о старении программ*, хотя не о физическом старении, а о моральном.

Необходимость внесения изменений в действующие программы как из-за обнаруживаемых ошибок, так и по причине развития требований приводит по сути дела к тому, что разработка программного обеспечения продолжается после передачи его пользователю и в течение всего времени жизни программ. Деятельность, связанная с решением довольно многочисленных задач такой продолжающейся разработки, получила название *сопровождения программного обеспечения* (рис. 1).

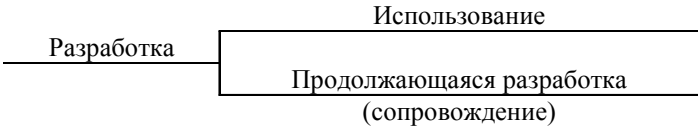


Рис. 1. Разработка, использование и сопровождение программного обеспечения

Исторически развитие концепций жизненного цикла связано с поиском для него адекватных моделей. Как и всякая другая, модель жизненного цикла является абстракцией реального процесса, в которой опущены детали, несущественные с точки зрения назначения модели. Различие назначений применения моделей определяет их разнообразие.

Основные причины, из-за которых нужно изучать вопросы моделирования жизненного цикла программного обеспечения, можно сформулировать следующим образом.

Во-первых, это знание даже для непрофессионального программиста помогает понять, на что можно рассчитывать при заказе или приобретении программного обеспечения и что нереально требовать от него. В частности, неудобные моменты работы с программой, ее ошибки и недоработки обычно устраняются в ходе продолжающейся разработки, и есть основания ожидать, что последующие версии будут лучше. Однако кардинальные изменения концепций программы — задача другого проекта, который совсем необязательно будет во всех отношениях лучше данной системы.

Во-вторых, модели жизненного цикла — основа знания технологий программирования и инструментария, поддерживающего их. Программист всегда применяет в своей работе инструменты, но квалифицированный программист знает, где, когда и как их применять. Именно в этом помогают понятия моделирования жизненного цикла: любая технология базируется на определенных представлениях о жизненном цикле, выстраивает свои методы и инструменты вокруг фаз и этапов жизненного цикла.

В-третьих, общие знания того, как развивается программный проект, дают наиболее надежные ориентиры для его планирования, позволяют экономнее расходовать ресурсы, добиваться более высокого качества управления. Все это относится к сфере профессиональных обязанностей руководителя программного проекта.

В настоящей работе модели жизненного цикла представлены в таком виде, позволяющем рассматривать их, абстрагируясь от специфики разра-

ботки конкретных программных систем. Описываются традиционные модели и их развитие, приспособленное к потребностям объектно-ориентированного проектирования.

1. МОДЕЛИ ТРАДИЦИОННОГО ПРЕДСТАВЛЕНИЯ О ЖИЗНЕННОМ ЦИКЛЕ

1.1. Общепринятая модель

Вероятно, самым распространенным мотивом обращения к понятию жизненного цикла является потребность в систематизации работ в соответствии с технологическим процессом. Этому назначению хорошо соответствует так называемая *общепринятая модель* жизненного цикла программного обеспечения, согласно которой программные системы проходят в своем развитии две *фазы*:

- разработка,
- сопровождение.

Фазы разбиваются на ряд *этапов* (рис. 2).

Разработка начинается с *идентификации потребности* в новом приложении, а заканчивается передачей продукта разработки в эксплуатацию.

Первым этапом фазы разработки является *постановка задачи и определение требований*. Определение требований включает описание общего контекста задачи, ожидаемых функций системы и ее ограничений. На этом этапе заказчик совместно с разработчиками принимают решение о создании системы. Особенно существен этот этап для нетрадиционных приложений.

В случае положительного решения начинается этап *спецификации системы в соответствии с требованиями*. Разработчики программного обеспечения пытаются осмыслить выдвигаемые заказчиком требования и зафиксировать их в виде спецификаций системы. Важно подчеркнуть, что назначение этих спецификаций — описывать внешнее поведение разрабатываемой системы, а не ее внутреннюю организацию, т.е. отвечать на вопрос, *что* она должна делать, а не *как* это будет реализовано. Здесь говорится о назначении, а не о форме спецификаций, поскольку на практике при отсутствии подходящего языка спецификаций, к сожалению, нередко приходится прибегать к описанию «*что*» посредством «*как*»[1]. Прежде чем

[1] Проблемы языка спецификаций не в том, что нельзя (или трудно) строго и четко описать, что требуется в проекте. В большей степени они связаны с необходимостью добиваться и поддерживать соответствие описания «*что*» нечетким, неточным

приступать к созданию проекта по спецификациям, они должны быть тщательно проверены на соответствие исходным целям, полноту, совместимость (непротиворечивость) и однозначность.

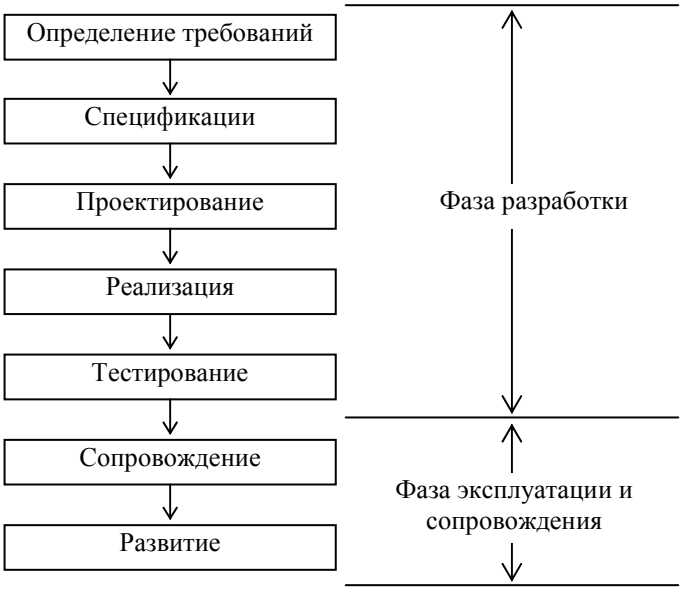


Рис. 2. Общепринятая модель жизненного цикла программного обеспечения

Разработка проектных решений, отвечающих на вопрос, *как* должна быть реализована система, чтобы она могла удовлетворять специфицированным требованиям, выполняется на этапе *проектирования*. Поскольку сложность системы в целом может быть очень большой, главной задачей

и часто противоречивым требованиям со стороны внешних по отношению к проекту людей. Нет оснований полагать, что эти люди будут знакомы с «самым хорошим языком спецификаций», что они будут заботиться о корректности своих требований. Задача этапа спецификаций в том и состоит, чтобы описание программы выстроить в виде логически выверенной системы, понятной как для заказчика данной разработки, будущих пользователей, так и для исполнителей проекта.

этого этапа является последовательная декомпозиция системы до уровня *очевидно реализуемых* модулей или процедур.

На следующем этапе *реализации*, или *кодирования* каждый из этих модулей программируется на наиболее подходящем для данного приложения языке. С точки зрения автоматизации этот этап традиционно является наиболее развитым.

В рассматриваемой модели фаза разработки заканчивается этапом *тестирования* (автономного и комплексного) и *передачей* системы в *эксплуатацию*.

Фаза эксплуатации и сопровождения включает в себя всю деятельность по обеспечению нормального функционирования программных систем, в том числе фиксирование вскрытых во время исполнения программ ошибок, поиск их причин и исправление, повышение эксплуатационных характеристик системы, адаптацию системы к окружающей среде, а также, при необходимости, и более существенные работы по совершенствованию системы. Все это дает право говорить об *эволюции системы*. В связи с этим, фаза эксплуатации и сопровождения разбивается на два этапа: собственно *сопровождение* и *развитие*. В ряде случаев на данную фазу приходится большая часть средств, расходуемых в процессе жизненного цикла программного обеспечения.

Понятно, что внимание программистов к тем или иным этапам разработки зависит от конкретного проекта. Часто разработчику нет необходимости проходить через все этапы, например, если создается небольшая хорошо понятная программа с ясно поставленной целью. Проблемы сопровождения, плохо понимаемые разработчиками небольших программ для личного пользования, являются в то же время очень важными для больших систем.

Такова краткая характеристика общепринятой модели. В литературе встречается много вариантов, развивающих ее в сторону детализации и добавления промежуточных фаз, этапов, стадий и отдельных работ (например, по документированию и технологической подготовке проектов) в зависимости от особенностей программных проектов или предпочтений разработчиков.

1.2. Классическая итерационная модель

Общепринятая модель жизненного цикла является идеальной, так как только очень простые задачи проходят все этапы без каких-либо *итераций* — возвратов на предыдущие шаги технологического процесса. При

программировании, например, может обнаружиться, что реализация некоторой функции очень громоздка, неэффективна и вступает в противоречие с требуемой от системы производительностью. В этом случае требуется перепроектирование, а может быть, и переделка спецификаций. При разработке больших нетрадиционных систем необходимость в итерациях возникает регулярно на любом этапе жизненного цикла как из-за допущенных на предыдущих шагах ошибок и неточностей, так и из-за изменений внешних требований к условиям эксплуатации системы.

Таковы мотивы *классической итерационной модели* жизненного цикла (рис. 3). Стрелки, ведущие вверх, обозначают возвраты к предыдущим этапам, квалифицируемые как требование повторить этап для исправления обнаруженной ошибки. В этой связи может показаться странным переход от этапа «Эксплуатация и сопровождение» к этапу «Тестирование и отладка». Дело в том, что рекламации, предъявляемые в ходе эксплуатации системы, часто даются в такой форме, которая нуждается в их перепроверке. Чтобы понять, о каких ошибках идет речь в рекламации, разработчикам полезно предварительно воспроизвести пользовательскую ситуацию у себя, т.е. выполнить действия, которые обычно относят к тестированию.

Классическая итерационная модель абсолютизирует возможность возвратов на предыдущие этапы. Однако это обстоятельство отражает существенный непреодолимый аспект программных разработок, не опирающихся на объектно-ориентированное проектирование: стремление заранее предвидеть все ситуации использования системы и невозможность в подавляющем большинстве случаев достичь этого. Все традиционные технологии программирования направлены лишь на то, чтобы минимизировать возвраты. Но суть от этого не меняется: при возврате всегда приходится повторять построение того, что уже считалось готовым.

Иное положение с объектно-ориентированными технологиями. Отказ от завершенности фаз и этапов, вместо чего предлагается распределять наращивание функциональности и интерфейсных возможностей по итерациям, позволяет ослабить требование переделки старого при возвратах. По существу, классическая схема остается верной, но только в рамках одной итерации и с одной важной поправкой: все полезное, что было сделано ранее, сохраняется. Понятно, что для программной системы в целом новый подход требует и новых моделей жизненного цикла, отражающих его особенности, отмеченные ранее. Об этом будет идти речь после изучения основных вариантов традиционных моделей жизненного цикла.

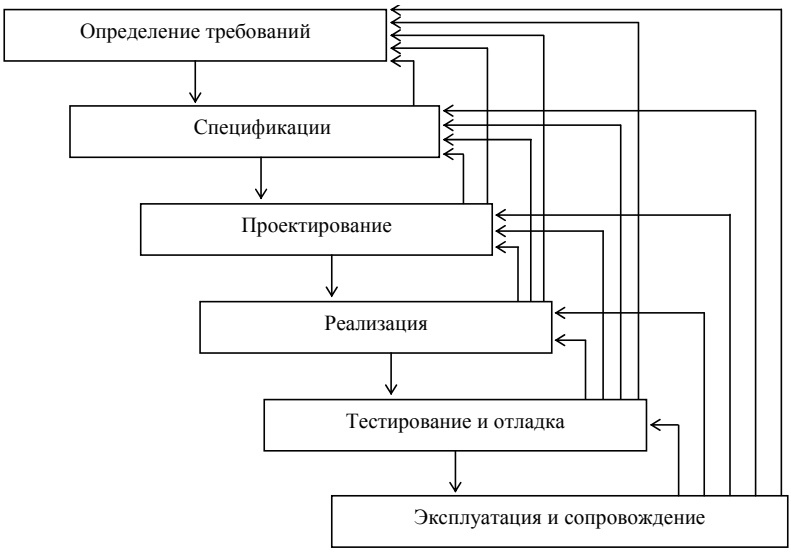


Рис. 3. Классическая итерационная модель

1.3. Каскадная модель

Некоторой более строгой разновидностью классической модели является так называемая *каскадная модель*, которую можно рассматривать в качестве показательного примера того, какими методами можно минимизировать возвраты.

Характерные черты каскадной модели:

- ♦ завершение каждого этапа (они почти те же, что и в классической модели) проверкой полученных результатов с целью устранить как можно большее число проблем, связанных с разработкой изделия;
- ♦ циклическое повторение пройденных этапов (как в классической модели).

Мотивация каскадной модели связана с так называемым *управлением качеством* программного обеспечения. В связи с ней уточняются понятия этапов, некоторые из них структурируются (спецификация требований и реализация).

На рис. 4 приведена схема каскадной модели, построенная как модификация классической итерационной модели. В каждом блоке, обозначающем этап, указано действие, которым этап завершается (наименования этих действий отмечены серым фоном). Из рисунка видно, что в этой модели тестирование не выделяется в качестве отдельного этапа, а считается лишь порогом, через который нужно перейти, чтобы завершить этап, точно так же, как и другие подобные действия.

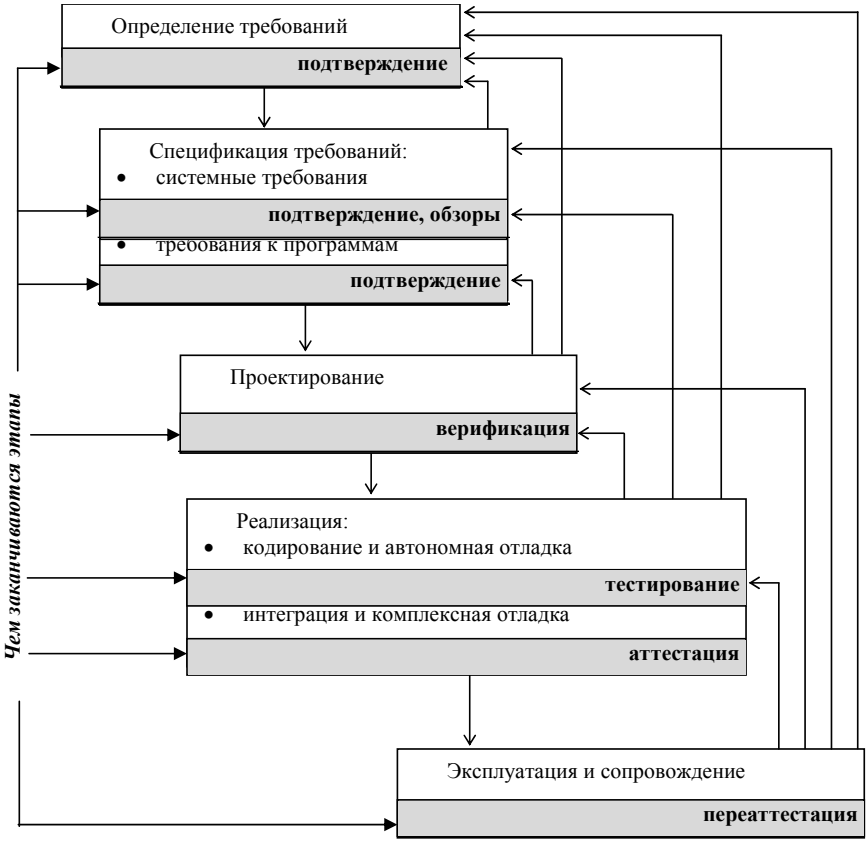


Рис. 4. Каскадная модель

В соответствии с каскадной моделью завершение этапа определения системных требований включает фиксацию их в виде специальных документов, называемых *обзорами* того, что от системы требуется (описание функций), а спецификация требований к программам — *подтверждением* выполнения зафиксированных в обзорах функций в планируемых к реализации программах. Кроме того, подтверждение предполагается и на первом этапе, т.е. после определения требований. Это отражает тот факт, что полученные требования необходимо согласовывать с заказчиком.

Результат проектирования *верифицируется*, т.е. проверяется, что принятая структура системы и реализационные механизмы обеспечивают выполнимость специфицированных функций.

Реализация контролируется путем *тестирования* компонент, а после интеграции компонент в систему и комплексной отладки проводится *аттестация*, т.е. проверка-фиксация фактически реализованных функций системы, описание ограничений реализации и т.п.

В ходе эксплуатации и сопровождения изделия устанавливается, насколько хорошо система соответствует пользовательским запросам, т.е. осуществляется *переаттестация*.

Каждая из указанных проверок может отослать разработчиков системы к повторению любого из ранее пройденных этапов, что иллюстрируется стрелками на рис. 4. В то же время, каскадная модель разработана в ответ на требование практики разработки программных проектов, в которых за счет преодоления проверочных барьеров достигается минимизация возвратов к пройденным этапам. Такая минимизация возможна не только в плане количества откатов по схеме: за счет ужесточения проверок разработчики пытаются ликвидировать прямые возвраты через несколько этапов. Соответствующая схема, называемая *строгой каскадной моделью*, представлена на рис. 5.

Поучительно проследить, как в строгой каскадной модели исправляются ошибки ранних этапов. В соответствии с данной схемой разработчики любого этапа в качестве исходных материалов для своей деятельности, т.е. *задания на разработку*, получают результаты предыдущего этапа, прошедшие соответствующую проверку (в идеале исполнители этапа могут вовсе не знать о более ранних этапах). При проведении работ этапа может быть выяснено, что задание невыполнимо по одной из следующих причин:

- оно противоречиво, т.е. содержит несовместные или невыполнимые требования;
- не выработаны критерии для выбора одного из возможных вариантов решения.

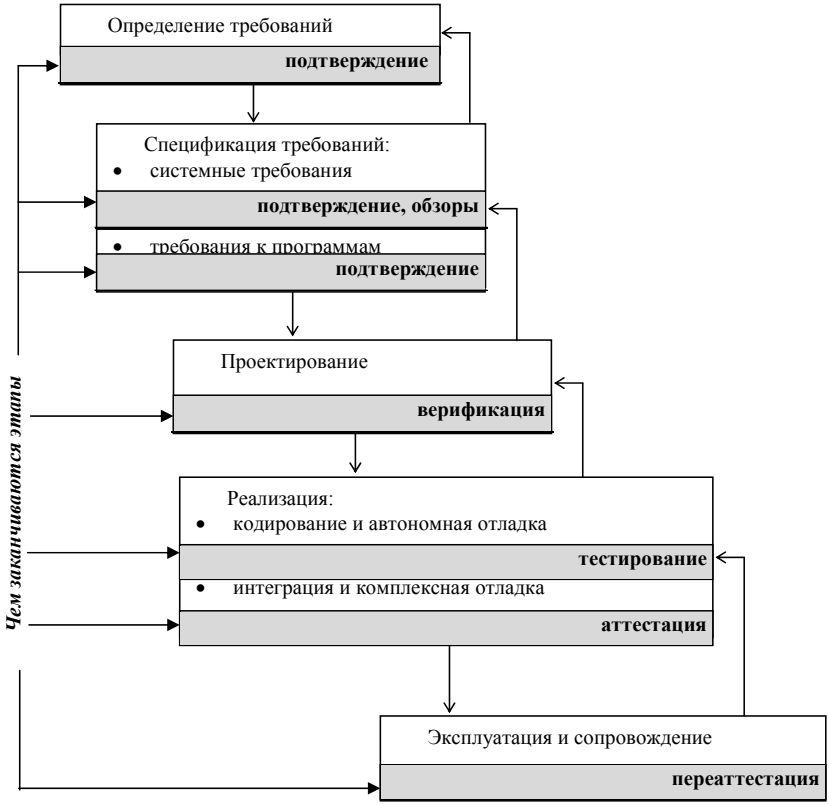


Рис. 5. Строгая каскадная модель

Обе ситуации квалифицируются как *ошибки задания*, т.е. как ошибки предыдущего этапа. Для исправления обнаруженных ошибок работы предыдущего этапа возобновляются. В результате ошибки либо ликвидируются, либо констатируется невозможность их непосредственного исправления. В первом случае работы этапа, вызвавшего возврат, возобновляются с откорректированным заданием. Второй случай квалифицируется как ошибка более раннего этапа.

Строгая каскадная модель фиксирует два важных момента жизненного цикла:

- точное разделение работ, заданий и ответственности разработчиков этапов и тех, кто, проверяя работы, инициирует переход к следующему этапу;
- малые циклы между соседними этапами, в результате которых достигается компромиссное задание.

Первый момент — это шаг к осознанию фактического разделения труда, из которого вполне осуществимо явное выделение технологических и организационных функций, выполняемых на каждом этапе. В результате появляется возможность постановки задачи автоматизированной поддержки этих функций. Второй момент можно трактовать как совместное выполнение работ соседних этапов, т.е. их перекрытие. Однако в рамках каскадной модели эти обстоятельства отражаются лишь косвенно. Продуктивность явного включения их в качестве элементов модели жизненного цикла демонстрируется в следующем разделе.

1.4. Модель фазы—функции

Чрезвычайно важным мотивом развития моделей жизненного цикла программного обеспечения является потребность в подходящем средстве для комплексного управления проектом. По существу, это утверждение указывает на то, что модель должна служить основой организации взаимоотношений между разработчиками, и, таким образом, одной из ее целей является поддержка функций менеджера. Это приводит к необходимости наложения на модель контрольных точек и функций, задающих организационно-временные рамки проекта.

Наиболее последовательно такое дополнение классической схемы реализовано в модели Гантера в виде матрицы «фазы—функции». Уже из упоминания о матрице следует, что модель Гантера имеет два измерения:

- *фазовое*, отражающее этапы выполнения проекта и сопутствующие им события;
- *функциональное*, показывающее, какие организационные функции выполняются в ходе развития проекта и какова их интенсивность на каждом из этапов.

В модели Гантера отражено то, что выполнение функции на одном этапе может продолжаться и на следующем. На рис. 6 представлено фазовое измерение модели. Жирной чертой (с разрывом и стрелкой, обозначающей

временное направление) изображен процесс разработки[2]. Контрольные точки и наименования событий указаны под этой чертой. Они пронумерованы. Все развитие проекта в модели привязывается к этим контрольным точкам и событиям.

Рис. 6. Фазовое измерение модели фазы—функции

В данной модели жизненный цикл распадается на следующие перекрывающие друг друга фазы (этапы):

- *исследования* — этап начинается, когда необходимость разработки признана руководством проекта (контрольная точка 0), и заключается в том, что для проекта обосновываются требуемые ресурсы (контрольная точка 1) и формулируются требования к разрабатываемому изделию (контрольная точка 2);

[2] Для моделей реальных проектов целесообразно длины отрезков между контрольными точками выбирать пропорционально оценкам временных соотношений между этапами. Если фактическое время выполнения этапа оказывается не соответствующим соотношениям на схеме, то это свидетельствует об ошибке планирования работ: неудовлетворительны либо предварительная оценка, либо темпы работы. Таким образом, хорошая модель жизненного цикла может рассматриваться в качестве важного инструмента планирования.

- *анализ осуществимости* — начинается на фазе исследования, когда определены исполнители проекта (контрольная точка 1), и завершается утверждением требований (контрольная точка 3). Цель этапа — определить возможность конструирования изделия с технической точки зрения (достаточно ли ресурсов, квалификации и т.п.), будет ли изделие удобно для практического использования, ответить на вопросы экономической и коммерческой эффективности;
- *конструирование* — этап начинается обычно на фазе анализа осуществимости, как только документально зафиксированы предварительные цели проекта (контрольная точка 2), и заканчивается утверждением проектных решений в виде официальной спецификации на разработку (контрольная точка 5);
- *программирование* — начинается на фазе конструирования, когда становятся доступными основные спецификации на отдельные компоненты изделия (контрольная точка 4), но не ранее утверждения соглашения о требованиях (контрольная точка 3). Совмещение данной фазы с заключительным этапом конструирования обеспечивает оперативную проверку проектных решений и некоторых ключевых вопросов разработки. Цель этапа — реализация программ компонентов с последующей сборкой изделия. Он завершается, когда разработчики заканчивают документирование, отладку и компоновку и передают изделие службе, выполняющей независимую оценку результатов работы (независимые испытания начались — контрольная точка 7);
- *оценка* — фаза является буферной зоной между началом испытаний и практическим использованием изделия. Она начинается, как только проведены внутренние (силами разработчиков) испытания изделия (контрольная точка 6) и заканчивается, когда подтверждается готовность изделия к эксплуатации (контрольная точка 9);
- *использование* — начинается в ходе передачи изделия на распространение и продолжается, пока изделие находится в действии и интенсивно эксплуатируется. Этап связан с внедрением, обучением, настройкой и сопровождением, возможно, с модернизацией изделия. Он заканчивается, когда разработчики прекращают систематическую деятельность по сопровождению и поддержке данного программного изделия (контрольная точка 10).

На протяжении фаз жизненного цикла разработчики выполняют следующие технологические (организационные) функции (классы функций):

- планирование,
- разработка,
- обслуживание,
- выпуск документации,
- испытания,
- поддержка,
- сопровождение.

Перечисленные функции на разных этапах имеют различное содержание, требуют различной интенсивности, но, что особенно важно для модели, совмещаются при реализации проекта. Это функциональное измерение модели, наложение которого на фазовое измерение дает изображение матрицы фаз—функций в целом (см. рис. 7, на котором интенсивность выполняемых функций отражается густотой закраски клеток матрицы).

Состав организационных функций и их интенсивность могут меняться от проекта к проекту в зависимости от его особенностей, от того, что руководство проекта считает главным или второстепенным. К примеру, если исходная квалификация коллектива не очень высока, в список функций может быть добавлено обучение персонала. Иногда бывает важно разграничить планирование и контроль (по Гантеру контрольные функции явно не выделяются). При объектно-ориентированном проектировании роль моделирования возрастает настолько, что его целесообразно перевести из разряда методов проектирования в явно выделенную технологическую функцию, о чем речь впереди.

Модель учитывает соотношение технологических функций и фаз жизненного цикла, чем она выгодно отличается от простых (или ограниченных?) ранее рассмотренных «идеальных» моделей. По-видимому, простота-ограниченность «идеальных» моделей есть следствие отождествления выделяемых этапов с технологической операцией, преобладающей при их выполнении. В то же время, задача отражения итеративности в модели Гантера в явном виде не предусматривается. Хотя само по себе перекрытие смежных фаз проекта и выпуск соответствующей событиям документации — путь к минимизации возвратов к выполненным этапам, более содержательные средства описания итераций в модель не закладываются.

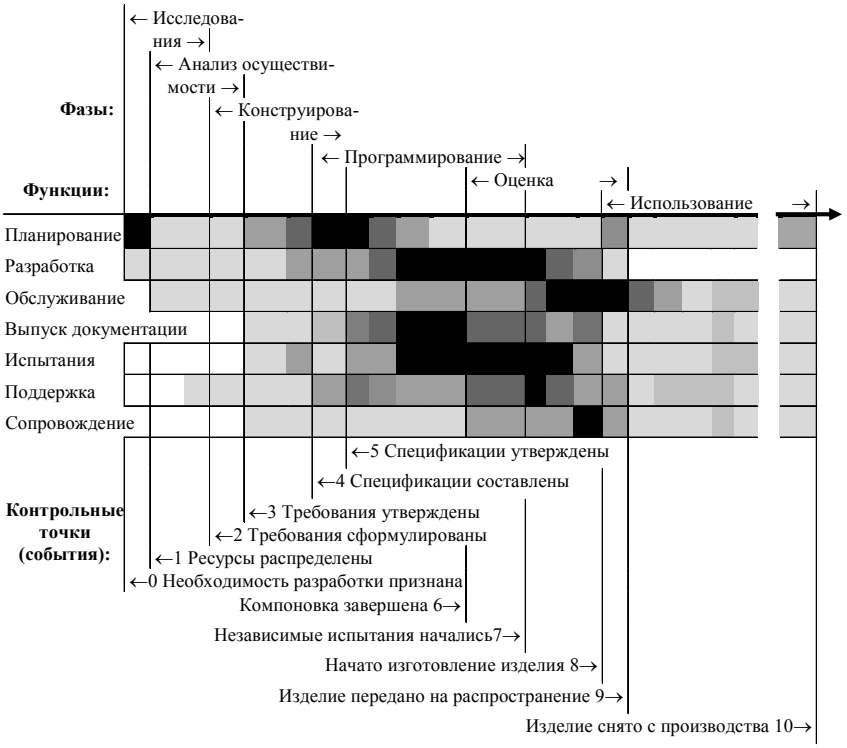


Рис. 7. Матрица фазы—функции модели Гантера

Если попытаться развить модель Гантера с целью учета итеративности, то, очевидно, придется предусмотреть *расщепление линии жизненного цикла*, как это представлено на рис. 8. Но это влечет и расщепление матрицы интенсивностей выполняемых функций: было бы необоснованно считать, что интенсивности при возвратах сохраняются. В целом, по мере продвижения разработки к своему завершению, они должны уменьшаться. Таким образом, матрица интенсивностей приобретает новое измерение, отражающее итеративный характер развития проекта.

Итеративность неизбежна при разработке сложных программных изделий, а потому ее планирование целесообразно. Однако рассматривая традиционные подходы к развитию проектов, можно заметить, что они не пыта-

ются использовать итеративность в качестве метода проектирования и стремятся лишь к минимизации возвратов.

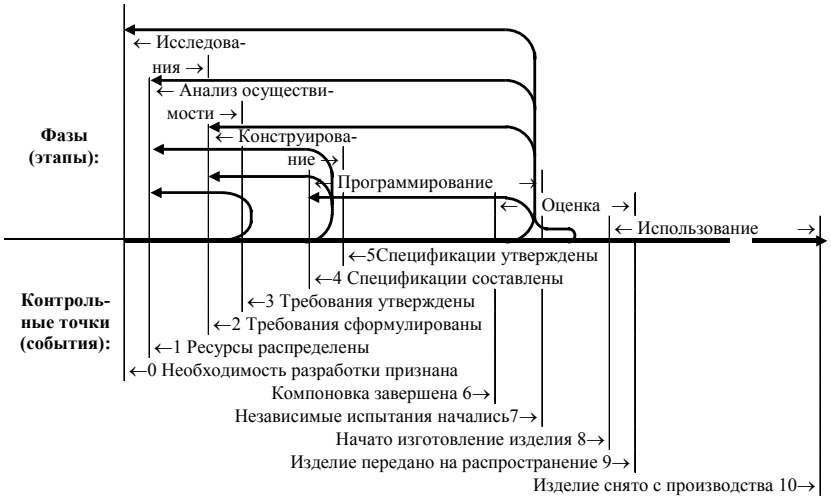


Рис. 8. Учет итеративности в модели фазы—функции
(фазовое измерение, показаны лишь некоторые возвраты)

2. ОБЪЕКТНО-ОРИЕНТИРОВАННЫЕ МОДЕЛИ ЖИЗНЕННОГО ЦИКЛА

В технологическом плане отношение к итеративности развития проекта коренным образом отличает объектно-ориентированный подход от всех последовательных методологий. Для традиционных подходов итерация — это исправление ошибок, т.е. процесс, который с трудом поддается технологическим нормам и регламентам. При объектно-ориентированном подходе итерации никогда не отменяют результаты друг друга, а всегда только дополняют и развивают их.

2.1. Принципы объектно-ориентированного проектирования

Принципиальные моменты, в которых объектно-ориентированный подход к развитию проектов стоит сопоставить с традиционными последовательными методологиями, сводятся к следующему:

- **Итеративность развития.**

  Начиная с фазы анализа и до завершения реализации, процесс объектно-ориентированного проектирования в противоположность последовательному развитию строится как серия итераций, которой возможно предшествует определенный период последовательного изучения предметной области и задач проекта в целом (этапы *определения требований* и *начального планирования*).

- **Наращивание функциональности в соответствии со сценариями.**

  Наращивание функциональности проектируемого изделия представляется как развитие сценариев, которые соответствуют описаниям (диаграммам) взаимодействия объектов и отражают отдельные стороны функционирования. Эти описания предписывают развитие на этапе программирования операционной базы проекта: она вырабатывается исходя из сценариев уровня проектирования (конструирования). Полная функциональность состоит из функциональностей всех сценариев. Таким образом, данная стратегия довольно близка классическому методу пошаговой детализации, при использовании которого функциональность наращивается путем уточнения (доопределения) модулей нижнего уровня. Однако в отличие от этого метода итеративное наращивание требует, чтобы в результате каждой итерации изделие получало полностью готовую функциональность, планируемую реализуемым сценарием. Последующие итерации добавляют уже другую функциональность, которая планируется другим сценарием.

- **Ничто не делается однократно.**

  Последовательный подход предполагает, что анализ завершен перед конструированием, завершение которого предшествует программированию. Перекрытие этапов (см. п. 1.4) ослабляет это предположение, но принципиально ситуацию не меняет. В большинстве объектно-ориентированных проектов анализ никогда не завершается в течение всего развития проекта, а процесс конструирования сопровождает разработку в ходе всего ее жизненного цикла.

♦ **Оперирование на размножающихся фазах подобно.**

Как в начале проектирования, на последующих итерациях анализ предшествует конструированию, за которым следует программирование, тестирование и другие виды работ.

При объектно-ориентированном проектировании в ходе итеративного наращивания обыкновенно выполняются вполне традиционные этапы:

- *Определение требований*, или *планирование итерации*, — фиксируется, что должно быть выполнено на данной итерации в виде описания области, для которой планируется разработать функциональность на данной итерации, и что для этого нужно. Обычно этот этап включает отбор сценариев, которые должны быть реализованы на данной итерации.
- *Анализ* — исследуются условия выполнения планируемых требований, проверяется полнота отобранных сценариев с точки зрения реализации требуемой функциональности.
- *Моделирование пользовательского интерфейса* — коль скоро итерация должна обеспечивать функционально законченную реализацию, требуется определить правила взаимодействий, необходимые для активизации требуемых функций. Модель интерфейса представляет пользовательское представление поведения объектов данной итерации.
- *Конструирование* — обычная декомпозиция проекта, проводимая в объектно-ориентированном стиле. Конструирование включает построение или наращивание иерархии системы классов, описание событий и определение реакции на них и т.д. В ходе конструирования определяются объекты, реализуемые и/или доопределяемые на данной итерации, и набор функций (методов объектов), которые обеспечивают решение задачи данной итерации.
- *Реализация* (программирование) — программное воплощение решений, принятых для данной итерации. Необходимым компонентом реализации здесь считается автономная проверка соответствия составляемых модулей их спецификациям (в частности, должно быть обеспечено требуемое поведение объектов).
- *Тестирование* — этап комплексной проверки результатов, полученных на данной итерации.
- *Оценка* результатов итерации — этап включает работу, связанную с рассмотрением полученных результатов в контексте проекта в целом. В частности, должно быть выяснено, какие задачи проекта можно решать с учетом результатов итерации, на какие ранее поставленные во-

просы получены ответы, какие новые вопросы возникают в новых условиях.

2.2. Модификация модели фазы—функции

Традиционность этапов объектно-ориентированного развития проекта в рамках одной итерации позволяет ставить задачу моделирования процесса итеративного наращивания как модификацию существующих моделей жизненного цикла. В настоящем разделе такая модификация осуществляется для модели фазы—функции Гантера.

В сравнении с моделью Гантера фазовое измерение жизненного цикла при объектно-ориентированном проектировании почти не изменяется: появляется лишь один дополнительный этап: «Моделирование пользовательского интерфейса», который в старой схеме можно рассматривать как часть этапов анализа и/или конструирования. Однако это весьма существенное дополнение, характеризующее подход в целом. Главный мотив явного рассмотрения моделирования в жизненном цикле при объектно-ориентированном развитии проектов связан со следующими двумя особенностями:

- **Распределение реализуемых требований по итерациям.**

  Совокупность сценариев, реализуемых на очередной итерации, и набор ранее реализованных сценариев всегда образуют *законченную*, хотя и *неполную версию системы*, предлагаемую пользователям. По разным причинам, в том числе для исключения двусмысленностей в понимании, необходимо представление планируемого для реализации в виде моделей, согласующих взгляд на систему со стороны пользователей (а также заказчиков и других заинтересованных лиц) с точкой зрения разработчиков. Эти модели появляются в ходе этапа анализа, что отражается в их названии: *модели уровня анализа*.

- **Особый стиль наращивания возможностей системы и ее развития.**

  Представление системы как набора взаимосвязанных различными отношениями классов — основа декомпозиции проекта при объектно-ориентированном подходе. Каждая новая итерация расширяет этот набор путем добавления новых классов, вступающих в определенные отношения с ранее построенной системой классов. Выполнить такое расширение корректно без абстрагирования от деталей реализации существующего, а если учитывать перспективу, то и без такого же абстрактного представления добавляемых классов практически невозможно. Ины-

ми словами, требуется построение *моделей уровня конструирования*, которые задают реализационное представление проектируемой системы.

В приведенном выше перечне этапов жизненного цикла итерации при объектно-ориентированном подходе явно выделено моделирование уровня анализа, которое сводится к построению модельного представлению сценариев. Но это только один аспект проектного моделирования. Как было только что показано, другой, не менее существенный аспект моделирования, проявляется при конструировании. Наконец, есть еще третий аспект моделирования, связанный с предъявлением каждой версии программного изделия пользователю, представление которого о системе, разумеется, не имеет отношения к моделям уровня конструирования и лишь косвенно связано с моделями уровня анализа. Таким образом, если следовать гантеровскому стилю описания жизненного цикла, то правильнее будет выделять не этап моделирования (как это, следуя уже сложившейся традиции, чаще всего делают), а *технологическую функцию моделирования*, пронизывающую весь процесс разработки проекта.

В новой схеме жизненного цикла появляется строго регламентированное расщепление, единственное для всей последовательности работ (рис. 9). Но этот маршрут отражает не корректировку ошибочно принимаемых решений, а вполне запланированный акт, фиксирующий то, что в ходе выполнения итераций происходит наращивание возможностей изделия.

Следует отметить еще одну модификацию схемы Гантера, отраженную на рисунке. В рамках этапа оценки выделен специальный вложенный этап *Пополнение базового окружения проекта*, смысл которого сводится к планированию и реализации переиспользования программного обеспечения. Любой объектно-ориентированный проект развивается исходя из некоторой уже существующей среды классов и других компонентов. Это *базовое окружение* проекта интенсивно используется и, в свою очередь, пополняется средствами, возникающими в результате итеративного наращивания. Полезность сохранения таких средств для текущего проекта и для последующих разработок очевидна. В этой связи целесообразно в рамках выполнения этапа оценки производить дополнительную работу, направленную на сохранение полезного в депозитарии проектов.

По вполне понятным причинам в объектно-ориентированном проектировании несколько изменяется содержание ряда этапов, что нашло свое отражение в количестве и наименованиях событий на рисунке.

Рис. 9. Фазовое измерение модели жизненного цикла при объектно-ориентированном развитии проекта

Обсуждая модель жизненного цикла при объектно-ориентированном развитии проекта, необходимо указать на работы, которые выходят за рамки стандартизованного итерационного процесса. Это *начальная фаза проекта*, которая выполняется на старте в ходе исследований и анализа осуществимости, и *фаза завершения проекта* (*итерации*), с выполнением которой работы над проектом (над итерацией) заканчиваются.

Смысл работ начальной фазы — общее планирование развития проекта. Помимо традиционного содержания, вкладываемого в этапы определения требований к проекту в целом, они должны стать основой разработки еще в двух отношениях:

- требуется определить *ближайшую задачу и перспективные задачи проекта*. Первая из них — задача первой итерации, в ходе которой, в частности, готовится первый рабочий продукт, предъявляемый заказчику. С точки зрения развития проекта, решение ближайшей задачи

должно обеспечить осуществимость последующего итеративного наращивания возможностей системы (об этом разговор еще предстоит). От качества этих двух результатов зависит судьба проекта в целом. Перспективные задачи — это планируемое развитие, которое допускает корректировку в дальнейшем;

- требуется выбрать *критерии оценки результатов* итераций. Эти критерии могут варьироваться в зависимости от направленности проекта, прикладной области и других обстоятельств.

Фаза завершения проекта (итерации) охватывает часть жизненного цикла, которая отражает деятельность разработчиков, связанную с рабочими продуктами итерации после получения результатов. Она вполне аналогична традиционной фазе эксплуатации и сопровождения, однако есть и отличия, обусловленные тем, что объектно-ориентированный проект обычно имеет дело с иерархиями версий системы, отражающими наращивание возможностей. Данная фаза перекрывается с этапом оценки.

Традиционные работы фазы завершения включают в себя:

- поставку, или пакетирование изделия для потребителя (контрольная точка 9 на рис. 9);
- сопровождение программного продукта (по причине разнообразия вариантов организации этих работ они редко описываются структурно, т.е. с разбиением на этапы);
- этап окончания работ (контрольные точки 11, 12): оповещение о прекращении сопровождения и сворачивание деятельности по поддержке версии (версий)[3].

Как уже говорилось, для объектно-ориентированного проектирования существенными являются работы, связанные с переиспользованием рабочих продуктов. До фазы завершения переиспользование обычно рассматривается для текущего проекта (этап пополнения базового окружения). После того, как приложение (рабочий продукт итерации) используется некоторое время, и оно может рассматриваться как *готовое*, в рамках данной фазы осуществляется:

[3] Этап окончания работ мог бы быть представлен во всех традиционных моделях, но в то время, когда эти модели разрабатывались, ему не придавали особого значения. Вместе с тем, когда речь идет о совместной поддержке нескольких версий (а именно такая ситуация типична для объектно-ориентированного проектирования) окончание работ игнорировать нельзя.

- выделение общих (т.е. непривязанных к проекту) переиспользуемых компонентов (обычно эти работы связываются с событием передачи системы на распространение — контрольная точка 10).

Одним из существенных моментов объектно-ориентированного проектирования является отказ от традиционного постулата о том, что все требования к системе сформулированы заранее. Следовательно, при моделировании жизненного цикла вообще и его фазы завершения в частности нужно учитывать обработку потока внешних требований на всех этапах. Этому вопросу еще будет уделено внимание, а пока можно считать (как чаще всего и бывает), что требования, поступающие на фазе завершения итерации, рассматриваются как относящиеся к следующим итерациям, т.е. к следующим версиям системы. В таком случае завершение итерации означает сопровождение программного изделия, а затем окончание работ с данной версией. Пожелания к развитию проекта в этот период учитываются как требования к последующим (возможно, еще не начатым) итерациям. Окончание проекта рассматривается как отказ от сопровождения всех версий системы. Стоит сопоставить это положение с традиционными подходами к проектированию, когда учет пожеланий к системе в процессе ее эксплуатации чаще всего означает одно: организацию нового проекта (быть может, специального), цель которого — учет новых требований.

Несколько слов о функциональном измерении в модифицированной для объектно-ориентированного подхода матрице фазы—функции. Как было показано выше, целесообразно список технологических функций расширить за счет моделирования. Соответственно, следует определить в матрице Гантера строку интенсивностей для этой функции. В предположении о сохранении распределения интенсивностей других функций (рис. 7) распределение интенсивности для модифицированной модели жизненного цикла можно задать так, как это сделано на рис. 10, который показывает новый вид модели целиком (на рисунке контрольные точки жизненного цикла указаны своими номерами без пояснений).

Представленные распределения интенсивностей нельзя абсолютизировать. Наивно было бы предполагать стабильность интенсивностей технологических функций по итерациям. Следовательно, весь цикл развития проекта в матричном, двумерном представлении модифицированной гантеровской модели изобразить не удастся: оно не может показать изменение интенсивностей технологических функций при переходе от одной итерации к другой. По этой причине предлагается распределение интенсивностей технологических функций рассматривать как «среднестатистическую» интегральную по итерациям тенденцию. Практическая полезность рассмотрения

функционального измерения — не в конкретном распределении интенсивностей технологических функций в реальных проектах, а в том, что оно заставляет руководство проекта думать о расстановке сил в коллективе разработчиков и вообще о правильном распределении кадровых ресурсов проекта.

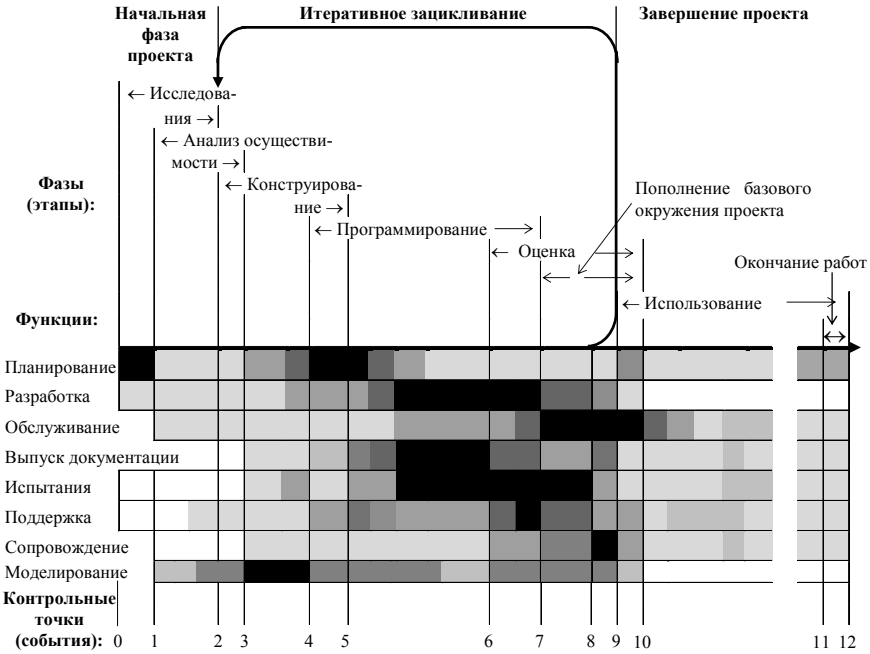


Рис. 10. Модель фазы—функции, модифицирования для объектно-ориентированного развития проекта

2.3. Параллельное выполнение итераций

Любой программный проект, заслуживающий привлечения менеджера для поддержки разработки, — это процесс, развиваемый коллективно. Следовательно, уместно ставить вопрос, как должна отражаться в модели жизненного цикла одновременность деятельности исполнителей коллектива. По вполне понятным причинам, это является одним из мотивов разработки моделей.

В модели, следующей гантеровской схеме фазы — функции, это качество процесса разработки программного изделия отражено с помощью функционального измерения, показывающего, какие технологические функции выполняются одновременно. В рамках объектно-ориентированного подхода явно выделяется еще один вид технологического параллелизма: одновременная разработка нескольких итераций разными группами исполнителей (словосочетание «разные группы» не надо понимать буквально — по существу, это групповые роли, и конкретная группа исполнителей вполне может одновременно отвечать за разработку сразу нескольких итераций).

Технологический параллелизм означает принципиальную осуществимость одновременной разработки нескольких итераций. Однако это не означает разрешения механического их слияния, поскольку итерации зависят одна от другой. К примеру, невозможно наращивание еще не построенной системы классов, нельзя использовать функцию с неизвестными условиями ее корректного выполнения. Говоря о совмещении работ, нужно всегда знать подобные и другие виды зависимостей. Следует различать следующие области:

- *область недопустимого совмещения* — когда выполнение одной работы непосредственно зависит от результатов другой работы;
- *область возможного совмещения* — когда зависимость ослаблена тем, что ожидаемые результаты предшествующей работы хорошо описаны (например, построены и проверены модели этапов конструирования, хотя программирование еще не выполнено);
- *область рационального совмещения* — когда зависимость работ фактически тем или иным способом экранирована (предшествующая работа выполнена, хотя, быть может, не до конца проверена, составлен и проверяется протокол взаимодействия работ и др.).

Одновременность выполнения разных итераций можно представить в виде схем, показанных на рис. 11.

На рис. 11 а) приведена расшифровка этапов итераций. По сравнению с общей моделью (рис. 10), здесь представлено более мелкое дробление этапов: явно выделены планирование, которое для начальной итерации является частью общего этапа анализа осуществимости, и тестирование как перекрывающаяся часть общих этапов программирования и оценки.

Рис. 11 б) демонстрирует три одновременно выполняемые итерации: вторая начинается в ходе выполнения программирования первой итерации с таким расчетом, чтобы ее этап программирования начался после окончания тестирования первой итерации. Планирование третьей итерации начинается одновременно с этапом программирования второй итерации.

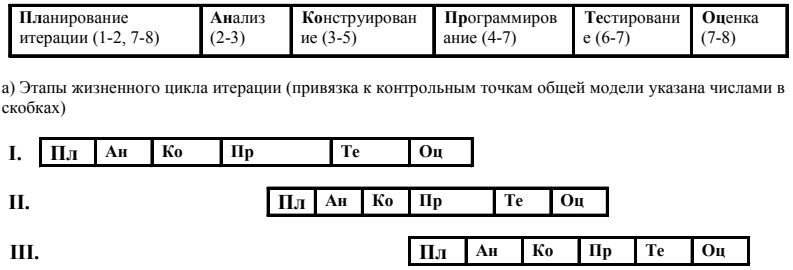


Рис. 11. Распараллеливание выполнения итераций проекта

Рис. 11 в) показывает области недопустимого, возможного и рационального совмещений, а также область последовательного выполнения двух итераций. *Недопустимость совмещения* означает, что для планирования очередной итерации нет достаточно полной информации, как следствие, оно не может быть выполнено эффективно. В ходе конструирования наступает момент, когда такая информация появляется, следовательно, появляется возможность активизации работ над новой итерацией. Определение области *рационального совмещения* работ двух итераций отражает то, что было бы неразумно начинать этап программирования новой итерации, когда рабочий продукт предыдущей итерации не протестирован (совмещение, изображенное на рис. 11 б) удовлетворяет этому условию). Область последовательного выполнения указывает на то время, которое соответствует началу следующей итерации после завершения работ над предыдущей (совмещения нет).

Определение перечисленных областей повышает гибкость распределения времени выполнения проекта. Тем не менее, планируя работы, лучше

не рассчитывать на совмещения итераций, а оставлять эту возможность как резерв временного ресурса проекта. Таким образом, оказывается, что итеративность объектно-ориентированного проектирования обладает дополнительной устойчивостью к рискам невыполнения проектного задания.

2.4. Моделирование итеративного наращивания возможностей системы

В предыдущих моделях жизненного цикла объектно-ориентированного программного обеспечения не был наглядно выделен важный аспект подхода: постепенное наращивание возможностей системы по мере развития проекта. Для его отражения можно предложить представление жизненного цикла в виде *спирали развития*, которая показана на рис. 12[4].

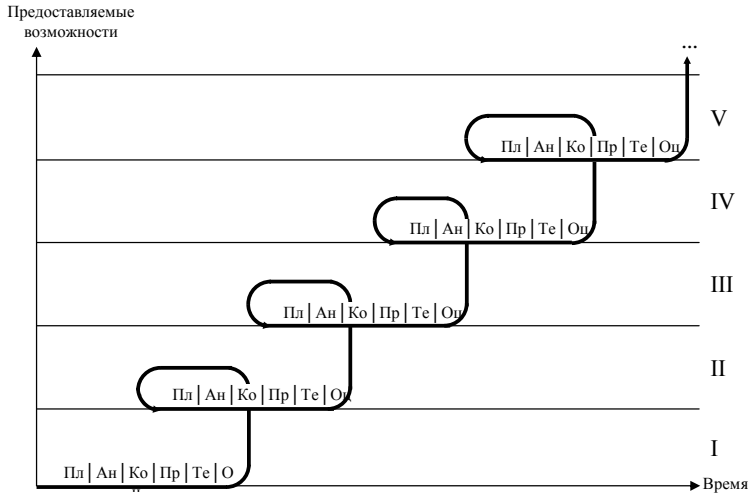


Рис. 12. Спираль развития объектно-ориентированного проекта

На рисунке горизонтальные отрезки с пометками, имеющими тот же смысл, что и в предыдущей модели, — это итерации. Они помещены в про-

[4] В несколько модернизированном виде здесь приводится ставшая классической модель Г. Буча.

странство предоставляемых в зависимости от времени возможностей системы. Линии, параллельные временной оси, отображают уровни пользовательских возможностей, реализуемых на итерациях (римскими цифрами справа указаны номера итераций). Стрелки-переходы между итерациями учитывают условия совмещения работ, о которых шла речь выше. Этой моделью подчеркивается тот факт объектно-ориентированного развития проектов, что возможности, предоставляемые очередной итерацией, никогда не отменяют уровня, достигнутого на предшествующих итерациях.

Постепенное наращивание возможностей системы по мере развития проекта часто изображают в виде спирали, раскручивающейся на плоскости от центра, как это показано на рис. 13. В соответствии с этой простой (грубой) моделью развитие проекта описывается как постепенный охват все более расширяющейся области плоскости по мере перехода проекта от этапа к этапу и от итерации к итерации. По существу, данная модель делает акцент на том, что объектно-ориентированное развитие приводит к постепенному расширению прикладной области, для которой используются конструируемые рабочие продукты.

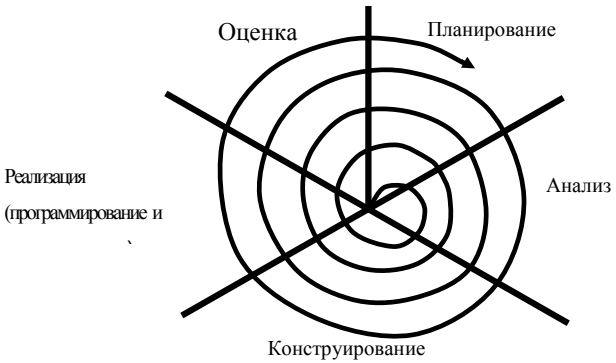


Рис. 13. Модель расширения охвата прикладной области объектно-ориентированной системой

Про объектно-ориентированное развитие проектов часто говорят, что оно предполагает, что традиционные этапы жизненного цикла разработки программной системы никогда не кончаются. Модель раскручивающейся спирали наглядно показывает смысл этого тезиса.

В данной модели можно усмотреть еще один аспект конструирования программных систем — типичную схему развития коллектива разработчиков, который, начиная от первого своего проекта, постепенно пополняет накапливаемый багаж переиспользуемых в разных системах компонентов.

В отличие от предыдущих моделей, обе спиралевидные модели никак не отражают тот факт, что у проекта есть фаза завершения. Как следствие, они предполагают, что все модификации какой-либо версии программной системы, которые требуются после ее выпуска, будут относиться к одной из следующих версий. На практике очень часто это положение нарушается: приходится поддерживать (и в частности, модифицировать) сразу несколько версий системы.

3. ТРЕБОВАНИЯ К ПРОГРАММНОМУ ИЗДЕЛИЮ И ЖИЗНЕННЫЙ ЦИКЛ

В настоящем разделе рассматривается еще один очень важный мотив моделирования жизненного цикла программных изделий. Речь идет о более пристальном внимании к изучению требований при развитии программных проектов. Традиционные технологии рассматривают определение и анализ требований в рамках предварительного этапа, предшествующего собственно разработке, за который выявляется вся информация для последующего конструирования. Утверждается, что успешность дальнейшей работы над проектом прямо зависит от того, насколько полно и тщательно выполнен аналитический этап, что внесение корректив в зафиксированные требования приводит к необходимости повторения проектирования и всех других последующих этапов. Иными словами, изменение требований в процессе разработки рассматривается как ошибка аналитического этапа. Однако эта парадигма явно противоречит практике, что нашло отражение в известном афоризме: любая полезная программа нуждается в модификациях, а бесполезная — в документации.

Более соответствуют реальному положению технологии программирования, основывающиеся на методе итеративного наращивания предоставляемых пользователям средств системы, в частности, на базе объектно-ориентированного проектирования. Как уже было сказано, в таких технологиях постулируется, что все этапы разработки системы рассредоточиваются по итерациям, каждая из которых завершается предъявлением продукции, реализующей не все, а только выделенные для нее требования. Соответст-

венно, на следующих итерациях этапы анализа, конструирования и т.д. продолжаются, а не повторяют пройденное в стиле исправления ошибок.

Понятно, что проблемы, связанные с определением и анализом требований, не исчезают и в этом случае, но благодаря специальной организации труда преодолевать трудности можно с меньшими затратами. Именно это обстоятельство побуждает к исследованию моделей жизненного цикла, которые более точно отражают рациональные схемы анализа и оперирования с требованиями.

3.1. Проблемы определения и анализа требований

В наиболее общем виде понятие требований сводится к следующим двум аспектам, фиксируемым для выполнения конструкторских работ:

- средства программного изделия, в которых нуждается пользователь для решения своих проблем или достижения определенных целей;
- характеристики программного изделия, которым должна обладать система в целом или ее компонент, чтобы удовлетворять соглашениям, спецификациям, стандартам или другой формально установленной документации.

Даже это не очень точное определение понятия требования указывает, что в реальности очень трудно, исходя из аморфных и противоречивых пожеланий, выявить, что конкретно и в каком виде должно быть воплощено в программном изделии. Требования первичны по отношению к программной разработке, определяют все ее развитие, являются начальным звеном в слагаемых качества конструируемых программ. А потому задача *управления требованиями* должна рассматриваться в качестве одной из главных задач проекта, претендующего на реальную полезность для пользователя.

Основные проблемы управления требованиями, с которыми приходится сталкиваться при их анализе, сводятся к следующему.

♦ **Требования имеют много источников.**

Даже если программная система разрабатывается по заказу, существует широкий круг людей, так или иначе заинтересованных в развитии проекта. Это, разумеется, и будущие пользователи, и заказчик, и другие лица, которые осознают как необходимость автоматизации деятельности с помощью данной системы, так и рамки, за которые выходить не стоит. Указанные и другие персоналии, в частности, сами разработчики и их руководители имеют свои, как правило, взаимно противоречивые представления о задачах проекта. Это тем более так, когда разработка претендует на удовлетворение рыночной потребности. Лица, от которых

зависит, какие работы целесообразны для реализации в проекте, называются *инициаторами работ*;

- **Требования не всегда очевидны.**

Смысл утверждения в том, что инициаторы работ далеко не всегда знают, какими средствами должна обеспечиваться поддержка автоматизируемой деятельности, в каких интерфейсных формах эта поддержка должна быть выражена. Очень часто не получается четко выделить и саму автоматизируемую деятельность;

- **Требования не всегда легко выразить словами.**

Интуитивное представление о том, какие средства должны предоставляться, чаще всего не формулируются явно. Приводится множество противоречивых примеров, наводящих соображений, но не описание нужных средств. В этой связи одна из главных задач анализа — представить требования в виде согласованных между заказчиком и разработчиками (одинаково понимаемых) утверждений, схем, диаграмм, моделей и т.п.;

- **Существует множество различных типов требований и различных уровней их детализации.**

Совокупность требований весьма многопланова и соотносится с различными аспектами проекта. Следовательно, одной из задач анализа является типизация имеющихся сведений о требованиях и распределение их по этапам и итерациям разработки;

- **Требования чаще всего взаимосвязаны и взаимозависимы, иногда противоречивы.**

Связи между требованиями обусловлены, в первую очередь, тем, что пожелания к разработке даются в системе понятий, которая исходит из предметной области и поведения пользователя, решающего задачи из этой области. Не следует ожидать, что связи между требованиями будут хорошо отслежены, что заранее будет сформулирована система объектов, которые воплощаются в программном изделии. Все, на что можно рассчитывать, получая сведения о требованиях, — это неформальное представление о том, кто будет работать с системой и зачем ему это нужно. Как следствие, в задачу анализа входит выявление взаимосвязей и взаимозависимостей, устранение противоречий путем выработки рациональных компромиссов;

- **Требования всегда уникальны.**

При формулировке требований как регламента разработки всегда должны быть найдены свойства или значения свойств, по которым они различаются: не существует двух равнозначимых требований. Это не

так, если рассматривать исходный материал для требований. Тем не менее, не следует сразу отбрасывать некоторое новое требование только по причине того, что оно кажется похожим на ранее рассмотренные. Необходимо проанализировать, какие дополнительные стороны оно характеризует, и выявить аргументированный ответ на вопрос, действительно ли данное требование является новым. По существу, утверждение об уникальности требований означает то, как они должны быть представлены в проекте в результате анализа (требование к требованиям);

- **Набор требований чаще всего является компромиссом.**

Это компромисс между пожеланиями инициаторов работ, направленный на максимально возможное расширение сферы применения системы. Существует много заинтересованных людей, чьи усредненные требования должны быть удовлетворены в рамках выполняемых ими функций в прикладной области;

- **Требования изменяются.**

Фиксируемые в заказе на разработку требования к системе, претендующей на широкую сферу применения и долгую жизнь, не являются застывшими и незыблемыми. Они изменяются как из-за учета новых факторов и пожеланий, так и в связи с выявлением особенностей проекта в ходе его разработки. Следовательно, необходимо так строить аналитическую работу, чтобы иметь возможность оперативно изменять получаемые результаты, учитывать в них изменения и дополнения исходной информации;

- **Требования зависят от времени.**

Это положение указывает на то, что пробное и экспериментальное знакомство с первыми получаемыми результатами (программными и документными) вероятно повлечет за собой корректировку требований. Как следствие, нужно иметь в виду, что при выпуске очередной версии промежуточных рабочих продуктов, при переходе от релиза к релизу вполне реальна ситуация проведения анализа требований вновь, а потому анализ и следующие за ним этапы должны быть организованы так, чтобы минимизировались переделки программ и документов.

Список проблем, связанных с требованиями, легко продолжить, но уже и этого достаточно, чтобы понять, что необходимы специальные приемы и методы оперирования с потоками требований, сопровождающих развитие проекта. Применительно к настоящей работе следует выделить то, как эти обстоятельства отражаются на моделях жизненного цикла развивающихся проектов. Существенно, что учет появляющихся требований приводит к

необходимости продолжения аналитических работ за пределами этапа анализа. Это можно делать по-разному, но всегда приходится выполнять так называемую *трассировку требований*, обсуждению которой посвящен следующий раздел.

3.2. Трассировка требований

Независимо от уровня первоначальной проработки требований к проекту, не стоит рассчитывать, что требования всегда будут оставаться неизменными. Необходимо быть готовым к тому, что в любой момент развития появятся новые требования, некоторые старые требования изменятся, другие — отпадут. Но основная сложность управления процессом изменения требований не в этом, а в том, что изменения одних требований влияет на другие и нужно отслеживать такие влияния. Влияние изменений требований естественным образом распространяется на все рабочие продукты проекта, в том числе на программные рабочие продукты.

Любое предложение по развитию конструируемой системы может быть классифицировано как требование одного их трех видов:

- *дополнительное требование*, которое отражает ранее не рассмотренный аспект системы;
- *модифицирующее требование*, которое изменяет одно или несколько уже существующих требований;
- *отменяющее требование*, принятие которого исключает одно или несколько уже существующих требований.

Вид требования отражает различия анализа нового требования в контексте существующих соглашений. Целью такого анализа является поддержка целостности системы требований: нахождение противоречий между требованиями и достижение приемлемых компромиссов. Следует отметить, что требования могут оказаться противоречащими не только друг другу, но и уже принятым проектным решениям. В работах с меняющимися требованиями большое место занимает отслеживание связей проекта, благодаря которому определяется деятельность, необходимая как для реализации требований, так и для распространения изменений, связанных с требованиями по проекту.

Таким образом, вопрос о том, принять или отклонить требование, является очень ответственным, зачастую влекущим за собой цепь связанных решений на всех уровнях проектирования. Чтобы сделать получение ответа на него обоснованным, необходимо выполнение как минимум двух условий:

- требования должны быть заданы в виде, допускающем однозначное представление в моделях уровня анализа и конструирования, и способ такого представления унифицирован для всего проекта;
- в проекте должны инструментально поддерживаться связи между требованиями и другими компонентами рабочих продуктов, и эта поддержка должна быть обеспечена.

Представление требований и пожеланий, исходящих от инициаторов работ, ни в коей мере не способствует соблюдению указанных условий. Следовательно, они должны быть трансформированы, т.е. преобразованы к виду, приспособленному для анализа. Прохождение исходного требования через последовательность трансформаций от одного представления к другому, сопровождающееся соответствующим анализом, называется *трассировкой требования*. Основное назначение трассировки в том, чтобы в любой момент развития проекта сохранялась целостность и непротиворечивость конструируемой системы, реализующей принятые требования[5].

Трассировка — это основной инструмент анализа, проводимого в рамках управления изменениями требований. В первую очередь трассировке подвергаются требования, предъявленные первоначально, т.е. до того, как проект начал развиваться. Но было бы неправильно ограничиваться только ими, поскольку их связи с другими требованиями как явные, так и обнаруживаемые в ходе анализа, также требуют соответствующего анализа и других работ, связанных с реализацией требований.

В результате трансформаций строятся представления требований, вид которых приспособлен для выяснения целесообразности реализации требований. Если на некотором уровне трансформаций установлено, что данное требование отвергается, то дальнейшие преобразования его не производятся. Выделяются следующие представления требований:

1. *Исходное представление* — текстовое описание пожеланий к системе, заданное в свободной форме. Это описание, в частности, может фактически содержать одновременно несколько требо-

[5] Следует обратить внимание на то, что целостность и непротиворечивость — не характеристики принимаемых требований, а качества, которыми должна обладать конструируемая система. При построении системы, предназначенной для практического применения, всегда достигается компромисс между возможно противоречащими друг другу требованиями. Противоречия предъявляемых требований есть следствие различий интересов инициаторов работ, очень часто именно они становятся стимулом для поиска новых решений, для перехода от одной версии системы к другой, т.е. являются источником развития системы.

ваний, отражающих разные аспекты проекта, — *элементарные составляющие требования*.

2. *Унифицированные представления* — исходное представление требования разбивается на элементарные составляющие, которые описываются в виде, приспособленном для дальнейшего использования на всех проектных уровнях. В частности, здесь могут применяться формализованные описания элементарных составляющих требований. Во всяком случае, на уровне унифицированного представления достигается однозначность понимания требований.
3. *Типизированное представление* — каждое из элементарных составляющих требования приписывается к некоторому типу. В результате формируется набор атрибутов элементарных требований и их значений. Эта информация допускает почти формальное сопоставление элементарных требований с различными требованиями, уже представленными в проекте. Сопоставление проводится на разных уровнях иерархии типов требований к системе.
4. *Модельные представления уровня анализа* — образы элементарных требований как элементы аналитических моделей системы: моделей ситуаций использования и динамики взаимодействий, которые используются для оценки требований.

Если требование принимается на уровне анализа, то трассировка продолжается на следующих уровнях, и можно говорить о продолжении последовательности трансформаций в реализации требования в компонентах программного изделия:

5. *Модельные представления уровня конструирования* — образы элементарных требований в диаграммах классов, состояний и других компонентах архитектуры системы. На этом уровне требования принимаются или отклоняются в зависимости от их соответствия уже разработанной части проекта.
6. *Программные представления* — программные рабочие продукты и их фрагменты, которые рассматриваются в качестве образов требований, представленных очередной версией системы.
7. *Документные представления* — фрагменты документов, сопровождающих программный код и предназначенных для поддержки деятельности пользователей.

Схема на рис. 14 иллюстрирует приведенную последовательность трансформаций. Первые три представления требований изображены в виде

совокупностей стрелок, которые при переходе от одного представления к другому становятся все более упорядоченными.

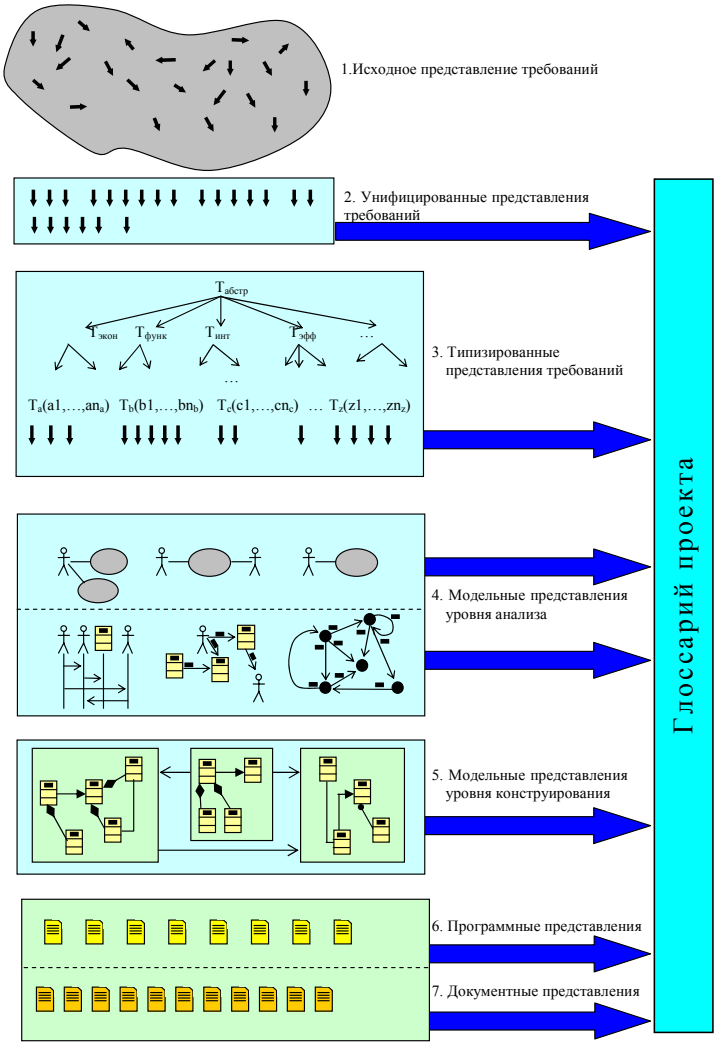


Рис. 14. Схема трансформации требований

Иерархия типов требований представлена на рисунке следующим образом. Верхний уровень — это абстрактный тип, свойства которого присущи требованиям всех типов (они сводятся к стандартизованному набору операций объединения, пересечения атрибутов, сравнения значений атрибутов и др.). Можно сказать, что $T_{абстр}$ задает регламент, которого следует придерживаться при оперировании с требованиями. Следующий уровень содержит четыре обязательных типа: $T_{экон}$, $T_{функ}$, $T_{инт}$ и $T_{эфф}$, которые объединяют требования экономического характера (пределы стоимости, рентабельность и пр.), функциональные требования, требования к интерфейсу и эффективности. Многоточием обозначены типы, которые, добавляются из-за специфики проекта. $T_a(a1,\ldots,an_a)$, $T_b(b1,\ldots,bn_b)$, $T_c(c1,\ldots,cn_c)$, …, $T_z(z1,\ldots,zn_z)$ — это конкретные типы, к которым приписываются элементарные составляющие требований (в скобках указаны их атрибуты).

Модельные представления уровней анализа и конструирования изображены в виде условных схем различных видов. Программные и документные представления — это текстовые файлы, пиктограммы которых показаны на рисунке.

На схеме представлен блок, обозначающий *глоссарий проекта*. Это очень важный технологический инструмент согласования понятий, используемых в программной разработке. Глоссарий может пополняться на любой стадии трассировки требований, когда появляются новые понятия, смысловую трактовку которых нужно зафиксировать. Тем самым глоссарий отражает текущее понимание проекта в целом. Важно подчеркнуть, что когда разработчики игнорируют деятельность по ведению глоссария, система понятий проекта все равно складывается, но стихийность этого процесса приводит к дополнительным издержкам коммуникаций работников.

Приведенная схема наглядно показывает то, что в той или иной форме вынуждены делать разработчики для преодоления трудностей управления требованиями. Она может рассматриваться в качестве проекции жизненного цикла на задачи анализа требований. Каждое требование, поступающее для анализа, проходит вполне традиционные этапы жизненного цикла, правда, в несколько специфичном виде: учитываются только те работы, которые имеют отношение к моделированию требований. Но эта абсолютизация трассировки не является недостатком. Напротив, явное выделение задач управления требованиями уже само по себе способствует более успешному их решению.

3.3. Учет трассировки требований в модели жизненного цикла

Трассировка требований и распространение изменений, связанных требованием, — это еще один мотив развития моделей жизненного цикла. При построении модели следует указать этапы, когда производятся те или иные шаги, связанные с трассировкой. По этой причине следует различать два варианта работы с требованиями в объектно-ориентированном проекте.

1. Требование или группа требований обрабатываются до начала работ над итерацией;
2. Требование или группа требований поступают, когда работы итерации начались.

Первый вариант полностью укладывается в схему модифицированной модели фазы — функции (рис. 9, 10). Если требование (группа требований) принимается для данной итерации и используется при разработке сценария, который будет реализовываться (контрольные точки 2, 8), то указанные на схеме трассировки работы включаются в аналитическую и конструкторскую деятельность. В противном случае оно либо откладывается до последующих итераций, либо отклоняется.

Второй вариант прерывает последовательный процесс выполнения итерации — необходима немедленная реакция на поступающие требования, после которой (а во многих случаях и параллельно с которой) прерванный процесс выполнения итерации возобновляется. По существу, выполняется *мини-цикл* обработки требований, который нужно изобразить в качестве дополнительного элемента модели, описывающей объектно-ориентированное развитие проекта с учетом трассировки. При этом в модели, как и в первом варианте, следует отразить возможные результаты анализа требования:

- *требование отклоняется* — работа с требованием прекращается;
- требование принимается к реализации на текущей итерации;
- реализация требования откладывается до следующих итераций.

На рис. 15 показано фазовое измерение модифицированной матрицы Гантера (рис. 9, 10), дополненное *мини-циклом* обработки одного требования или группы требований, обрабатываемых совместно. Контрольные точки (события) в данной модели те же, что и в прежней матрице фазы — функции. При построении модели используется прием, который ранее (при учете итеративности в модели — см. п. 2.4) был назван расщеплением линии жизненного цикла. Следует обратить внимание на прерывистую часть линии, ведущей от точки принятия решения к линии итеративного зацикливания. Она выделена, чтобы отразить тот факт, что для анализируемого

требования, реализация которого отложена до одной из последующих итераций, работы этапа программирования не проводятся. Возобновление непрерывности линии указывает, что на этапе оценки для данного требования начинаются работы по обоснованию включения его в планы реализации одной из будущих итераций.

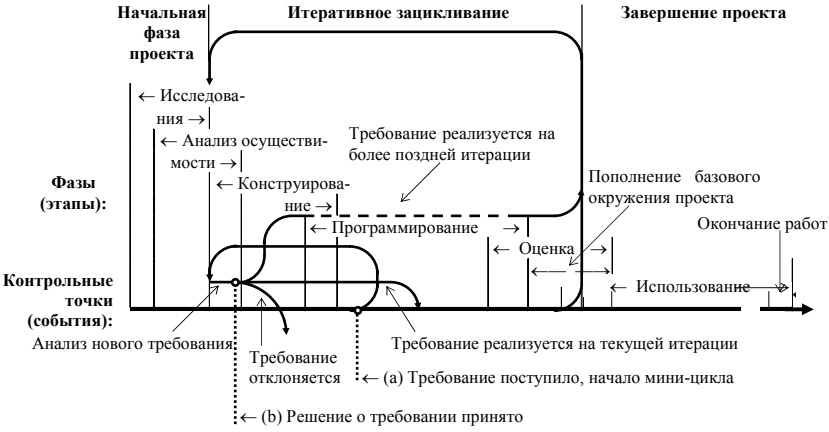


Рис. 15. Фазовое измерение модели жизненного цикла при объектно-ориентированном развитии проекта, дополненное обработкой требования в мини-цикле

Понятно, что в этой модели отобразить поток требований, поступающих при развитии проекта, невозможно (по этой причине на рисунке контрольные точки a и b выделены пунктиром). Постулируется, что все они обрабатываются в четыре этапа:

1) поступление требования или совместной группы (контрольная точка (a), которая может появиться в любой момент этапов конструирования, программирования или оценки);
2) расщепление, переход к анализу;
3) принятие решения (контрольная точка (b) на общем участке этапов анализа и конструирования);
4) планирование срока или будущей итерации реализации.

3.4. Особенности первой итерации

Модель жизненного цикла с мини-циклами обработки требований вполне адекватно описывает процесс поставленной разработки проекта программного обеспечения в его стационарный период, т.е. когда начальная итерация развития проекта уже пройдена. Однако она не учитывает тот факт, что первая итерация всегда является особой. При ее выполнении закладываются основы сопутствующих проекту системы типов требований, глоссария и других составляющих поддержки процесса трансформации требований, которые в стационарный период используются.

Первую итерацию обычно характеризует следующее.

- Разработчики еще не достигли достаточно глубокого понимания проблем предметной области, ее приоритетов и критериев.
- Круг инициаторов работ, а значит, потенциальных консультантов сформировался далеко не окончательно. Следовательно, есть опасность начать делать не ту систему.
- Мало информации о том, достаточно ли полон набор требований для объективного принятия проектных решений. Приходится работать на уровне гипотез (важное следствие предыдущих тезисов).
- Еще не сформированы базовые элементы декомпозиции системы, которые должны стать точками последующего итеративного роста. Они являются первыми, а значит, пробными для реализации компонентами.
- Если команда разработчиков формируется для данного проекта, то расстановка кадров может быть далеко не оптимальной.
- Часто разработчики в начале проекта не вполне владеют методами, инструментами и т.п., как следствие, работа на первой итерации имеет учебный аспект.

Можно выделить и другие особенности первой итерации, которые обусловлены тем, что она задает направление развития проекта на все время будущей жизни программы. Неудачен выбор направления — осуществимость продуктивного итеративного наращивания возможностей программной системы сомнительна. Удачный выбор — это минимизация затрат на последующие переделки, реальная возможность использования принципа итеративного наращивания, облегчение решения задачи отслеживания связей и др.

Для первой итерации с ее ближайшей проектной задачей роль этапов анализа и конструирования очень высока. Высока и цена ошибочных решений. Осознание этого приводит к разработке специальных методов и под-

ходов, которые целесообразно применять на первой итерации, а точнее, когда велика степень неопределенности выбора. Эти методы не только не исключают, но и предполагают переделку проектных решений, переписывание программного кода и т.д., т.е. отчасти нарушают основные каноны объектно-ориентированного проектирования.

Отклонением от канонов на первой итерации следует признать и возврат к традиционному принципу проектирования, постулированному для последовательно развиваемых программных проектов: не приступать к программированию, пока *все требования к системе* не будут переработаны в ее спецификации. Разница лишь в трактовке слов «все требования к системе». Для первой итерации при объектно-ориентированном проектировании они означают предварительное накопление необходимого и достаточного *базового набора требований*, который позволяет обеспечить надежную основу дальнейшего итеративного наращивания возможностей. Именно этот набор определяет первую ближайшую задачу, решаемую в начале развития проекта, с точки зрения архитектуры системы в целом.

Большинство требований на уровне анализа выражается в виде сценариев, которые надо реализовывать на данной итерации. Это в полной мере относится и к первой итерации. Но здесь *исходный комплект сценариев* играет две дополнительные роли:

- он рассматривается как один из способов изучения прикладной области. Разработчики согласуют реализуемые сценарии с инициаторами работ и, тем самым, уточняют свое понимание задач проекта, назначение системы;
- являясь аналитическим выражением базового набора требований, исходный комплект должен представлять этот набор *репрезентативно*, т.е. так, чтобы обеспечивать надежное развитие проекта.

Очень важна еще одна особенность первой итерации: к оценке ее результатов нельзя подходить с позиций утилитарной полезности получаемых программных продуктов. Как следствие, смещаются критерии качества: на первый план выступают задачи демонстрации осуществимости проекта, продуктивности выбранного подхода и полноты базового набора требований с точки зрения итеративного наращивания. Иными словами, по указанным выше причинам трудно ожидать, что первая итерация приведет к реально работоспособному программному изделию, но правомерно требовать от нее доказательства того, что данная команда, используя данный метод в конкретных условиях, в дальнейшем приведет проект к успешным результатам. Это мнение должно сложиться у заказчиков, руководства и, что не менее важно, у работников в коллективе исполнителей.

Большую часть особенностей первой итерации выразить в модели жизненного цикла не представляется возможным. Они влияют на выбор методов ведения проектов на первой итерации. В свою очередь, эти методы можно проецировать на модели жизненного цикла. В качестве примера одной из таких моделей ниже приводится схема, описывающая организацию начальных работ, которая принята в центре объектно-ориентированных технологий фирмы IBM. Данный метод получил название «Сначала в глубину», что соответствует выбранной стратегии.

Суть метода состоит в том, что разработка первой итерации проводится мини-циклами реализации выбираемых сценариев. Используется два критерия отбора сценариев для мини-циклов:

- реализацию можно осуществить быстро;
- получаемые результаты можно продемонстрировать наглядно и убедительно.

Полнота базового набора требований в методе «Сначала в глубину» достигается за счет анализа последовательно выполняемых мини-циклов для выделенных сценариев. Если в какой-то момент обнаруживается, что для полноты необходимо расширение исходного комплекта сценариев, то комплект пополняется.

На рис. 16 представлена еще одна модификация гантеровской модели жизненного цикла, отражающая развитие работ на первой итерации методом «Сначала в глубину». Модель модифицирована в следующих отношениях.

- По сравнению со стационарным периодом время, отводимое для анализа и конструирования, существенно увеличивается за счет этапа программирования (конкретные временные соотношения зависят от особенностей выполняемого проекта и от условий его выполнения).
- На общей части этапов анализа и конструирования (контрольные точки 2, 3) выделяется явно работа по определению и утверждению базового набора требований и сценариев для реализации (группа сценариев обозначена овалом с внутренними незакрашенными кружками). Таким образом, появляется контрольная точка 2′ [6].
- Этапы анализа и конструирования для выбранных сценариев выполняются совместно (жирная стрелка между овалами). В результате

[6] Чтобы не нарушалась нумерация контрольных точек, принятая для прежних моделей, дополнительные контрольные точки указаны номерами, помеченными символом апострофа. Расшифровка прежних контрольных точек приводится только для тех событий, которые меняют свое содержание.

строятся модели сценариев (контрольная точка 3′, модели обозначены закрашенными кружками), которые рассматриваются в качестве исходных данных для спецификации реализуемых компонентов (контрольная точка 5).

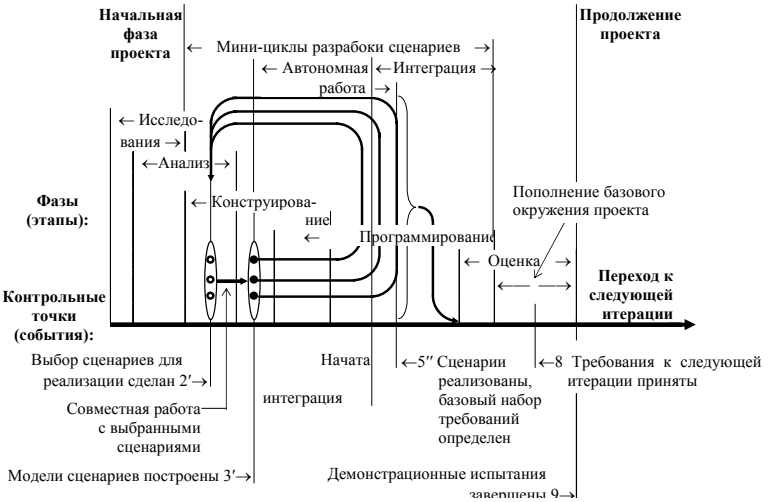


Рис. 16. Фазовое измерение модели жизненного цикла при объектно-ориентированном развитии проекта методом «Сначала в глубину»

- Для каждого из сценариев образуется мини-цикл его разработки. Разработка мини-циклов включает в себя перекрывающиеся этапы автономной работы и интеграции сценариев (контрольные точки 3′, 5″ и 5′, 7).
- Фиксируется событие готовности результатов всех мини-циклов (контрольная точка 5″), которое означает завершение формирования базового набора требований.
- Интеграция сценариев предполагает ревизию (возможно и переписывание кода, и даже перепроектирование реализации каких-либо из сценариев). Она должна быть закончена к началу этапа пополнения базового окружения проекта в рамках оценки (контрольная точка 7).
- Вместо подготовки к распространению системы для прикладного использования проводятся демонстрационные испытания, после завер-

шения которых (контрольная точка 9) осуществляется переход к следующей итерации.

- Прерывания процесса выполнения первой итерации для обработки дополнительных требований не допускаются. По существу это означает, что остается единственный вариант работы с такими требованиями: откладывание их анализа и реализации до последующих итераций.

Организация работ методом «Сначала в глубину» не предполагает предварительной подготовки элементов системы поддержки процесса разработки, которые зависят от прикладной области. Инструментальная часть этой системы вполне может быть универсальной, в частности, когда разработчики выполняют совместно не первый проект, это, скорее всего, так и будет, но информационная часть системы всегда определяется областью применения программного изделия. В ходе первой итерации к моменту выбора сценариев для реализации должна быть составлена предварительная, гипотетическая система типов требований, которая меняется под воздействием сведений, получаемых при выполнении мини-циклов, а также при интеграции сценариев. Итоговая система типов требований есть один из результатов этапа пополнения базового окружения проекта. Ситуация с глоссарием аналогична, с той лишь разницей, что нет необходимости до этого этапа фиксировать гипотетические положения о прикладной области, о составляемых моделях и т.п. Предварительно (до начала мини-циклов разрабоки сценариев) в глоссарий следует включить сведения о стратегии разработки, соглашения о технологических регламентах, т.е. все то, что носит универсальный характер.

Из приведенной модели видно, что метод «Сначала в глубину» дает вполне приемлемое представление о том, как происходит объектно-ориентированное проектирование. Разработчики могут рассматривать мини-циклы в качестве прототипа итеративного наращивания, а поскольку каждый из мини-циклов обозрим, к концу первой итерации достигается решение задачи, о которой шла речь выше: доказательство того, что данная команда, используя данный метод в конкретных условиях, в дальнейшем приведет проект к успешным результатам.

3.5. Фаза завершения

Завершение проекта редко описывают в моделях жизненного цикла структурно. Обычно этот период только обозначается, а все его работы лишь классифицируются. Возможно, что разнообразие вариантов организации эксплуатационной поддержки препятствуют систематическому их изу-

чению. Понятно, что не стимулирует изучение этих работ неявное и не соответствующее действительности положение о том, что требования к системе, возникающие на фазе завершения, относятся уже к другому проекту. В то же время, промышленная разработка программных систем всегда нуждается в организации как можно более скорых откликов на пользовательские запросы: рекламации, пожелания и требования развития.

В контексте изучения жизненного цикла с точки зрения обработки требований задачу моделирования фазы завершения можно описать в стиле, который был использован при учете трассировки требований. Не стоит ожидать от этого описания, что оно даст единственно возможный рецепт работы. Цели моделирования иные. Во-первых, это систематизация действий, которые необходимо выполнять в качестве реакции на пользовательские запросы. Во-вторых, это явное разграничение реакций, относящихся к текущей версии, к одной из следующих версий, к другому проекту. Для развития проектов в объектно-ориентированном стиле такое разграничение очень важно из-за необходимости осуществлять одновременную поддержку разных версий в сочетании с итеративным наращиванием возможностей системы.

Основой моделирования фазы завершения проекта (итерации) является обратная связь с пользователями системы. Это обычные сообщения о ходе эксплуатации: мнения, рекламации, пожелания, нарекания, претензии и т.п. Все подобные сообщения могут быть классифицированы, ранжированы по степени важности для развиваемого (на данной фазе — обслуживаемого) проекта. Но это обстоятельство в модели не учитывается: считается, что из пользовательских сообщений *извлекаются требования* к проекту в целом или к его итерации. Сообщения, а значит, и требования могут поступать в ходе эксплуатации в течение всего периода использования системы или ее версии. Требования нуждаются в трассировке, о которой шла речь выше. По этой причине в качестве отправного момента моделирования фазы завершения проекта (итерации) служит модель жизненного цикла, учитывающая трассировку.

В представленной на рис. 17 модели описываются операционные маршруты, возникающие в связи с обработкой одного требования в ходе эксплуатации программной системы. Для упрощения предполагается, что операционные маршруты не зависят от накапливаемой информации. Это заведомо огрубляющее предположение в том смысле, что принятие решений о требовании фактически делается не только на основании первичных установок, но и с использованием знаний о системе, пользователях, о текущем представлении о приоритетах и предпочтениях. Можно считать, что подоб-

ные сведения используются в точках разветвления операционных маршрутов, но то, как осуществляется такое использование, моделью не описывается.

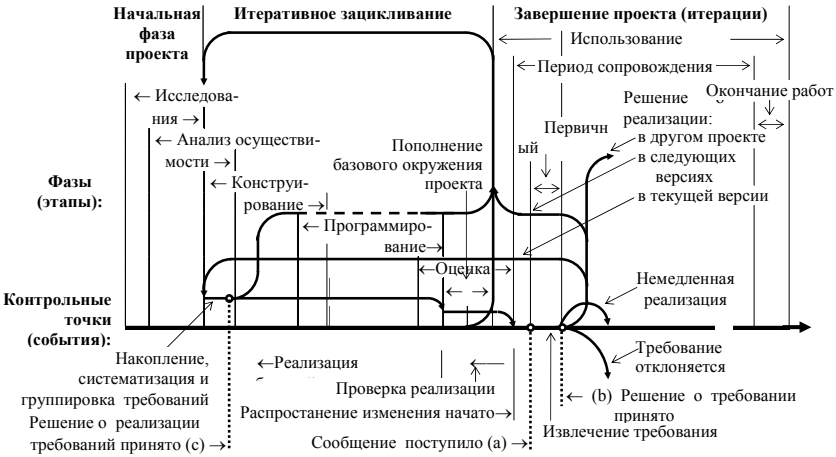


Рис. 17. Модель обработки требований в период эксплуатации системы

Аналогично тому, как это было при организации мини-цикла для трассировки требований, в модели периода эксплуатации началом обработки является *поступление сообщения* о ходе эксплуатации системы, которое можно трактовать как содержащее требования (контрольная точка (a)). Это событие может возникать в любой момент периода сопровождения, т.е. обсуждаемая модель является естественным продолжением модели с обработкой требования в мини-цикле (см. п. 3.3). Так как отобразить весь поток сообщений невозможно, рассматривается операционный маршрут только одного сообщения, при этом постулируется, что все сообщения обрабатываются следующим образом:

- *Поступление сообщения* (контрольная точка (a)).
- *Первичный анализ*, в ходе которого из сообщения извлекаются требования. Этот период должен быть максимально кратким, ибо пользователю необходимо знать о реакции разработчиков на его сообщение. Результатом первичного анализа является *принятие решения о требовании* (контрольная точка (b), в которой происходит расщепление ли-

нии жизненного цикла). Возможны следующие не взаимоисключающие друг друга варианты такого решения:

- *немедленная реакция* — действия, направленные на быстрое устранение замечания, если это возможно, либо указание пользователю сроков, когда и как будут учтены поступившие претензии, либо указание пути преодоления трудностей (возможно, временные). Немедленная реакция выполняется всегда, в том числе совместно с другими решениями;
- *требование отклоняется* — действия, указывающие пользователю причины отклонения требований и пути преодоления трудностей;
- *реализация требования в текущей версии* — если претензии обоснованы, а устранение замечаний, ошибок и т.п. возможно в рамках обслуживаемой версии, то организуется мини-цикл обработки требований в итерации;
- *реализация требования в одной из следующих версий* — если устранение замечаний в рамках обслуживаемой версии невозможно или нецелесообразно, то требование передается для исполнения на одной из следующих итераций проекта;
- *реализация требования в другом проекте* — если выясняется, что в данном проекте требование реализовать невозможно или нецелесообразно, то, быть может, оно станет одним из аргументов в пользу организации нового проекта.

- *Мини-цикл обработки требования* начинается с *анализа*, цели которого обычны для объектно-ориентированного развития проектов. В частности, определяется осуществимость реализации на данной итерации или целесообразность переноса ее на другую итерацию, образуется группа требований, которые должны быть реализованы совместно. Выработка этих решений о стратегии реализации требований приурочивается к контрольной точке (c). Особенностью анализа в данном случае является то, что он проводится, как возобновленный процесс, так как основные работы итерации уже выполнены.
- *Реализация* отобранных *требований* на данной итерации осуществляется по обычной схеме, включающей конструирование и программирование и оценку. В качестве специфики следует указать на особую роль проверочных работ — дополнительный этап *проверки реализации*, который вкладывается в этап оценки. Эти работы обязательно должны включать повторение проверки того, что было отлажено ранее. Таким образом, пополнение базового окружения проекта приоб-

ретает дополнительное содержание: накопление тестовой базы проекта.

- *Распространение изменений* (контрольная точка 10) — деятельность, направленная на то, чтобы сделанные исправления стали доступны для всех пользователей обслуживаемой версии. При массовом использовании программного изделия эта работа может потребовать значительных ресурсов.

Фаза завершения итерации включает этап окончания работ, содержание которого сводится к сворачиванию деятельности с данной версией программного изделия. Предварительное оповещение о наступлении этапа (контрольная точка 11) важно для пользователей, чтобы они смогли либо перестроиться, либо найти аргументы (в частности, ресурсы) в пользу продолжения сопровождения изделия. Как показывает практика, чтобы не потерять своих пользователей, очень часто приходится продолжать поддерживать весьма старые разработки, вкладывая в это солидные средства.

ЗАКЛЮЧЕНИЕ

Обсуждение жизненного цикла представлено в настоящей работе как последовательное развитие и уточнение понятий под влиянием потребностей развивающихся методов и технологий программирования. Однако из этого не должно складываться впечатление, что столь же прямолинейна историческая линия развития представления о том, какие этапы и как проходятся в течение жизни программы. Напротив, начиная с семидесятых годов XX столетия, когда сформировалась потребность в изучении жизненных циклов, и до наших дней варианты их моделей все множатся и множатся. Причина тому — особенности проектов, требующие учета и организационно-технологической поддержки. В качестве иллюстрации этого тезиса далее упоминаются лишь некоторые особенности, которые нашли свое отражение в реальных моделях жизненного цикла:

- совместная разработка программного обеспечения и оборудования (общего или специального) назначения,
- разработка программного обеспечения для встроенных систем,
- разработка программного обеспечения по уже существующему прототипу,
- разработка, включающая быстрое предварительное построение прототипа,
- построение сетевых комплексов,

- решение задач переиспользования программного обеспечения,
- учет требований повышенной надежности разрабатываемых программно-аппаратных систем,
- разработка адаптивных систем,
- разработка систем с настраиваемым интерфейсом,
- конструирование программных инструментов,
- использование технологических сред для разработки программных систем (различные варианты моделей).

Этот список может быть продолжен. Так, обнаруживаются и отражаются в моделях особенности жизненного цикла долго и быстро живущих программ, длительных, средних и коротких проектов. Как показывает анализ моделей, предлагаемых для разных ситуаций, они лишь уточняют и дополняют общие положения, отслеживанию которых было уделено внимание в предыдущих разделах.

Среди всех мотивов моделирования жизненного цикла особое место занимает систематизация работ, выполняемых при разработке программного обеспечения. Систематизация — первый шаг на пути автоматизации любого производства, и в частности, производства программ. Следующие шаги — определение технологических маршрутов деятельности работников данного производства, выявление узких мест, доступных для автоматизации, и разработка инструментов для них. Далее процесс развивается вширь и вглубь: охватываются автоматизацией другие части технологических маршрутов, совершенствуются ранее построенные инструменты, формируются методы их эффективного применения. Последнее означает формирование новых технологий, и, как следствие, появляется потребность автоматизации новых видов деятельности, обусловленных данными технологиями. Наконец, наступает момент, когда совокупность потребностей в автоматизации разного рода деятельности, связанных, хотя и не обязательно напрямую, с первоначальной систематизацией, формирует качественно иную потребность в комплексной автоматизации. Это время появления стандартов и стандартных решений, интеграции сложившихся технологий и доработка того, что не вписывается в интегральную схему.

В предыдущем абзаце представлен эскиз формирования произвольного автоматизированного технологичного производства. Не является исключением и процесс технологизации производства программного обеспечения. Есть, конечно, специфика, но она не столь значительна, чтобы считать данное производство чем-то исключительным.

Первая стадия автоматизации программирования связана с *поддержкой этапа программирования* жизненного цикла. Здесь проявляется и специфи-

ка: систематизация работ по производству программного обеспечения осуществлялась после осознания того, что поддержка кодирования, хотя и способствует росту производительности труда, но не является достаточным для промышленного конструирования программ. Появление на этой стадии систем программирования и всевозможных средств помощи в наборе текстов предшествовало периоду, когда начинали внедряться разного рода отладочные средства. Именно в это время (т.е. лишь к концу шестидесятых годов) в ответ на потребность в разработке больших и сложных программ было осознано понятие жизненного цикла.

Сразу же обнаружилось узкое место программирования как производства: неразвитость методологий этапа конструирования, с одной стороны, а с другой — невозможность сведения оценочных работ к тестированию. По существу это две стороны одной медали: нечеткость постановок задач на программирование влечет за собой большую часть трудностей этапа проверки. В результате сформулированной потребности в строгих спецификациях проекта появилось осознание того, что *этап конструирования может быть регламентирован вполне технологично*. И удачные организационные технологии стали появляться. Чаще это специализированные технологии, предназначенные для разработки программ особого рода, но иногда и общие технологии, некоторые из них впоследствии стали автоматизированными (в качестве конкретного примера из этого ряда уместно указать на IDEF-технологию). Позднее стали формироваться регламентирующие методики работы с требованиями на этапе анализа. И хотя формализованных технологических процедур, допускающих полные автоматические проверки, в аналитической части добиться так и не удалось (что свидетельствует об объективной трудности данной области), прогресс заметен: сегодня можно говорить о поддержке накопления первичных требований и их систематизации, об отслеживании связей между требованиями и их реализациями в проекте и др.

Понятно, что описанный процесс не столь прямолинеен, как он представлен выше. В огромной мере на него влияли языкотворчество и тщетные надежды на то, что появление очередного «самого хорошего» языка приведет к «решению всех проблем». Для становления технологий производственного программирования наиболее заметными оказались методология структурного программирования и объектно-ориентированное программирование. Первая из них позволила осознать ограниченность способностей человека, необходимых при составлении больших программ, вторая — дала толчок к разработке методов декомпозиции, приспособленных для преодоления сложности. Объектно-ориентированное программирование привело к

необходимости модернизации основополагающих принципов проектирования программ и, в частности, к новому понятию жизненного цикла (см. разд. 2 и 3).

Но, несмотря на это и другие влияния, стадия комплексной автоматизации технологий программирования стала возможной только при соответствующем уровне развития техники, который позволил эффективно применять выразительные графические возможности при выполнении технологических процедур конструирования программного обеспечения. Немаловажным обстоятельством, позволившим перейти к комплексной автоматизации, стало осознание того, что нельзя говорить реально о промышленном программировании без поддержки технологических функций *на всех этапах* жизни программ. Примерно в начале девяностых годов появился термин — CASE-технология (Computer Added Software Engineering — компьютерная поддержка разработки программ), которым стали обозначать использование систем, обладающих комплексными автоматизированными средствами поддержки разработки и сопровождения программ.

Замечено, что, впрочем, вполне объяснимо, что наиболее удачным оказалось использование CASE-систем в тех специальных областях, в которых уже были успехи и опыт технологичной практической работы, пусть даже лишь на организационном уровне, а также в тех случаях, когда специальная область уже была обеспечена надежной теоретической базой. В первую очередь здесь следует упомянуть о CASE-системах разработки баз данных в развитых реляционных СУБД (к примеру, Oracle Designer 2000 в системе Oracle). Успехи CASE-систем общего назначения скромнее, скорее всего по причине отсутствия универсальных методов, пригодных для развития любых проектов. Поскольку представление о модели жизненного цикла всегда является основой технологии, это еще раз подтверждает правомерность построения разнообразных моделей.

Сегодня универсальные CASE-системы строятся из расчета не всеобщего назначения, а в рамках применения развитых, но все-таки специальных методологий. Несомненный прогресс в данной сфере достигнут для проектирования, ориентированного на моделирование на этапах анализа и конструирования. В рамках объектно-ориентированного подхода разработан специальный унифицированный язык моделирования UML (Unified Modeling Language), который рассматривается как основа проектирования в методологии итеративного наращивания возможностей объектно-ориентированных программных систем. На базе этого языка построен ряд CASE-систем общего назначения с весьма развитыми средствами. Наиболее заметной из них является Ration Rose фирмы Ration Software, предло-

жившей на рынок не только инструментарий для использования UML, но и комплексную методологию производства систем — Ration Unified Processing (RUP). Данная методология претендует на охват всех аспектов технологий современного программирования. Не вдаваясь в детали того, насколько эти претензии обоснованы, уместно отметить, что в качестве CASE-системы Ration Rose обладает множеством средств, очень полезных для поддержки связи первых этапов проектирования с этапом составления программ (кодирования), а также с этапом оценки. В частности, проверяется, что моделирование на разных этапах согласовано, что модельные соглашения, определения классов, других элементов моделей и их взаимосвязи непротиворечивы. Уровень автоматического анализа высок настолько, что позволяет строить по моделям так называемые реализации по умолчанию. Это заготовки программного кода, включающие в себя описания классов и их методов в том виде, который можно извлечь из моделей. Программист дополняет заготовки фрагментами, детализирующими конкретную реализацию.

Построение реализации по умолчанию — не нововведение Ration Rose. До этой системы оно активно применялось и в рамках систем визуального программирования, и еще раньше в специализированных CASE-системах, используемых, например, в развитых СУБД. Последнее примечательно: именно для СУБД удалось связать реализацию по умолчанию с графическими моделями информационных систем (ER-диаграммы). В Ration Rose и других UML CASE-системах поддерживается построение реализаций по умолчанию по моделям общего, а не специального назначения.

Реализация по умолчанию является лишь одним из приемов поддержки связей между этапами жизненного цикла разработки программного обеспечения с использованием Ration Rose. Именно идея комплексной поддержки связанности рабочих продуктов разных этапов, а не отдельные приемы, которые появлялись и ранее, — главное для данной CASE-системы. Программное воплощение этой идеи, пусть даже с существенными недоработками следует отнести к явным достоинствам данного инструментария. Что же касается претензий RUP на охват «всех рациональных технологий», то в ней больше рекламы, чем фактического результата. Делается попытка механического объединения средств, инструментов и методов довольно многих «рациональных» подходов, но это приводит к эклектике, а для пользователя — к нефиксированной технологии, что по сути своей означает одно — отсутствие технологии. Применяя данную систему, пользователь обязан выстроить свои регламенты: когда, как и в каком качестве будут применяться те или иные средства, методы, инструменты. Если эти регламенты

окажутся технологичными, то можно рассчитывать на поддержку Ration Rose, но, к сожалению, не в части проверки принимаемых для формируемой технологии соглашений.

Претензии на всеобщность RUP не следует рассматривать как предложение универсальной технологии, которая, по-видимому, недостижима. Но собрание рациональных приемов и методов да еще с основательной автоматизированной поддержкой само по себе представляет ценность: возможно построение с его помощью конкретных технологий, ориентированных на наиболее подходящие для них представления о жизненном цикле. Будут ли определены вследствие таких построений стандарты и стандартные решения в области промышленного производства программного обеспечения, не столь существенно по сравнению с пользой от автоматизации для решения конкретных задач.

Вопросы, которые затрагивались в настоящей работе, освещены в многочисленных публикациях, посвященных технологии программирования. Не все из них выдержали испытание временем. В этой связи в качестве приятного исключения можно указать на работу Ф. Брукса «The Mythical Man-Month. Essay on Software Engineering» (русский перевод первого издания, вышедшего в 1975г., см. в [1], юбилейного издания 1995г. — в [2]). Эта монография по праву считается одной из лучших книг не только по данной тематике, но и по программированию вообще. Сопоставление двух ее изданий явно показывает, что проблемы, которые приходится решать при управлении программными проектами, почти не изменились со времени перфокарт. Меняется только техническая поддержка.

Из ранних работ, не потерявших своей актуальности, прежде всего следует обратить внимание на монографию Гантера [3], содержащую, кроме представленной выше модели, много полезной информации для организации работ над программными проектами. Систематизированные сведения о понятии жизненного цикла и его применении в промышленном программировании можно найти в книге [4], которая, к тому же, дает представление о состоянии дел в этой области в СССР к началу восьмидесятых годов. Весьма обстоятельное исследование задач и методов проектирования и разработки программного обеспечения выполнено Боэмом. Его книга [5] постоянно цитируется и в наши дни.

Современное представление о технологии проектирования программных систем прочно связано с методологией объектно-ориентированного программирования. Всестороннее изложение данного подхода, его концепций, а также общих методов разработки проектов в объектно-ориентированном стиле можно найти в книге Буча [6]. UML и методы его

использования в практической разработке программных проектов хорошо изложены авторами этого языка в монографии [7]. Понятия, связанные с CASE-технологиями, достаточно четко излагаются в работах [8, 9]. В частности, в последней из упомянутых публикаций достаточно подробно освещаются вопросы CASE-технологий, связанных с проектированием информационных систем.

Следующие ссылки помогут получить сведения об упомянутых выше конкретных разработках. Книга [10] дает наиболее полное представление о СУБД Oracle, в частности, об Oracle Designer 2000 и его месте в системе. IDEF-технология хорошо представлена в документе [11]. Информацию о RUP в целом и Ration Rose в частности можно найти на сайте [12].

СПИСОК ЛИТЕРАТУРЫ

1. Брукс Ф.П. Как проектируются и создаются программные комплексы. — М.: Мир, 1979.
2. Брукс Ф.П. Мифический человеко-месяц, или как создаются программные системы. — СПб: Символ-Плюс, 1999.
3. Гантер Р. Методы управления проектированием программного обеспечения. — М.: Мир, 1981.
4. Липаев В.В. и др. Технология проектирования комплексов программ АСУ. — М.: Радио и связь, 1983.
5. Боэм Б.У. Инженерное проектирование программного обеспечения. — М.: Радио и связь, 1985.
6. Буч Г. Объектно-ориентированное проектирование с примерами применения. — М.: Конкорд, 1992.
7. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя. — М.: ДМК, 2000.
8. Новоженов Ю.В. и др. Объектно-ориентированные CASE-средства // СУБД. — 1966, № 5-6. С. 119-125
9. Вендров А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. — М.: Финансы и статистика, 1998.
10. Ричардс и др. Oracle ® 7.3. Энциклопедия пользователя. — Киев: «Диа-Софт», 1997.
11. IEEE IDEF0. “Standard Users Manual for the ICAM Function Modeling Method — IDEF0”. — IEEE draft standard P1320.1.1. 1997.
12. Rational Unified Process. — Copyright © 2001 Rational Software Corporation. http://www.rational.com/products/rup/index.jsp

Л. В. Городняя, О. Н. Очаковская

ДИНАМИКА ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Рассматривая программы и программные системы, как формы представления знаний, трудно удержаться от попытки исследования **динамики представления знаний** на основе аналогии с развитием программ и программных систем.

Движущими силами этого развития являются: необходимость разных видов **эффективной** деятельности, потребность в **уточнении** представления знаний и установление новой информации, которая раньше не попадала в поле зрения или наблюдатель не был готов ее понять. Динамика представления знаний сводится к переходу от одного представления к другому.

Успешность эффективной деятельности ограничена «пропускной способностью» поля зрения. Это ограничение систематически преодолевается посредством обобщения, приводящего к представлениям более высокого порядка — представлениям более мощным, более организованным, например, к процедурам, функциям, фреймам, шаблонам, макросам. Последовательность шагов обобщения можно называть **индуктивным развитием** представления знаний. В методике программирования индуктивное развитие соответствует восходящим методам «снизу вверх». Переход систем искусственного интеллекта от семантических сетей к системам фреймов также пример индуктивного развития. Как правило, индуктивное развитие имеет некоторые пределы. Такие пределы при возрастании **меры информативности** используемых средств рассматриваются Д. Скоттом [1]. Интересен случай, когда пределом является теория, достаточная для порождения всей достоверной информации, установленной к текущему моменту времени. При разработке программ в качестве такого предела используется система программирования.

В результате индуктивного развития представления знаний наблюдается тенденция к возрастанию доли средств **декларативного** характера (таких как описания, отношения, формирователи, типы, фреймы, семантические сети, иерархии понятий, аксиоматические системы) в сравнении с долей средств **процедурного** характера (таких как действия, операции, операторы, процедуры, интерпретаторы, задания). Эта тенденция обуславливает рост эффективности применения дедуктивных методов и может рассматриваться как стимул к переходу от индуктивного к **дедуктивному развитию**. Дедуктивный вывод осуществляет переход от потенциальных знаний к ак-

туальным. Традиционно для этих целей в системах искусственного интеллекта используется метод резолюций, системы продукций и другие средства. **Чередование стадий** индуктивного и дедуктивного развития можно рассматривать как обоснование выбора метода программирования в зависимости от уровня развития знаний о решаемой задаче (зрелость, **уровень изученности**).

Применение развиваемых таким образом представлений может потребовать возврата к менее структурированным средствам (например, для упрощения обратной связи с областью, породившей решаемые задачи). Такой переход является **конкретизацией** представления знаний. В методике программирования конкретизация соответствует нисходящим методам «сверху вниз».

Независимо осуществляемое развитие приводит к задаче установления **эквивалентности** между различными системами представления знаний. При решении этой задачи возникают предпосылки для целенаправленного дедуктивного развития, что приводит к выравниванию потенциала систем (вводятся недостающие понятия, выполняются аналогичные построения, реализуются подобные инструменты). Таким образом, выделено три типа переходов: индуктивное и дедуктивное развитие и конкретизация. Эта классификация соответствует классификации трансформаций программ в теории смешанных вычислений, предложенной А.П. Ершовым [2]. Приведенные наблюдения находят аналогии на более широком материале.

СПИСОК ЛИТЕРАТУРЫ

1. Скотт Д. Теория решеток, типы данных и семантика // Данные в языках программирования. — М.: Мир, 1982. — С.25–53.
2. Ершов А.П. Смешанные вычисления: потенциальные применения и проблемы исследования // Тез. докладов и сообщений / Всесоюзная конф. «Методы математической логики в проблемах искусственного интеллекта и систематическое программирование», Ч. 2, Вильнюс, 1980. — С.26–55.

Л. В. Городняя, Ф. А. Мурзин

ПСИХОЛОГИЯ ПРОГРАММИРОВАНИЯ

Несколько лет назад И.В. Поттосин с большим энтузиазмом поддержал идею спецкурса «Психология программирования» на матфаке НГУ. По его мнению, такой курс совершенно необходим при профессиональной специализации программистов. Теперь накоплен опыт преподавания в этом направлении, и материал курса рассматривается как направление исследования и разработки учебных систем информатики.

1. ВВЕДЕНИЕ

Психологические и социальные аспекты процессов разработки и применения информационных систем, по мнению авторов, нашли свое отражение в эволюции языков и систем программирования, что позволяет рассматривать последние как удобный материал для исследования формализуемых моделей психологической информатики.

Человеческие факторы во многом определяют практику разработки и совершенствования информационных систем, а также влияют на трудоемкость проектов и жизнеспособность производственных команд. Поэтому целесообразно проанализировать проблемы формирования процесса профессиональной информатики как социальной деятельности, изучая личностные факторы, проблему лидерства, жизненный цикл программисткой команды и особенности прохождения производственного проекта через кризисные полосы. Обзор результатов исследований психологии программирования как проявления индивидуальности, взаимосвязи интеллектуальности и способности к принятию решений представляет несомненный интерес.

2. СОЦИАЛЬНЫЙ КОНТЕКСТ

Не вызывает сомнения связь производительности труда в информатике с механизмами поисковой активности и внимания, их видами и нарушениями. Экспериментальная база исследования призвана проявить конкретику таких связей. Она может формироваться как система мероприятий по предпроизводственной подготовке информатиков, включая молодежные проекты и конкурсы. Это позволяет в сжатые сроки подвергать практиче-

ской проверке формальные методы исследования связей в коллективах, оценивая особенности творческих процессов, осваивая технику переговоров в конфликтах и методы ведения коллективного проекта. Полезно найти подходы и навыки классификации производственного микроклимата и прогнозирования трудоемкости проектов в зависимости от квалификации персонала.

В центре внимания — психологические (человеческие) факторы, отраженные в процессе разработки и совершенствования информационных систем, а также влияющие на трудоемкость проектов и жизнеспособность производственных команд. (Психологическая информатика — ПИ.) В процессе исследования ПИ можно анализировать проблемы формирования профессиональной информатики как социальной деятельности, зависящей от личностных факторов, лидерства, жизненного ритма команды и особенностей прохождения производственного проекта через кризисные полосы. Особо интересны психология программирования как проявление индивидуальности, взаимосвязи интеллектуальности и способности к принятию решений, а также другие аспекты этико-социального контекста информатики.

ПИ граничит с соционикой и психофизиологией умственного труда, со стрессовыми и другими реакциями человека, влияющими на связь производительности труда в информатике с механизмами поисковой активности и внимания.

3. ПРИЛОЖЕНИЕ И ПЕРСПЕКТИВЫ

Исследование ПИ может дать рекомендации для разработки и совершенствования системы подготовки бакалавров и магистров, специализирующихся в области информационного и электронного бизнеса. Высокий темп развития информатики требует исследования глубинных психологических и социальных факторов, определяющих закономерности эволюции конструктивных идей информатики, смену парадигм программирования и других форм концентрации знания человеком, а также этико-социальный контекст современных технологий.

Основные акценты исследования человеческих факторов в процессах совершенствования и применения информационных технологий заключаются в анализе гуманитарных аспектов отечественной информатики и выработке на их основе рекомендаций по организации проектов, обладающих

исследовательскими аспектами, обычно сопутствующими деятельности в области информационного бизнеса.

4. СПЕЦИФИКА ПОДХОДА

Задача исследования ПИ в первую очередь нацелена на психологические и этико-социальные аспекты современных технологий и человеческой деятельности, выявленные при изложении и исследовании отечественных достижений информатики непосредственно их авторами, сумевшими объективно отразить роль личностных и субъективных факторов, определивших основные достижения известных научных школ в России и за рубежом. Все это формирует систему представления знаний по гуманитарным аспектам компьютерных наук, начиная с анализа разных явлений, связанных с восприятием информации и особенностями работы с текстами и изображениями. Ключевые вопросы следующие. Как отличить хорошую информационную систему? Откуда берутся хорошие информационные системы? Как обосновывается роль квалифицированных специалистов в рамках специально разработанных технологий?

Важные моменты любых технологий: ясность требований к квалификации специалистов, адекватность используемых средств, спецификация проектов систем, четкость графика разработки, определенность методов проверки качества результата, приспособленность проекта к возможности улучшения реализуемых систем и пр.

Сквозные вопросы следующие. Как исследовать гуманитарные аспекты применения и производства информационных систем? Можно ли опереться на самоанализ информатиков, наблюдения менеджеров и научную проверку известных гипотез? Богатый фактический материал дает экспериментальная, творческая, досуговая и учебная информатика. Большой интерес представляют психологические и поведенческие оценки, накопленные в опыте учебных заведений, производственных организаций и научно-исследовательских проектов по разработке информационных систем.

5. МЕТОДЫ И ПРИОРИТЕТЫ

Используются социологические методы для исследования производственных команд, связанных общей постановкой задачи. Анализируется влияние целей на основные аспекты рабочего ритма команды. Рассматриваются кризисы в производственном проекте с учетом проблемы лидерства.

Статистически исследуется длительность проектов и время пребывания в них исполнителей, а также измеримые характеристики информационных систем.

Системный подход используется для выбора приемов обеспечения стабильности путем изменений с учетом структуры проекта в контексте общих проблем больших проектов. Оценивается достоверность отчетов и результативность контроля за ходом работ, компетентность исполнителей и руководителей.

Комплексный подход позволяет унифицировать жизненный цикл информационной системы. Рассматриваются компьютерные игры и учебные тренажеры в условиях эволюции систем и обновления технологий. Отслеживается уровень изученности задач, решаемых информатиками при программировании. Важные факторы — общность и переносимость компонентов, приспособленных для повторного использования. Определяется направленность развития систем и принципы обработки уникальной информации. Формируются прототипы «идеальных» систем. Сравниваются консервативные и открытые системы.

6. УРОВЕНЬ ИЗУЧЕНОСТИ И ОБРАЗОВАНИЕ

Современные представления о социально-психологических факторах в информатике отражают преимущественно зарубежные достижения, значимость которых очевидна неспециалистам благодаря высокому качеству элементной базы.

Современное состояние исследований по проблемам ПИ представлено множеством интересных для практики результатов, слабо отраженных в отечественной литературе. Основные направления исследований по данной теме в мировой науке активно развиваются, они нацелены на решение проблемы стабильного производства информационных систем, обладающих длительным жизненным циклом или допускающих целенаправленные улучшения. Такие исследования, по существу, опираются на человеческий фактор.

Выявлены значительные трудности в преподавании студентам проблем поддержки полного жизненного цикла информационных систем, таких как определение требований, разработка спецификаций, организация сопровождения. Трудности обусловлены недостатком практического опыта у студентов. Следует отметить, что многие наши студенты теперь вовлечены в производство. Это помогает преодолеть трудности преподавания ПИ в условиях совмещения учебы с производственной деятельностью при

ловиях совмещения учебы с производственной деятельностью при использовании средств дистанционного образования. Результаты исследований ПИ используются фирмами, ведущими сертификацию специалистов и кадровый маркетинг.

Целесообразно развернуть проект, который выполнял бы работу по упорядочению и изучению гуманитарных факторов информатической деятельности, проявившихся в истории отечественной вычислительной техники и ее программного обеспечения. Труды Д.А. Поспелова, Я.И. Фета, И.В. Поттосина, собравших фактические материалы по истории отечественного программирования и школы А.П. Ершова, а также личный опыт участников проекта, рассматриваются как фактическая основа для выработки практических рекомендаций.

СПИСОК ЛИТЕРАТУРЫ

1. Городняя Л.В., Мурзин Ф.А. Об опыте преподавания спецкурса «Психология программирования» // Тез. IX Междунар. конф. «Применение новых технологий в образовании», Троицк, 1998. — С. 101–102.
2. Кирпотина Н.А. Модель процесса принятия решений и соответствующая классификация типов личности. // Там же. — С. 93–95.
3. Городняя Л.В., Мурзин Ф.А. Психология для программистов // Тр. IV Междунар. конф. «Перспективы систем информатики». Секция «Школьная информатика». — Новосибирск, 2001. — С. 29–30.
4. Андреева Т.А. О проблеме накопления результатов методических разработок школьных педагогов. // Тез. конф. «Информационные технологии в образовании» Москва. — Институт ЮНЕСКО, 1998. — С. 124–128
5. Городняя Л.В. Быть или не быть — для информатики вопрос не в этом // М.: Информатика и образование. — 2000. — Т. 7. — С. 80–84
6. Городняя Л.В., Кирпотина И.А. Электронное издание как механизм самообучения // Тр. Междунар. научно-практической конф. «Новые информационные технологии в университетском образовании». — Томск, 2000. — С. 101–102
7. Андреева Т.А. Об автоматизации подготовки и проведения заочных (электронных) олимпиад // Тр. научно-практической телеконф. «Информационные технологии в общеобразовательной школе», ноябрь 2000. — Новосибирск, 2000. — `http://www.edu.nsu.ru/ites`
8. Ким Н.А., Городняя Л.В. Измерители знаний по информатике // Тр. Междунар. научно-практической конф. «Новые информационные технологии в университетском образовании». — Новосибирск, 1999. — С. 97.

9. Берс А.А., Городняя Л.В., Поляков В.Г., Чурина Т.Г. Организация творческой работы учащихся в контексте общеобразовательной информатики // Там же. — С. 86.
10. Kalinina N., Kostyukova N. The Basic Principles of Building of Ergonomic Component of Automated Training System // Internat. J. of Occupational Safety and Ergonomics (JOSE). — Poland, 2001. — P. 203–207.
11. Городняя Л.В., Калинина Н.А., Костюкова Н.И. О соотношении кибернетики, математики и психологии // Тр. XXVIII Междунар. конф. «Информационные технологии в науке, образовании, телекоммуникациях и бизнесе» (IT+SE'2001). — Гурзуф, 2001. — С. 113–116.
12. Городняя Л.В. Откуда берутся хорошие программисты // Становление новосибирской школы информатики. Мозаика воспоминаний / Под ред. И.В. Поттосина. — Новосибирск, 2001. — С. 117–123.

Т. С. Васючкова

СТАНОВЛЕНИЕ ЭЛЕМЕНТОВ ПРОМЫШЛЕННОЙ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ В ПРОЕКТЕ СОЗДАНИЯ ОПТИМИЗИРУЮЩЕГО ТРАНСЛЯТОРА АЛЬФА-6 (1968–1972 ГОДЫ)

Проект по разработке оптимизирующего транслятора с языка АЛЬФА-6 на ЭВМ БЭСМ-6 был начат в январе 1968 г. Как позже признавались его руководители — акад. А.П. Ершов, д.ф.-м.н. И.В. Поттосин, Г.И. Кожухин — это была довольно авантюрная идея: сделать транслятор силами студентов механико-математического факультета Новосибирского государственного университета. Задача сложная и объемная — конечный объем системы составил 150 000 строк. Стартовый опыт студентов в программировании почти нулевой. В качестве образца использовался уже существующий транслятор АЛЬФА. Однако результат оказался успешным. Созданный транслятор эксплуатировался в ряде организаций для решения научных и прикладных задач. Одним из основных его потребителей стал Институт Атомной Энергии им. И.В. Курчатова (Москва).

В современных программных проектах применяются развитые инструментальные среды, методологии и технологии разработки программ. В 1968 г. всего этого не было. Тем интереснее посмотреть, как в условиях «полукустарной» работы команда проекта эмпирически приходила к удобным и достаточно эффективным технологическим решениям.

1. При согласовании рабочих вопросов возникло сразу несколько проблем. Во-первых, десять исполнителей совершенно нерационально использовали время на бесчисленные споры, обсуждения, заседания. В необходимости обсуждений сомнений не было. Но каждый выходил на заседания со своими проблемами и ждал их немедленного решения. Результативность таких собраний была невысокой. И первый урок менеджмента, полученный нами, был тот, что заседания надо тщательно готовить и приходить на них с конкретными вариантами решений. Во-вторых, локальные договоренности между двумятремя участниками проекта не всегда своевременно доводились до остальных исполнителей. Это могло приводить к ситуации, когда член команды, считавший, что он успешно продвигается вперед, вынужденно переделывал работу. В-третьих, решения по текущим про-

блемам часто оформлялись на бумаге в виде черновых записок, хранились рассредоточено и могли затеряться. Мы довольно быстро отреагировали на эти трудности. Решили, что всю текущую документацию надо хранить централизованно и поддерживать в актуальном состоянии. А инструментом единой коммуникативной среды сделали некое подобие «бортового журнала», назвав его «Книга жалоб и предложений». Это была амбарная книга, в которую каждый мог занести в письменном виде все проблемы, новые идеи, объявления об изменениях, заявки на общие заседания и прочее. Каждый член команды в обязательном порядке ежедневно читал эту книгу и учитывал ее при планировании своей работы. Координатором проекта был один из студентов. Он отвечал за организацию решения проблем, внесенных в «Книгу жалоб и предложений», определял сроки и график работ, готовил рабочие заседания, контролировал исполнение работ. Такая единая среда общения участников проекта была организована в «бумажном виде», а не в электронном — ведь персональных компьютеров еще не было! Она оказалась очень удобной и полезной — помогала оперативно решать вопросы, позволяла видеть прогресс работ и болевые точки, давала общую картину проекта на сегодняшний день. В современных технологиях разработки программ есть разные средства и инструменты, которые уже в электронном виде поддерживают идею централизованной коммуникативной среды участников программного проекта.

2. По предложению руководителей проекта была определена отдельная ролевая функция разработки единых стандартов представления информации, которой обмениваются блоки транслятора. В терминах нынешних технологий это называется — внутренние интерфейсы программной системы. В технологическом аспекте важным уроком для нас стало то, что создание внутренних интерфейсов должно предшествовать реализации отдельных частей системы. Сейчас это звучит совсем тривиально, но в те времена мы на собственных синяках и шишках хорошо усвоили, что отсутствие полной договоренности о внешних связях может повлечь многократные переделки твоей работы.
3. В рамках проекта была создана независимая служба тестирования. Ее задачами стали разработка и развитие единой системы тестов, а также разработка системы автоматизации тестирования. К тому времени как раз вышли классические работы Г. Майерса по надежности

и тестированию программ, идеи которых мы использовали в своем проекте. На собственных трудностях и ошибках мы поняли, что

(а) тестирование автором своей части программы является лишь начальным, «черновым» этапом перед систематической проверкой программы;

(б) тестирование — самостоятельная дисциплина со своими методами, стратегиями, инструментами;

(в) принципиально, что служба тестирования независима, так как тестовик — это враг программы (по Майерсу). Автор программы, составляя тесты, подсознательно пытается доказать, что у него все правильно. Тестовик же находится на иной психологической позиции: он хочет показать, что программа неправильна и составляет другие, «разрушительные» тесты;

(г) все перечисленное актуально для серьезных коллективных проектов. Создавая небольшую программу, автор сам исполняет роль тестовика.

4. К 1971 г. транслятор АЛЬФА-6 успешно использовался в нескольких организациях. От Института Атомной Энергии им. Курчатова поступило пожелание: дополнить транслятор новыми возможностями — развитым текстовым редактором, архивной подсистемой, хорошим отладчиком, диалоговой подсистемой запуска заданий. Теперь это называется «оболочкой транслятора». К тому времени примерно половина исполнителей ушла из проекта, защитив диплом и окончив НГУ. Оставшиеся готовились к завершению работ и передавали систему службе сопровождения.

А. П. Ершов выступил с предложением создать оболочку транслятора в очень короткий срок (3 месяца), используя новую идею Х. Миллза ролевого разделения работ в проекте. Первоначально автор назвал этот подход методом хирургической бригады, сейчас его чаще называют методом главного программиста. Традиционный подход нами уже был использован — транслятор делился на подзадачи, и каждый исполнитель полностью отвечал за свою подзадачу, выполняя роли проектировщика, кодировщика, тестовика. В новом методе работ все было непривычно и интересно — любая подзадача решалась целой бригадой, каждый член которой выполнял только отдельную функцию (роль) или несколько ролей. Четверо участников проекта с большим энтузиазмом решили на практике испытать метод Миллза — образовали бригаду и разделили между собой 10 ролей.

Завершив работу за 3 месяца (объем 28000 строк), мы оценили высокую пользу от разделения труда в программном проекте. Средняя производительность труда при разработке транслятора традиционным методом составила 5000 строк/чел-год, а при разработке оболочки транслятора методом Миллза составила 28000 строк/чел-год. Конечно, было бы неправильно сравнивать эти данные напрямую — ведь за 3 года существенно выросла наша квалификация, да и задачи различаются сложностью. Но по нашимобсуждениям и впечатлениям метод Миллза может увеличить производительность труда программиста в 2–3 раза, если удачно выбрана кандидатура главного программиста.

На своем опыте мы также убедились, как резко повышается ответственность за результат, когда ты исполняешь только отдельную роль и в напряженном темпе работ сильно зависишь от других участников, а они от тебя. Следует признаться, что метод Миллза мы применяли немного в искаженном виде. Формально роль основного оппонента лидера команды, а именно, роль «второго пилота», должна принадлежать одному человеку. Фактически каждый из членов команды с азартом и задором критиковал главного программиста и высказывал свои идеи. Однако, на этом наша свобода кончалась — окончательные решения принимал team leader.

Удивительно, что результативность разделения труда в программировании нам удалось увидеть в рамках одного проекта АЛЬФА-6. На первой стадии работ эффективным стало выделение ролей координатора, тестовика, разработчика внутренних интерфейсов. На второй стадии проекта мы увидели, что выгодно специализировать в качестве ролей все функции разработчиков.

В заключение интересно отметить, что время проекта АЛЬФА-6 пришлось на знаменитый «кризис программирования» в 70-х гг. Тогда резко подешевевшие компьютеры были востребованы для решения не только вычислительных задач. Спрос общества на компьютерную автоматизацию деятельности в самых разных сферах резко возрос, а профессиональные программисты обнаружили, что их собственный труд автоматизирован минимально, что разработка программ не может быть названа промышленной деятельностью.

Известно, что во время этого кризиса из-за неумения программистов организовать свой труд каждый четвертый проект кончался неудачей. Это исторический взгляд в прошлое с высот сегодняшнего дня. Тогда же мы находились внутри ситуации, не знали и не думали ни о каком кризисе. Наши руководители сами были молоды — им не было и сорока. Их тех-

нологические решения были нацелены на преодоление повседневных трудностей проекта.

Только теперь мы можем оценить, насколько эти решения были конструктивными и «попадали в цель», как в сложных проектах вызревали идеи и подходы, которые впоследствии сформировали сегодняшние технологии промышленного программирования.

ОСНОВНЫЕ ПУБЛИКАЦИИ ПО ПРОЕКТУ

1. Кожухина С.К., Козловский С.Э. Входной язык АЛЬФА-6 / Под ред. И.В.Поттосина. — Новосибирск, ВЦ СО АН СССР, 1976. — 109 с.
2. Аникеева И.Н., Буда А.О., Васючкова Т.С., Кожухина С.К., Козловский С.Э., Шелехов В. И. Руководство к пользованию системой автоматизации программирования АЛЬФА-6 / Под ред. А. П. Ершова. — М.: Наука, 1979. — 351 с.

ПЕРСОНАЛИИ

Руководители проекта АЛЬФА-6: А.П. Ершов, Г.И. Кожухин, И.В. Поттосин.

Исполнители: И.Н. Аникеева, С.Ф. Богданова, А.О. Буда, Т.С. Васючкова, В. Грибачевская, А.А. Грановский, А.А. Ерофеев, Л. Жукова, С.К. Кожухина, С.Э. Козловский, В. Коростелева, К. Макаров, Е. Мельникова, Н.А. Сизова, Т.И. Шведова, В.И. Шелехов, Т. Янчук.

Л. В. Городняя, Н. А. Калинина

ИССЛЕДОВАНИЕ ВОПРОСОВ ПРЕПОДАВАНИЯ ФУНКЦИОНАЛЬНОГО ПРОГРАММИРОВАНИЯ И КОМПЬЮТЕРНОЙ АЛГЕБРЫ В УНИВЕРСИТЕТЕ

В работе исследуются вопросы изучения и преподавания компьютерных наук и программотехники. В центре внимания — функциональное программирование и компьютерная алгебра. Они рассмотрены в качестве важных составляющих современного научного инструментария. Исследования в этой области начались при поддержке И.В. Поттосина, проявлявшего к ним живейший интерес.

По мнению авторов функциональное программирование наиболее естественным образом отражает сущность процесса обработки информации. Это достигается посредством сведения программы к базовым структурам. Полнота решений классов задач обеспечивается обобщением используемых средств до функций высших порядков.

Системы компьютерной алгебры составляют основу формирования естественного выражения математических свойств объекта. Они вскоре будут занимать свое достойное место в преподавании любых математических дисциплин. Поэтому знакомство с ними целесообразно при изучении фундаментальных основ современной науки в университетах.

В статье описывается опыт решения вышеописанных вопросов в учебных курсах кафедры программирования механико-математического факультета Новосибирского Государственного Университета.

Изложен опыт изучения и применения систем компьютерной алгебры. Представлен опыт преподавания функционального программирования. Эти курсы нацелены на овладение инструментарием научного работника. Кроме того, отражен опыт курса парадигм программирования в ВКИ НГУ. Этот курс ориентирован на подготовку профессиональных инженеров-информатиков.

1. ВВЕДЕНИЕ

Научный работник естественно сталкивается с достаточно сложными явлениями. Их восприятие требует навыков свободного абстрагирования, причем абстрагирования более высокого уровня, чем это дают стандартные языки программирования. Дело в том, что последние вынуждено и традиционно подчинены трем одновременно решаемым требованиям:

- представление и формализация решения задачи,
- вписывание реализации этого решения в количественные ограничения предполагаемой компьютерной обстановки,
- достижение приемлемой результативности при использовании решения.

Трудозатраты на удовлетворение второго требования устарели. Они не оправданы при современных характеристиках производительности оборудования. Приемлемая результативность решения многих исследуемых задач обеспечивается современными инструментами автоматически. Решение многих классов задач не требует специальных мер по преодолению стеснительных количественных ограничений. Навыки аккуратного распределения памяти во многих видах работы формируются как дань традиции. Успех вычислительного эксперимента не так уж сильно страдает от недостаточно оперативной реакции программных средств.

Многие классы задач требуют гибкого сочетания исследуемой и реальной моделей. Наиболее последовательно и универсально такое сочетание представляется в терминах символьной обработки информации. Функциональное программирование можно рассматривать как инструмент символьной обработки информации. Таким образом, оно способствуют расширению сферы научных исследований, основанных на вычислительном эксперименте.

Функциональная форма программы вычислительного эксперимента наиболее естественным образом позволяет рассматривать, моделировать и применять научные законы. Она отражает законы восприятия информации. На ее основе реализуется эффективная техника совершенствования знаний в форме алгоритмов. Движение в этом направлении достаточно плодотворно. Оно реально подкреплено общим прогрессом компьютерных технологий.

Последующее изложение следует такой схеме. Сначала мы рассматриваем общие проблемы изучения в университете компьютерных наук и программотехники. Затем приведен анализ основных особенностей функционального стиля программирования. Обращаем внимание, что мы переживаем бурный период многократной смены и улучшения технической компьютерной базы. В результате теория и практика программирования стали двигаться почти независимыми путями. Это тормозит развитие и того, и другого. Функциональное программирование — естественная грань, где взаимопонимание теоретиков и практиков сравнительно достижимо.

Компьютерная алгебра — достаточно удобный математический аппарат. Он создает естественную среду для научного работника, предпочитающего

алгебраические системы. Рассматриваются вопросы преподавания компьютерной алгебры как инструмента научного исследования, использования систем компьютерной алгебры (СКА) в общих математических курсах, а также вопросы, связанные с курсами специализации.

В заключении рассмотрена схема проникновения методов функционального программирования как в средства, способствующие применению компьютерной алгебры, так и в средства их реализации. Это подтверждается и исторически. Многие ранние системы компьютерной алгебры были реализованы на языках функционального программирования. Практически большинство систем компьютерной алгебры, такие как MACSYMA, REDUCE, реализованы на Лиспе. Некоторые реализованы на языке функционального программирования Рефал и др.

2. ПРОБЛЕМЫ ОБУЧЕНИЯ ИНФОРМАТИКЕ И ПРОГРАММОТЕХНИКЕ

Дефицит технических средств оказал сильное влияние на традиции преподавания алгоритмики и программирования. Успешность решения минимального спектра учебных задач зависела от умения учитывать традиционные схемы реализации информационных систем. Были крайне важны навыки эффективной организации процессов информационной обработки. (Они и сейчас важны в производстве.) Без аккуратного распределения памяти редкий эксперимент давал результат. Все это оправдывало учебное применение стандартных языков программирования.

Современное техническое оснащение учебных компьютерных классов допускает полноценное проведение вычислительных экспериментов и без стандартных языков программирования. Обеспечено исследование вычислительных и алгоритмических построений в предельно естественной форме. Причем, в форме, приближенной к общепринятой математической символике. Важно, что такая символика свободна от диктата стандартных реализационных стереотипов.

Это ставит на повестку дня рассмотрение средств и систем функционального программирования и компьютерной алгебры в качестве базовых информационных технологий. Они достаточны для освоения и изучения студентами алгоритмических и вычислительных методов. Они поддерживают и многие другие методы научных исследований.

Такие технологии образуют надежную и устойчивую инструментальную основу учебного процесса. Смягчается чувствительность учебного процесса к смене компьютерной базы. Повышается эффективность труда

научного работника в области постановки вычислительного эксперимента. Упрощается и воспроизведение эксперимента на разных компьютерных платформах.

На сегодняшний день наметилось расширение исследований в области преподавания информатики. Особенно это относится к обучению программотехнике. Важен по-прежнему акцент на наукоемкие аспекты разработки информационных систем. Существенно обострилась проблема производственного программирования для мощных компьютеров. Нерешенность проблем обучения проявляется на каждой стадии процесса разработки программ, таких как разработка требований, спецификаций, документации и создание систем тестов, организация тестирования, конструирование программ по прототипам, измерение качества программ [5,6].

Многие авторы [7] подчеркивают, что университетское образование — это основной путь передачи знаний из общества исследователей в промышленность и правительственные структуры. Следовательно, актуально при обучении разработке требований к информационной системе — обеспечить изучение оснований компьютерной науки. Специалисты должны владеть навыками демонстрации непредсказуемых неясностей и противоречий в решаемых задачах. Особенно в задачах, связанных с проблемами реальной жизни. В этом плане целесообразно до университета будущим студентам дать представление о следующих направлениях:

- компьютерные наука и инженерия,
- информационные системы и наука,
- промышленная инженерия и управление.

Попытки прямого обучения программотехнике не слишком успешны. Они увлекательны, но учебные групповые проекты малорезультативны. Мешает недостаточность личного опыта студентов. Им трудно понять сложность взаимодействия в проекте, различать требования и проект. Они не склонны работать по нисходящей технологии программирования. Поэтому специфику требований лучше изучать попозже. Необходимо личное участие хотя бы в одной достаточно сложной разработке.

Но такой метод подготовки студентов в области компьютерных наук требует большей ясности в отношении целей обучения. Потенциальное улучшение университетского образования потребует новой философии обучения, что повлечет пересмотр состава изучаемых тем. Следовательно, меняется комплект учебной литературы. Меняется набор обучающих инструментов. Требуется система учебных проектов. Нужна методика, обеспечивающая реальный перенос получаемых знаний в практику.

Последнее требует методичного подхода. В него входит включение во вспомогательные материалы к учебным проектам показательных результатов анализа области приложения. Такими результатами должна иллюстрироваться динамика развития спецификации требований к разрабатываемому продукту:

- следует представить принципы развития спецификаций по мере изменения условий применения продукта;
- необходимо уяснить важность изучения вариантов допустимых решений;
- должна быть видна роль проектной работы;
- полезно добиться понятности методов привлечения реальных клиентов.

Все это превращает каждый проект в часть слишком сложного курса. Он должен интегрировать полный жизненный цикл программного обеспечения. (Именно такая полнота обеспечивала успех механизма Новосибирских школ юных программистов!)

Возможно, такая организация учебных проектов помогает студентам понять важнейшие требования производства информационных систем:

- принципиальную открытость программного продукта,
- интегрирующее влияние жизненного цикла программного обеспечения,
- соединенность разных средств через обратные связи,
- эволюционную природу человеческих требований,
- необходимость изменений на разных фазах проекта.

Из этого следует, что жизненный цикл программы должен проектироваться:

- он подразумевает изменения спецификации,
- неизбежна множественность источников информации,
- принципиально отсутствие исчерпывающего источника,
- необходимо обеспечивать отслеживаемость процесса разработки,
- непрерывно возникает необходимость удостоверять соответствие промежуточного результата текущим требованиям,
- нужно фиксировать отношение требований к источникам.

Все это не укладывается в стандартный учебный регламент. Нужен определенный уровень знаний и зрелости. Они достигаются лишь при столкновении с реальными проблемами. Напрашивается вывод, что полноценное обучение программотехнике в университетском регламенте затруднительно. (Подразумевается при подготовке математика-исследователя.) [8].

Удается лишь помочь наиболее заинтересованным студентам научиться применять научное знание в инженерно-конструкторской деятельности. Такое знание предстает в области информационных технологий как культура систематических изменений. Это пробуждает производственный спрос на обучение разработке градуируемых информационных систем [9].

Естественный метод формирования культуры систематических изменений — практика деловых игр. Попытка проведения деловых игр по разработке программ обладает большим сходством с идеями Новосибирских школ юных программистов [17]. В сферу игры были вовлечены следующие виды работ:

- чтение и изменение чужого кода,
- обеспечение интеграции программных средств, сделанных разными командами,
- уяснение важности установки общего стандарта для обеспечения совместимости независимо получаемых средств,
- оценка роли документации в установке программных инструментов,
- выявление источников трудностей работы в команде.

По ходу игры выявились неприятные явления:

- студентов тяготит обреченность на провал в конкуренции с производственными средствами;
- большинство не в силах преодолеть привычку устного общения (мешает документальной регистрации решений);
- акцент на отчеты часто доминирует над качеством проекта;
- поощрение критицизма участников приводит иногда к провалу проекта (полезно дискутировать об улучшениях проекта, но трудно воздержаться от его немедленной ревизии);
- многие не успевают в срок получить результат, пригодный для демонстрации.

При выборе учебников были выдержаны следующие требования:

- желательно избежать как энциклопедии по программотехнике, так и ограниченного изложения отдельных методов;
- для начала необходима концентрация на низкоуровневом аппликативном программировании, но не отвергается переход к новым языкам программирования по мере понимания их основы;
- студенты получают подходящий прототип и установочные лекции (рекомендации по выбору необходимой техники исполнения заданий);
- имеется вводный курс по методам разработки программ.

Общее впечатление от результатов игры в целом благоприятное. Хотя не все получили пользу от курса (большинство просто наслаждалось такой формой учебы), многие уяснили сложность развития недокументированного кода. Они поняли, что информационная инженерия — это прежде всего искусство преодоления провалов как в оборудовании, так и в коллективе [10].

В целом сложившиеся в университетской практике подходы к образованию в этой области можно разложить на четыре линии:

- одновременно много концепций, несколько методов, чуть-чуть инструментов;
- концепции — отдельно, инструменты — отдельно;
- опережающее введение концепций;
- предоставление разных полезных инструментов.

По мнению педагогов, обилие инструментов отвлекает внимание студентов на синтаксис пользовательского интерфейса, что наносит ущерб постижению семантики изучаемых процессов информационной обработки. По мнению учеников, любой инструмент может пригодиться в будущей работе — он помогает при выполнении учебных работ или развлекает. Возможный выход видится в организации общей системы лабораторных работ. Подобный подход — практикум по физике. Он может сопровождаться лекционными курсами. Должны быть варианты по выбору [11].

Действительно, неопытным студентам трудно поверить в важность усложненной практики. Они не могут оценить, что синтаксис приложится сам в производственной деятельности. Кроме того, большая разработка не вписывается в университетское расписание, малая же — не дает полноценного опыта. Вот и сводится все к применению средств и методов решения изученных задач.

В качестве альтернативы желателен новый подход типа «погружения»:

- ранний импринтинг хороших навыков и практика,
- образование по полному жизненному циклу программ с обсуждением проблем в совместной работе,
- разработка по жесткому графику,
- сочетание ориентации на развиваемость с выбором варианта,
- учет различия в трудозатратах на изучение или решение задачи,
- последнее может поломать график,
- установление пропорций в уровне проработки частей проекта.

Такое погружение предполагает три фазы:

1) программирование в малом, т.е. для личного употребления,

2) информационные системы и программотехника — полный жизненный цикл, включая эксплуатацию,
3) защита проекта, проведение испытаний результата.

Желательно совместить активное обучение с пассивным восприятием.

Первое достигается через самостоятельное изготовление инструмента. Второе неявно происходит через изучение готовых прототипов, модифицируемых при изготовлении учебных программ.

Эксперимент по такой альтернативной схеме обучения программотехнике был проведен с привлечением молодых педагогов, любящих программирование. В каждой группе (около 12 человек) были распределены роли (руководитель, заказчик, технический консультант и т.д.) [12]. Предлагаемая схема близка к системе учебных проектов ВКИ НГУ [21], вводный проект которой можно рассматривать как пассивное восприятие прототипа изготавливаемой программы.

Новый слой трудностей вызывают парадигмы процесса совершенствования программного обеспечения. Некоторые авторы [13] отмечают, что многое здесь базируется на гипотезах вместо фактов. Нет общепринятых методик оценки практичности результата. Во что обойдется улучшение? Как долго оно дает выигрыш? В чем смысл улучшения (лучше качество, выше производительность, короче маршрут управления обработкой, полнее удовлетворение пользователя)?

Подытожим анализ причин затруднений в университетской подготовке по информатике и системному программированию [6]. Наиболее существенными признаются следующие причины.

1. Не существует общей теории, поддерживающей все аспекты развития программ и информационных систем.

2. Многие практики не знают существующих методов и, что более серьезно, не верят в их пользу.

Разработка программ не является малой частью компьютерных наук. Наоборот, части компьютерных наук могут, после определенной доводки, включаться в обязательное основание информационной инженерии [14]. Программотехника сейчас находится на уровне компьютерной науки 60-х годов. Это определяет приоритетные задачи научной деятельности в информатике на современном этапе:

- найти ясное и доходчивое изложение основ и достижений информатики и системного программирования;
- представить современное понимание проблем эволюции информационных систем и технологий;

- добиться достойной общественной оценки системной информатики.

При этом достаточно ограничиться информационными системами и инструментами, освоенными практиками. Не стоит распылять силы выходом за пределы инструментария, получившего массовое распространение. Иначе возникает риск остаться «за бортом». Самостийное развитие информационной и компьютерной инженерии вызовет к жизни свою компьютерную науку [15].

3. ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

С середины 80-х годов во многих университетах развитых стран функциональное программирование доминирует как основа лабораторного практикума. Это обеспечивает стыковку математических и компьютерных наук [18].

Такая возможность обусловлена взвешенной полнотой исходных идей функционального подхода. Языкам функционального программирования свойственна математическая выразительность. Они обладают концептуальностью и лаконизмом. На их основе достижима предельная общность построений. Реализациям функциональных языков свойственно богатство моделей исследуемых процессов. Они поддерживают традиционную для математики методику выделения базовых понятий. Это позволяет наиболее явно отражать природу решаемых задач, естественно, с точностью до мнемонично-лингвистических характеристик.

Язык программирования Лисп и сложившееся на его основе функциональное программирование [19] возникли достаточно давно. Они реально показали свои сильные стороны более чем убедительно. Особенно как инструментарий исследования и освоения новых областей применения вычислительной техники. Происходящее в настоящее время экстенсивное наращивание эксплуатационных характеристик оборудования многие приложения компьютерных наук превращает в исследования. Это аргумент для пересмотра преимуществ производственного императивного стиля программирования, избыточно чувствительного к особенностям аппаратных решений. Это и аргумент в пользу функционального подхода к решению задач. Такой подход позволяет снизить трудоемкость изучения и применения наукоемких методов обработки информации.

Именно функциональное программирование сумело преодолеть синтаксический разнобой независимо развиваемых средств и методов организа-

ции информационных процессов. Причем, в широком спектре приложений от искусственного интеллекта до автоматизации проектирования и медицинского обслуживания. Это удалось благодаря нацеленности на проявление и ортогонализацию семантических подсистем в организации исследуемых процессов. Не менее важна развиваемость представления унифицируемых структур данных и комплектов функциональных объектов. Все это позволяет языки функционального программирования рассматривать как средство сопряжения разнородных конструкций. На их основе возможно осуществлять любые интегрированные построения.

Выразительная сила функционального программирования позволяет добиться массовой изучаемости многих сложных задач, среди которых:

- исследование и разработка компиляторов [16];
- преобразование и оптимизация программ и процессов;
- любые эксперименты по управлению заданиями;
- лингвистическая обработка информации;
- построение изображений (не труднее, чем в Паскале, при этом преимущество — возможность анализа и конструирования программ прорисовки);
- гибкое управление базами данных и оптимизация информационных систем, вплоть до полного исключения потерь информации;
- моделирование общения

и многое другое.

Не исключено, что организация учебной практики по жизненному циклу программ на базе функционального программирования окажется вполне результативной. Она позволит студентам в более сжатые сроки накопить достаточный опыт. Это даст возможность сознательного приложения результатов компьютерных наук в практику разработки, развития и применения информационных систем. Это может существенно расширить сферу применения наиболее перспективных методов разработки надежных и эффективных программ.

Например, по нашему мнению, резко возрастает актуальность организации безопасной разработки программ. Безопасность требует доказательного конструирования и трансформационной техники модификации программ. Эксперимент по обучению такой технике работы может быть поддержан системой Френд [20]. Эта система обеспечивает применение базовых систем преобразований программ, представленных в виде Лисповских списков. В ней обеспечено построение своих специализированных преобразований. Возможно управление областями действия преобразований. Приме-

нение преобразований сопровождается протоколированием. Возможен откат на любую глубину.

Следует обратить внимание на эволюцию функционального программирования. К настоящему времени языки функционального программирования обогатились типовыми средствами практически всех известных подходов к представлению программ и организации вычислительного эксперимента и информационных процессов. В их числе средства для вычислений с повышенной точностью, обеспечена организация параллельных процессов. Возможна визуализация данных и программ. Имеются средства стандартного и объектно-ориентированного программирования. Поддержано управление компиляцией и конструирование компиляторов. Многие расхожие мнения относительно ресурсно-эксплуатационных трудностей функционального программирования давно утратили достоверность (если обладали ею!).

Обращаем внимание на свободно распространяемое программное обеспечение, доступное через Интернет. Оно содержит много систем семантической и визуальной обработки информации. Реализованы системы конструирования гипертекстов. Появились реализации, генерирующие код, совместимый с кодом Си-компиляторов. Реализация Common Lisp-а (GNU Clisp) ведет себя идентично на MS-DOS, Windows и UNIX, так как соответствующие версии продуцируются из одного программного кода. Существенно, что операции языка Лисп обладают машинно-независимой семантикой. Поэтому для Лисп-программ проблема переноса программ намного мягче, чем для стандартных языков, таких как Си, Паскаль, Фортран. Это делает функциональные языки программирования перспективными для образовательных применений. Особенно — на уровне средней школы. Успех Logo на этом поприще не случаен.

Студенты, изучавшие функциональное программирование на 4-м курсе ММФ НГУ, выражали сожаление, что их знакомство с функциональным программированием не состоялось раньше, когда была возможность попрактиковаться. Некоторые аспекты функционального программирования включены в программу курса «Парадигмы программирования» ВКИ НГУ [20]. Курс сопровождается учебными проектами и реферативными работами. Это позволяет развивать кругозор учащихся. Они получают возможность оценить исторический путь формирования идей информатики. Но этого не достаточно для формирования устойчивых навыков разработки и выработки стиля мышления. В сложившихся обстоятельствах от студентов требуется изрядная любознательность, активность, самостоятельность и заинтересованность.

4. ИСПОЛЬЗОВАНИЕ СИСТЕМ КОМПЬЮТЕРНОЙ АЛГЕБРЫ В УЧЕБНОМ ПРОЦЕССЕ

Сегодня аналитические вычисления на ЭВМ становятся современным инструментом многих исследователей. Параллельно с термином «аналитические вычисления на ЭВМ» более широко применяются эквивалентные ему термины «машинная алгебра» или «компьютерная алгебра». На сегодняшний день программы для аналитических вычислений от скромных программ, которые вначале выполняли только полиномиальные операции, превратились в сложные комплексы и системы [2]. Существует значительное число применений, использующих СКА, оказавшихся успешными только из-за того, что для решения поставленных задач в них использовались как аналитические, так и численные методы.

Появление развитых систем компьютерной алгебры требует значительной переориентации учебного процесса. Преподавание многих учебных курсов может стать значительно эффективнее, если активно использовать возможности автоматического проведения трудоемких алгебраических выкладок. Использование систем компьютерной алгебры позволяет выделить два направления, а именно: первое — применение СКА при изучении общих математических курсов и второе — использование СКА при подготовке студентов, специализирующихся в компьютерных науках.

В первом случае важно рассматривать использование СКА с учетом определенных предпосылок, позволяющих выработать внутреннюю готовность преподавателя ответить на группу таких вопросов, а именно: смогут ли компьютеры помочь студентам понимать математику; не будут ли механизмы вычисления, представляемые СКА, затушевывать математическое понимание предмета; может ли быть, что использование СКА будет ослаблять способность студента вычислять вручную; улучшает ли применение СКА обучение математике?

Во втором случае СКА рассматривается как система программирования с развитым входным языком, организация которой предполагает использование сложных структур данных и реализацию современных алгебраических алгоритмов. Разработка современных СКА во многом стимулировала и стимулирует появление новых алгебраических результатов.

Мы полагаем, что использование систем поддержки математических исследований, в широкий круг которых входят СКА, весьма желательно как при изучении общих математических курсов, так и при подготовке студентов, специализирующихся по информатике. В НГУ первая линия использования СКА поддерживается изучением базовых возможностей MAPLE в

общем вводном курсе «Основы работы на ЭВМ». Данная линия применения стремится упрочить использование СКА как естественную среду жизни математика, предоставляющую ему возможности проведения аналитических, численных и графических вычислений. Такой подход поддерживается заметным числом преподавателей НГУ как в основных, так и в специальных курсах.

Опыт Новосибирского государственного университета охватывает собой как использование СКА в качестве инструмента обучения для профессиональной подготовки программистов, так и использование СКА в общих математических курсах. На кафедре программирования несколько лет читается спецкурс «Системы и языки компьютерной алгебры». Он включает в себя рассмотрение архитектуры системы, организацию внутренних структур данных и базовых алгебраических алгоритмов СКА. В спецкурсе рассматривается опыт построения первых СКА, включая отечественные разработки, такие как АНАЛИТИК, АУМ [4] и другие, и наиболее значимые зарубежные системы компьютерной алгебры, такие как REDUCE, MAPLE и MATHEMATICA. Кроме того, в этом курсе рассматриваются методы применения, реализации и определения языков и систем компьютерной алгебры как ранние, так и сложившиеся в практике развития таких систем. Изучаются типовые конструкции языков компьютерной алгебры и особенности их реализационной семантики. Рассматриваются языки и структуры данных развитых систем компьютерной алгебры, таких как REDUCE, MAPLE, MACSYMA и ряда других. От студентов требуется общее представление о средствах и методах определения языков программирования, например, из курса методов трансляции или дискретной математики, а также знание более чем двух языков программирования.

Особый интерес представляет разработка систем компьютерной алгебры как метод обучения. Освоение методов СКА идет успешнее, если студент приобретает опыт участия в разработке СКА. Таким испытательным полигоном для студентов явилась разработка системы FABULA.

Система FABULA является инструментальной системой, ориентированной прежде всего на поддержку теоретических исследований по проблемам принятия решений в условиях интервальной и стохастической неопределенности. Основными функциями системы FABULA являются проведение аналитических преобразований булевых выражений в символьной форме и решение оптимизационных задач и систем уравнений в булевых алгебрах. Входной язык системы FABULA является развитым языком программирования. Он содержит средства как для задания выражений, функ-

ций, уравнений и других объектов языка, так и средства для выполнения преобразований над ними. Язык реализует историю хранения вычислений и удобную среду для выполнения преобразований. Язык имеет паскалеподобные конструкции управления и достаточно богатый набор средств манипулирования булевскими функциями. Конструирование системы осуществляется при помощи традиционных развитых методов разработки транслирующих систем, современных методов построения систем компьютерной алгебры и методов объектно-ориентированного программирования.

5. ЗАКЛЮЧЕНИЕ

В этой статье нам показалось своевременным отметить необходимость усилить акценты на преподавание функционального программирования и компьютерной алгебры в университетском учебном процессе. Данные дисциплины отражают естественную форму выражения математических форм объекта и, по нашему мнению, займут свою нишу как в преподавании математических так и компьютерных дисциплин. Это может послужить существенным противовесом избыточно прямолинейному стилю применения современных экстенсивно развивающихся информационных систем и восстановить навыки аналитических подходов к информатике.

Особо подчеркнем следующие выводы:

- полноценное обучение программотехнике в университетах требует практикума по полному жизненному циклу программ;
- функциональное программирование снижает трудозатраты на полный жизненный цикл программ по сравнению со стандартными языками программирования;
- системы компьютерной алгебры — удобный полигон для математического исследования многих классов задач, решение которых в функциональном стиле достаточно для получения студентами представительного опыта разработки программ.

Можно предложить следующую схему «погружения» студентов в проблемы разработки программ:

- применение СКА при решении задач по изучаемым в университете математическим дисциплинам;
- изучение функционального программирования на первом курсе, т.е. ранее стандартных языков программирования или одновременно с ними (не позднее!);

- практикум по моделированию СКА и усовершенствованию отдельных алгоритмов;
- включение лучших студенческих работ в университетскую библиотеку СКА.

СПИСОК ЛИТЕРАТУРЫ

1. Карпов В.Я., Карягин Д.А.,Самарский А.А. Принципы разработки пакетов прикладных программ для задач математической физики // ЖВТ и МФ. — 1978. — Т. 18, Вып. 2. — С.458–467.
2. Давенпорт Дж., Сирэ И., Турнье Э. Компьютерная алгебра. Системы и алгоритмы алгебраических вычислений. — М.: Мир, 1991. — 350 с.
3. Scott D.S. Symbolic Computation and Teaching // Lect. Notes in Comput. Sci. — 1996. — Vol. 1138. — P.1–20.
4. Калинина Н.А., Поттосин И.В., Семенов А.Л. Универсальная система проведения аналитических преобразований Аум. // Тр. совещ. по системам и методам аналитических вычислений на ЭВМ и их применению в теоретической физике. — Дубна: ОИЯИ, 1983. — Д11-83-511. — С.7–18.
5. Software Engineering Education Newsletter. — Software Engineering Technical Council, 1996. — Vol.14, No 5. — `http://www.tcse.org`, `http://www.computer.org`
6. Third International Workshop on Software Engineering Education. — Berlin, March, 1996.
7. Macaulay L., Mylopoulos J. Requirements Engineering: An Educational Dilemma // Software Engineering Education Newsletter. — Software Engineering Technical Council, 1996. — Vol.14, No 5. — P.21–22.
8. Mellor P. Software Engineering Education Workshop // Ibid. — P. 22–24.
9. Mead N. Preliminary Program: 1996 Conference on Software Engineering Education // Ibid. — P. 25–26.
10. Easterbrook S.M., Arvanitis Th. N. Preparing Students for Software Engineering // 3-d Intern. Workshop on Software Engineering Education. — Berlin, 1996. — P. 28–31.
11. Glinz M. The Teacher: Consepts! The Students: Tools! // Ibid. — P. 32-34.
12. Gorski J., Jurkowlaniec M. Experience with Introducing a New Method of Training in SE // Ibid. — P. 35–40.
13. Cappers J. The Pragmatics of Software Procass Improvements // Software Engineering Education Newsletter. — Software Engineering Technical Council, 1996. — Vol.14, No 5. — P. 13–16.
14. van Amstel J.J. We Do It Ourself // 3-d Intern. Workshop on Software Engineering Education. — Berlin, 1996. — P. 6–10.
15. Gibbs N.E. Software Engineering and Computer Science: the Impending Split? // Educatin&Computing. —1991. — N 7. — P. 111–117.

16. Rumbaugh J., Blaha M., Premerlani W., et al. Object-Oriented Modelling and Design. — Prentice-Hall, 1991.
17. Поттосин И.В. Программные системы информатики и работы Г.А.Звенигородского // Программные системы. — Новосибирск, 1995. — С. 9–16.
18. Functional Programming Languages in Education / Ed. by P. H. Hartel, M. J. Plasmeijer. — Springer-Verlag, 1995. — (Lect. Notes Comput. Sci.; 1022).
19. Сеппанен Й., Хьвенен Э. Мир Лиспа. — М.: Мир, 1990.
20. Городняя Л.В. Парадигмы программирования в профессиональной подготовке информатиков // Проблемы специализированного образования. — Новосибирск, 1998. — С. 115–124.
21. Городняя Л.В., В.Н.Касьянов. Подход к специализации по информатике и программированию в рамках системы непрерывного образования. — Новосибирск, 1995. — 59 с. — (Препр. / РАН. Сиб. отд-ние; № 23).

В.А. Евстигнеев

НАУКОМЕТРИЧЕСКИЕ ИССЛЕДОВАНИЯ В ИНФОРМАТИКЕ

Наука — это самоорганизующаяся система, развитие которой управляется ее информационными потоками, а наукометрия — это совокупность количественных методов изучения развития науки как информационного процесса.

Цель наукометрических исследований — дать объективную картину развития научного направления, оценить его актуальность, потенциальные возможности, законы формирования информационных потоков и распространения научных идей. Реализация этой цели включает в себя ряд конкретных задач, совокупное решение которых должно дать ответ на большинство поставленных вопросов.

Однако события последних десяти лет привело науку в России в плачевное состояние, при котором традиционные методы исследования вряд ли что-нибудь дадут. Эти методы рассчитаны на применение в условиях стабильности общества, а не во время перемен. Цель данной работы — дать обзор методов и результатов наукометрических исследований в области программирования, которые были проведены в середине 80-х гг. в Новосибирском филиале ИТМ и ВТ АН СССР под руководством автора.

1. ВВЕДЕНИЕ

В задачи науковедения входит, среди прочих, поиск оптимальной структуры научных учреждений и наиболее эффективных методов организации науки как объекта управления, а также определение скорости развития научных направлений, индивидуальной и коллективной продуктивности труда ученых и т.п. С этой целью в науковедении используются различные модели науки: экономическая, гносеологическая, логическая, информационная и др. В настоящих исследованиях мы будем исходить из информационной модели, так как феноменологически науку можно рассматривать как процесс получения существенно новой информации. Этот процесс носит последовательный и коллективный характер; всякая научная работа базируется на некотором множестве ранее выполненных работ. Наука — это самоорганизующаяся система, развитие которой управляется ее информационными потоками, а наукометрия — это совокупность количественных методов изучения развития науки как информационного процесса.

2. ЦЕЛИ НАУКОМЕТРИИ

Цель наукометрических исследований — дать объективную картину развития научного направления, оценить его актуальность, потенциальные возможности, законы формирования информационных потоков и распространения научных идей.

Реализация этой цели включает в себя решение ряда конкретных задач (см. ниже), совокупное решение которых должно дать ответ на большинство поставленных вопросов. Достаточно подробное наукометрическое исследование такого научного направления как программирование (в основном системное и теоретическое программирование) было проведено в середине 80-х годов в Новосибирском филиале ИТМ и ВТ небольшим коллективом в составе Ф.Р. Енгибарова, В.Н. Киселева, О.А. Логиновой, Н.И. Снытниковой, Л.И. Булыгиной, Л.И. Кульковой и др. Результаты их иследований содержались в ряде отчетов и библиографических указателях наукометрического типа.

3. МЕТОДЫ ИССЛЕДОВАНИЙ И ЗАДАЧИ В НАУКОМЕТРИИ

Известны следующие общие методы, используемые в наукометрии [1].

1. Статистический метод.
2. Метод подсчёта числа публикаций.
3. Метод «цитат-индекса».
4. Метод «контент-анализа».
5. Тезаурусный метод.
6. Сленговый метод.

Хотя все или практически все наукометрические методы имеют статистическую природу, тем не менее выделяется отдельный *статистический* метод, под которым понимается метод, использующий в качестве наукометрических индикаторов все измерители, кроме числа публикаций, ссылок и отдельных слов. В число измерителей науки, охватываемых статистическим методом, попадают такие измерители, как число учёных, журналов, заказов на годовые комплекты журналов в библиотеках и информационных центрах, открытий и другие, имеющие относительно крупные «единицы измерения», либо несоизмеримые с другими наукометрическими измерителями (например, энергия ускорителей элементарных частиц).

Измеритель (индикатор), определяющий метод подсчёта публикаций, — это число научных продуктов, под которыми здесь понимаются

книги, статьи, отчёты и т.д. Иногда при расчёте общего объёма научной продукции эти типы не различаются, иногда же книги, оригинальные статьи, обзорные статьи и прочее берутся с разными весами.

Наукометрический метод «цитат-индекс» базируется на обязательности ссылок на использованную литературу в научных публикациях; в основе его лежит наукометрический индикатор «число цитат или ссылок». В основе современных применений данного метода лежит индекс Гарфилда — регулярно выходящий начиная с 1964 г. «Индекс научных ссылок», издаваемый Институтом научной информации, возглавляемом Ю. Гарфилдом. Он состоит из нескольких частей. Основную часть Индекса составляет *указатель ссылок*, который позволяет установить, кто цитирует фиксированную работу данного автора. Фиксируется лишь фамилия первого автора работы, название работы не приводится, название журнала даётся в сокращенном виде. Название работы, фамилии и адреса всех авторов можно найти в *указателе источников*, представляющем вторую часть Индекса. Третья часть Индекса —*пермутационный предметный указатель* — позволяет найти авторов, в заголовках работ которых встречается данное слово. И наконец, последняя часть Индекса содержит результаты библиометрического анализа охватываемых Индексом цитирующих и цитируемых журналов.

Наукометрический метод «контент-анализ» происходит от социологического контент-анализа средств массовой коммуникации. Процедура контент-анализа заключается в сведении рассматриваемого текста к ограниченному набору определенных элементов (слов или, реже, предложений), которые затем подвергаются подсчёту и анализу.

В основе тезаурусного метода лежит одноимённый метод, широко применяемый в теории и практике информационного поиска. Упор в тезаурусном методе делается на содержательный анализ терминов для отбора среди публикаций текстов, релевантных запросу.

Недостатком этих двух методов является отсутствие *воспроизводимости* результатов подсчётов, достаточной для наукометрических исследований. Один из возможных путей увеличения воспроизводимости результатов — использование сленгового метода.

Сленговый метод, как и другие наукометрические методы, основывается на вероятностных представлениях и является статическим. Являясь результатом развития «контент-анализа» и тезаурусного методов, сленговый метод опирается, кроме того, на традиции статистической лингвистики, изучающей частотные распределения слов и создавшей частотные словари. В качестве индикатора в этом методе берётся не число «символов», как это

делается в контент-анализе, и не число «терминов» (ключевых слов), как это делается в тезаурусном методе, а просто число слов, точнее, словарных слов.

4. ЗАДАЧИ, РАССМАТРИВАЕМЫЕ В НАУКОМЕТРИИ

Наукометрический анализ включает в себя самые разнообразные задачи в зависимости от выбранного метода. Так, при использовании статистического метода решаются такие, например, задачи, как временные динамики числа открытий, числа журналов, числа учёных, динамика соавторства и т.д. Использование метода подсчета числа публикаций даёт возможность решать такие задачи, как распределения публикаций по странам, языкам, по типу издания и пр.

При проведении наукометрических исследовоаний мы остановились на ряде задач, которые не требовали больших затрат на их решение, с одной стороны, и давали бы достаточную картину развития программирования как научного направления. В частности, решались следующие задачи (без привязки к соответствующим методам):

- исследование динамики изменения числа научных публикаций по программированию;
- исследование структуры пространства научных журналов как каналов передачи научной информации;
- статистическое исследование языка научных публикаций (сленг-анализ);
- анализ библиографических ссылок в научных публикациях (цитат-анализ);
- исследование структуры и динамики развития незримых научных коллективов;
- исследование структуры научных связей ученого (научное окружение ученого).

Этот список не исчерпывает всех направлений исследований в наукометрии (в частности, совершенно не затрагивались вопросы финансирования научного направления), но дает представление о том, с каких позиций проводилось исследование динамики программирования как научного процесса.

5. ПОТОК ПУБЛИКАЦИЙ

5.1. Общие положения [1]

Так как основной продукт, производимый ученым, есть публикация, то анализ развернутого во времени потока публикаций является основным фактором, характеризующим динамику развития научного направления. Естественно возникающая трудность при проведении такого анализа — что считать за публикацию. Дело в том, что публикации различаются по ряду признаков: объем (от тезисов в 1 страничку, представленных на конференцию, до фолианта в 1000 страниц мелкого шрифта), место публикации (нерецензируемый «карманный» сборник трудов или престижный журнал с авторитетным штатом рецензентов), способ распространения (отчеты, изготовленные поштучно, препринты, рассылаемые по списку, и монография, изданная солидным тиражом и находящаяся в свободной продаже), число соавторов и т.д.

Известны различные подходы к преодолению этих проблем. Так, например, выбирается базовая публикация (чаще всего — журнальная публикация фиксированного объема) и все остальные публикации измеряются в этих единицах. Может быть также принято соглашение о приравнивании одной книги некоторому числу эталонной журнальной публикации. Так, известны следующие соотношения: 1:4, 1:6, 1:18 и т.д.

Ряд исследователей предлагает измерять объем публикаций в печатных листах. Такой подход внедрялся при подготовке списка печатных трудов, где предлагалось указать для каждой публикации ее объем в печатных листах. Однако такие подходы не решают задачу полностью и чаще под числом публикаций понимают число заголовков публикаций. При наличии соавторов в списках печатных трудов они указываются в столбце — примечании, где просто указывается факт наличия соавторов (фразой «в соавторстве»).

Число научных трудов есть величина, которая хорошо коррелирует с известностью ученого, с его вкладом в науку. Имеется значительная корреляция между престижем научного учреждения с числом выпускаемых им работ. Конечно можно утверждать, следуя Д. Прайсу, что число публикаций — «это очень плохая шкала». «В самом деле, — пишет далее он, — кто мог бы осмелиться попытаться уравновесить одну статью Эйнштейна по теории отнсительности хотя бы сотней статей бакалавра Джона Доу о кон-

стантах эластичности для различных древесных пород в лесах Нижнего Базутоленда, по одной константе на статью».

Можно, конечно, иронизировать по этому поводу, а можно подойти к этому делу по другому. Чтобы показать гениальность статьи Эйнштейна, нужна публикация сотен статей с расшифровкой всех деталей как специалистам, так и широкой научной общественности. В то время как сотня статей бакалавра освещает тематику констант эластичности самодостаточно. Что же касается научной ценности, то отвергать эти работы только потому, что они не востребованы в данный момент, совершенно нельзя. В мире науки мы часто встречаемся с, казалось бы, бессмысленными публикациями, касающихся совершенно «бесполезных исследований». Примерами таких работ могут служить исследования жизни насекомых, языков малочисленных народов и других аналогичных тем.

5.2. Динамика роста числа публикаций

Наиболее известным результатом исследований динамики роста числа публикаций является возможность аппроксимации кривых роста числа публикаций *экспонентой*. Иногда в этих целях используется сумма *экспонент*. При увеличении числа «колен» эмпирической кривой роста числа публикаций авторы, не желающие отказываться от языка экспонент, говорят о так называемой *скользящей экспоненте*. Понятно, что такая аппроксимация лишена смысла, если нам не удастся установить вид зависимости параметров этой скользящей экспоненты от положения точки на кривой роста. Если же такая зависимость будет найдена для данной кривой роста, то это будет означать аппроксимацию этой кривой не экспонентой, а более сложной зависимостью.

К *линейной* аппроксимации эмпирических данных по росту числа публикаций прибегают при исследованиях коротких временных интервалов. При расширении временного интервала для кривых роста часто получают зависимости так называемого *логистического типа*.

Наукометрический индикатор «число научных публикаций» разработан больше, чем какой-либо из известных индикаторов. Тем не менее он требует дальнейшей проработки. В частности, должна быть разработана формализованная процедура «взвешивания» публикаций разных типов и отдельных публикаций. Основой такого «взвешивания» может служить ранг публикаций, издания или типа издания, в котором помещена эта публикация. Ранг же может определяться средствами цитатного, социологического или какого-либо другого анализа.

Перспективы применения подсчета числа публикаций определяются результатами опорных исследований числа публикаций как наукометрического индикатора латентных переменных научной деятельности. Число публикаций может служить индикатором *признания* ученого, его *известности*, *вклада в науку*, *продуктивности*, *престижа научного заведения*.

Согласно зафиксированным в литературе опорным соображениям качественного порядка, число публикаций может служить индикатором *элитности* ученого, его *активности*, *интенсивности*. Большое число монографий в данном научном направлении связывают с его *насыщенностью*, а число журнальных статей — с его *конфликтностью*. *Ценность* журнала определяют по частотам отражения их публикаций во вторичных изданиях.

Скорость роста числа публикаций связывают с *актуальностью* данного научного направления или с его *перспективностью*.

По скорости роста числа публикаций данного научного коллектива в данный момент, соотнесенной с общей логистической кривой их роста для этого коллектива, предлагается определять *фазы развития данного коллектива*. Коллективы, находящиеся в разных фазах своего развития, перспективны в разной степени.

Стационарные распределения числа публикаций также могут быть использованы для выявления *проблемных* научных направлений, которые характеризуются распределениями, имеющими более короткие «хвосты», нежели традиционные направления. В проблемном направлении публикации рассеяны в меньшем числе журналов, чем в непроблемном.

Таким образом, возможности применения числа публикаций в качестве индикатора научной деятельности в принципе обширны. Однако до широкого практического использования результатов этих исследований еще далеко, поскольку не установлена полная картина корреляций числа научных публикаций ни с системой индикаторов, ни с системой латентных параметров науки. Кроме того, уже вычисленные корреляции устанавливались на основе гауссовой математической статистики, тогда как распределения, между которыми устанавливались эти корреляции, имеют сугубо негауссову природу. Все это означает, что опорные исследования числа публикаций должны вестись фактически заново средствами негауссовой математической статистики с широким систематическим охватом всей системы индикаторов и латентных переменных науки.

Следует напомнить, что все наукометрические оценки носят *статистический* характер. Фиксируемые опорными исследованиями связи числа публикаций как наукометрического индикатора с латентными переменными научной деятельности также носят поэтому статистический характер.

Это означает, что, не гарантируя верности оценки в каждом конкретном случае, среднестатистический метод подсчета числа публикаций при его использовании совместно с другими методами дает положительный эффект.

5.3. Динамика роста числа публикаций в программировании (1956–1975 гг.)

Данный раздел написан по материалам работы автора [2]. Рассматриваемый промежуток времени (1956–1975) охватывает время от появления первой статьи в открытой печати до массового внедрения ЭВТ в науку и промышленность. Можно утверждать, что этими исследованиями были затронуты все или почти все публикации (статьи, тезисы докладов, монографии, препринты). Не учитывались отчеты и другие научные рукописи. В период начального развития, когда публикаций мало, данные о числе публикаций весьма разбросаны и их можно аппроксимировать только на отдельных интервалах.

Итак, первое, с чем приходится сталкиваться — это крайняя неравномерность числа опубликованных работ; периоды подъема перемежались периодами спада и застоя. Так как актуальное научное направление характеризуется экспоненциальным ростом числа публикаций, было проведено исследование соответствия роста числа публикаций экспоненциальному закону. Заметим, что рост числа рефератов в реферативных журналах, рост объема информационного массива системы СКИФ и числа специализированных журналов по программированию и смежным вопросам имел ярко выраженный экспоненциальный характер, т.е. в целом в мире развитие программирования соответствовало развитию актуального направления. Однако аппроксимация всего 20-летнего периода жизни советского программирования связана со столь большим разбросом данных, что говорить об экспоненциальном законе не приходится.

Разбивая временной интервал 1956--1975 гг. на подынтервалы, можно выделить четыре подынтервала, где возможна хорошая аппроксимация:

- 1956–1958 гг.: аппроксимирующая кривая $y = 2e^{0.95x}$;
- 1959–1961 гг.: аппроксимирующая кривая $y = 4.02e^{0.917x}$;
- 1962–1968 гг.: аппроксимирующая кривая $y = 34.3e^{0.224x}$;
- 1971–1974 гг.: аппроксимирующая кривая $y = 159.66x - 132.66$.

Из приводимых данных видно снижение скорости роста, определяемого показателем экспоненты и последующим переходом к линейной функции, что свидетельствует об исчерпании потенциальных возможностей данного

направления. Это, не в последнюю очередь, объясняется произошедшей в начале 70-х годов ориентацией на линию ЕС ЭВМ и переносом основных исследований из сферы Академии наук в сферу промышленности. Но самым интерсным в этой аппроксимации является ее пилообразный характер, не встречавшийся (насколько это мне известно) в наукометрических исследованиях, а также наличие спадов в 1966–1971 и 1974–1975 гг. Складывалось впечатление, что в каждый из отмеченных выше интервалов развивалась какая-либо идея, которая быстро исчерпывала себя, не порождая новых идей, либо отбрасывалась в сторону в угоду новой, взятой со стороны идее. Эта мысль подтверждается результатами анализа библиографических ссылок.

Чтобы учесть влияние прошедших в 1968 и в 1970 гг. Всесоюзных конференций, отметим, что публикация тезисов докладов приходится на период спада с 1967 по 1971 гг. (119 — в 1967, 91 — в 1968, 70 — в 1969, 67 — в 1970, 27 — в 1971). Интересно посмотреть, на какие темы публиковал А.П. Ершов статьи в эти годы [3].

1967 г. — из 10 публикаций 8 относятся к статьям из книги «АЛЬФА — система автоматизации программирования», изданной в Новосибирске. Кроме того, одна — по теоретическому программированию и еще одна — по системе АИСТ-0 (издана за рубежом).

1968 г. — из 5 публикаций одна — обзорная, две — по теоретическому программированию, одна — руководство к пользованию системой АЛЬФА и доклад на конференции ИФИП.

1969 г. — из 9 публикаций практически все являются информационными и редакторскими.

1970 г. — из 6 публикаций практически все являются информационными.

1971 г. — из 7 публикаций 6 содержательных и только одна информационная.

Отсюда видно, что вызвать публикации--отклики на статьи А.П.Ершова могут только содержательные статьи на русском языке с некоторой задержкой, требуемой для осознания этих статей.

6. НЕЗРИМЫЕ АВТОРСКИЕ КОЛЛЕКТИВЫ [4]

Незримым авторским коллективом мы будем называть группу ученых, связанную отношением «быть соавтором». Другими словами между каждой

парой ученых существует по крайней мере одна цепочка, состоящая из учёных, связанных этим отношением. Незримый авторский коллектив может быть представлен неориентированным графом, вершины которого суть учёные — члены незримого коллектива, и две вершины смежны тогда и только тогда, когда соответствующие учёные являются непосредственными соавторами в какой-либо публикации[1]. Как велик может быть такой коллектив, и какую он может иметь структуру? Чтобы ответить на эти вопросы, нужно попытаться построить соответствующий граф, используя библиографические указатели или другой имеющийся материал. Так, изучая развитие незримых авторских коллективов в программировании на заре его развития, был построен соответствующий граф с использованием библиографических указателей «Математика в СССР за 40 лет» и «Математика в СССР за 50 лет». Использование таких указателей объяснялось тем, что программирование возникло при тесном контакте с математикой. Этот граф показал, что первые два коллектива — московско-новосибирский и киевский — входили в качестве двух непересекающихся по вершинам связных подграфов в большой математический коллектив.

Пусть дан граф незримого авторского коллектива. Для него можно определить центр как вершину или группу вершин, равноотстоящих от наиболее удалённых вершин. Такой центр можно рассматривать как неформального лидера коллектива. Если же в коллективе есть и формальный лидер, то может быть поставлен вопрос об устойчивости этого коллектива.

Описанный выше подход носит статический характер. Можно рассматривать развитие коллектива в динамике, когда с течением времени к графу добавляются новые вершины и рёбра и одновременно часть прежних элементов удаляется. Такие графы достаточно наглядно отображают перемены в коллективе, связанные, например, с уходом прежнего формального лидера.

7. ГРАФ РАЗВИТИЯ ИДЕЙ [2]

Граф развития идей отображает связь высокоцитированных работ между собой. Исследуемый период времени разбивается на отрезки примерно равной длины и для каждого такого отрезка определяется понятие высокоцитируемой работы. Тогда ориентированный граф, вершины которого суть высокоцитируемые публикации и две вершины v и w которого соединены дугой (v, w), если публикация w цитирует публикацию v, называется *гра-*

[1] Все неопределяемые здесь термины теории графов могут быть найдены в [5].

фом развития идей. На таком графе отчётливо видны источники новых идей. Так на графе развития идей, приведённом в [2], видно, как происходит замещение отечественных источников зарубежными.

7.1. Цитирование работ ученого

Цитирование работ учёного является одной из важнейших характеристик его научной деятельности. Однако применение её на практике упирается в проблему сбора ссылок. Индекс Гарфилда не может дать объективную оценку цитируемости работ конкретного учёного из-за далеко неполной выборки журналов. За пределами внимания остаётся цитирование в книгах, сборниках трудов, препринтах и т.д. А это очень важно, так как показывает использование результатов в практической деятельности учёных и практиков. В НФ ИТМ и ВТ было налажено отслеживание цитирования работ А.П. Ершова и сотрудников НФ в новых публикациях, поступавших в библиотеки А.П. Ершова и НФ, а также цитирование этих работ по Индексу Гарфилда. Собранная информация помещалась в ежегодные библиографические указатели публикаций НФ наукометрического типа. Примером такого указателя может служить Указатель трудов А.П. Ершова, выпущенный к его 50-летию силами НФ и ВЦ СО АН СССР [3].

В указателе, состоящем из двух частей, помещены как библиография публикаций А.П.Ершова, так и сведения об их цитировании. В библиографии публикации располагаются в хронологическом порядке с 1955 по 1981 г. В рамках каждого года работы располагаются по алфавиту названий (вначале работы, изданные на русском, затем на иностранных языках). В качестве дополнения помещён алфавитный указатель трудов и список соавторов.

В указателе цитированных работ за библиографическим описанием печатной работы А.П. Ершова следует перечень работ, авторы которых ссылались на данную публикацию. Этот перечень упорядочен по времени издания, что позволяет проследить динамику развития интереса к результатам, изложенным в основной публикации. Из данного перечня изъяты все работы сотрудников ВЦ и НФ. Это связано, с одной стороны, с необходимостью оставаться в разумных рамках по объёму указателя, а с другой стороны, с желанием подчеркнуть независимое от личного общения с А.П. Ершовым распространение его идей и взглядов, т.е. читаемость его работ.

7.2. Научные связи

Можно определить научное пространство учёного N как совокупность учёных $\{S\}$, связанных с N различными научными отношениями, как например, связи типа *научный руководитель — оппонент, автор книги — титульный редактор, автор книги — рецензент (не анонимный)*, а также связи, устанавливаемые выражениями благодарности за содействие в написании работы, за высказанные замечания, за прочтение рукописи и т.д. Такая работа по выявлению научных связей А.П. Ершова по защитам была проделана на базе массива из 300 авторефератов диссертаций, защищавшихся на советах, членом которых был А.П. Ершов.

8. ПРОСТРАНСТВО ЖУРНАЛОВ

В заключение укажем еще на одну задачу, относящуюся к наукометрии, — это задача исследования пространства журналов. (Конкретно нами рассматривались журналы по программированию и смежным вопросам [4].) Рассмотрим граф, вершины которого суть журналы и две вершины A и B соединены дугой (A, B), если статьи из журнала A цитировались в статьях в годовом комплекте (томе) журнала B не менее k раз, где порог цитирования k устанавливается эмпирическим путём. В полученном орграфе вершины подразделяются на вершины-входы, вершины, принадлежащие бикомпонентам или компонентам сильной связности, и вершины из бесконтурной части. Первые соответствуют журналам, оказывающим сильное влияние на остальные журналы, а сами не подвергаются постороннему влиянию. Это очень важная группа журналов. Журналы второй группы образуют как бы супержурналы, каждый из которых состоит из журналов, соответствующие вершины которого составляют одну бикомпоненту. Поэтому при подготовке обзоров и при комплектовании фондов библиотек нужно отслеживать все журналы, образующие бикомпоненту. Остальные журналы имеют ограниченное влияние.

Для журналов по программированию было проведено исследование на базе журналов за 1979 и 1982 гг. В 1979 г. было три журнала-входа (Datamation и два журнала фирмы IBM), которые влияли на супержурнал из 9 журналов, включающий такие издания, как Communication of the ACM, Journal of ACM и др. Этот супержурнал оказывал влияние на все остальные журналы. В 1982 г. была несколько другая картина, хотя общая схема осталась прежней. Бывшие входы потеряли свое влияние; их место занял журнал Communication of the ACM, состав супержурнала изменился в сторону

расширения — в него вошло уже 12 журналов, включая такой журнал, как Theoretical Computer Science. Это отражало изменение в расстановке сил (потеря лидирующих позиций IBM), усиление влияния теоретических исследований, появление новых перспективных направлений (к примеру искусственный интеллект).

9. ЗАКЛЮЧЕНИЕ

К сожалению, описываемые методы плохо применимы в условиях переходного периода от одного стабильного состояния к другому. Резкое сокращение финансирования науки, массовый отъезд учёных за границу, распад СССР привели к ликвидации сложившихся авторских коллективов, резкому уменьшению числа публикаций и изменению их структуры, нарушивших сложившиеся традиции цитирования и т.д. и т.п. Кроме этого, большое и еще не полностью осознанное влияние на развитие науки оказала и продолжает оказывать система электронных публикаций в Интернете. Всё это делает применение традиционных методов наукометрии неэффективным и может быть даже бесперспективным.

СПИСОК ЛИТЕРАТУРЫ

1. Хайтун С.Д. Наукометрия: Состояние и перспективы. — М.: Наука, 1983.
2. Евстигнеев В.А. Развитие программирования в СССР в 1956–1975 гг.: взгляд с точки зрения наукометрического анализа // Теория и практика систем информатики и программирования — Новосибирск: НГУ, 1988. — С. 72–80.
3. Андрей Петрович Ершов. Библиографический указатель литературы. — Новосибирск: ВЦ СО АН СССР, 1981.
4. Евстигнеев В.А. Методы теории графов в наукометрии: исследование структуры пространства журналов и незримых коллективов в программировании. Новосибирск, 1987. (Препр. / АН СССР. Новосиб. филиал. ИТМ и ВТ им. С.А.Лебедева; № 4).
5. Евстигнеев В.А., Касьянов В.Н. Толковый словарь по теории графов в информатике и программировании. — Новосибирск: Наука. Сиб. предприятие РАН, 1999. — 291 с.

А.Р. Данилин

ВОСПОМИНАНИЯ О ВЛШЮП

В конце 70-х годов была осознана необходимость формирования компьютерной грамотности у широкого круга учащейся молодежи. К этому моменту в разных регионах Советского Союза уже велась активная работа в этом направлении. Идейным и организационным лидером этого процесса стала группа школьной информатики при ВЦ СОАН СССР во главе с А.П. Ершовым. Именно этой группе совместно с журналом «Квант» удалось осуществить весьма сложный проект — организацию Всесоюзной заочной школы юных программистов. Идейным манифестом всего движения стала статья А.П. Ершова «Программирование — вторая грамотность».

Разумеется, для успешной реализации такого широкомасштабного проекта требовалась большая помощь в организации проверки работ учащихся. Такая помощь и была организована на базе центров по работе со школьниками в области информатики в разных регионах в форме открытия региональных филиалов Заочной школы. В частности, Уральский филиал был организован на базе Свердловского педагогического института. Следует отметить, что к этому моменту в Свердловске был накоплен значительный опыт по работе с учащимися: работали кружки, проводились городские олимпиады по программированию. Благодаря энтузиазму заведующего кафедрой вычислительной математики и программирования Свердловского педагогического института В.Г. Житомирского в этом институте был создан не только вычислительный центр, оснащенный самой современной на тот момент техникой, но и открыта педагогическая специальность «Организатор учебного процесса на базе ЭВМ». Именно студенты этой специальности вели большую часть работы по проверке заданий учащихся заочной школы.

Мне, в то время молодому кандидату наук, работавшему на кафедре у В.Г. Житомирского, было поручено возглавить Уральский филиал ВЗШЮП. В Уральский филиал кроме Свердловской области входили Башкирия, Удмуртия и Челябинская область. Естественно, что, когда было принято решение об организации в Новосибирске в 1980 году летней школы юных программистов для лучших учащихся ВЗШЮП, мне пришлось участвовать в формировании команды Уральского филиала и везти эту ко-

манду в Новосибирск. В команду вошли представители всех этих областей и республик и различных возрастных групп с 6 по 10 класс.

У организаторов школы к тому моменту уже был накоплен значительный опыт проведения подобных мероприятий, так как первая всесоюзная школа была для новосибирцев уже пятой. Кроме этого сотрудники группы школьной информатики сами являлись руководителями школьных кружков и объединений и тем самым не только имели представление о содержании и методах соответствующей работы, но и имели рядом с собой значительный отряд подготовленных помощников из числа своих учеников.

Вся практическая работа школы в то время держалась на трех «китах» (по крайней мере так это выглядело на мой взгляд, как руководителя делегации). Назову всех поименно в алфавитном порядке: Геннадий Анатольевич Звенигородский, Юрий Абрамович Первин, Нина Ароновна Юнерман. Разумеется, за спиной этой группы была мощная поддержка Андрея Петровича Ершова со всем его «весом» в научных, педагогических и государственных структурах.

Работа школы была организована в формах, близких к вузовским: лекции, семинары, практические занятия. При этом большое внимание уделялось практической работе с ЭВМ, поскольку многие, а точнее, подавляющее большинство участников школы впервые получило доступ к настоящей ЭВМ. Отметим, что многие ребята приехали из сел и небольших городков, где в то время вычислительной техники просто не было.

Кстати, память об этих школах согревает теплым чувством сердца многих ее учеников во уже более 20 лет. В частности, один из них, проживающий в небольшом райцентре Башкирии — г. Туймазы, недавно прислал мне по моей просьбе список лекций, прочитанных в этой школе. Не могу не привести его здесь.

Тема 1. Работа на терминалах и перфораторах.
Тема 2. Особенности систем РАПИРА 1 и ШПАГА 2.
Тема 3. Режимы использования вычислительных машин.
Тема 4. ОС ДИСПАК или ДИАПАК.
Тема 5. Диалоговая сервисная система "КРАБ".
Тема 6. Язык программирования СЕТЛ.
Тема 7. Язык программирования АЛГОЛ-ГДР.
Тема 8. Диалог с системой ДИСПАК и дополнительные возможности системы КРАБ.
Тема 9. Особенности и примеры работы системы СЕТЛ.

Этот факт интересен не только забытыми терминами «компьютерной» жизни тех лет, но и тем, что ученики школы до сих пор поддерживают связь через океаны и континенты со мной и друг с другом.

Важным элементом учебного процесса было проведение двух конференций. Одна проходила в начале школы, и на ней школьники рассказывали о работах, выполненных дома. Вторая — в конце — целиком посвящалась работам, сделанным за время летней школы. Необходимость выступить на этой конференции придавала практическим занятиям реальный смысл.

С течением времени учебный процесс школы видоизменялся, и наряду с небольшим числом общих лекций, возникли циклы лекций по более узкой тематике, предназначенные для группы, занятой выполнением конкретного проекта. Позднее такие группы трансформировались в мастерские под руководством мастера — профессионала и при помощи подмастерьев из числа выросших учеников. Именно участие школьников не только в роли учеников, но и в роли организаторов, консультантов, помощников, подмастерьев помогало поддерживать партнерские отношения между старшими и младшими и создавало особую атмосферу школы.

Особенно это стало заметным, когда школа переехала в лагерь на берегу Обского моря, и жизнь ее участников стала более насыщенной общением. При этом подросли, набрались опыта, стали студентами бывшие воспитанники (не только новосибирцы), которые взяли на себя большую часть работы по организации и проведению школы. Люди, знакомые с методикой коллективных творческих дел, без труда узнавали проявления этой методики в жизни школы.

Школа учила не только программированию, но и общению. Костры, поэтические вечера, школа бальных танцев, футбольные матчи между Европой и Азией способствовали сближению всех участников школ и, нередко, многолетнему сотрудничеству.

В школе учились не только ученики, но и педагоги. На каждой школе работал семинар для учителей. И если на первых школах обсуждались вопросы кружковой работы в области программирования, то в дальнейшем на первый план вышли темы, связанные с введением нового тогда для общеобразовательной школы курса информатики.

Трудно переоценить роль новосибирских летних школ в распространении методических идей по преподаванию информатики. На одном из таких семинаров обсуждался еще готовящийся к изданию учебник под редакцией А.П. Ершова. Андрей Петрович привез рукопись, попросил педагогов прочесть ее и высказать свои суждения, внимательно выслушал всех, с чем-то согласился, где-то твердо стоял на своем, говоря: «Здесь легкой жизни не

ждите». Не случайно многие педагоги — участники летних школ — стали пионерами преподавания нового предмета.

Замечательно, что распространялась по стране не только «новосибирская» методика преподавания информатики, но и атмосфера летних школ. Возникли дочерние школы и компьютерные клубы (не как место для игр, а как сообщество единомышленников). Непосредственно мне пришлось столкнуться с компьютерным клубом «Виртуал» из г. Кургана и Летней школой информационных технологий в г. Миассе. Организаторы и клуба, и школы не скрывали, что идея создания этих объединений появилась у них после участия в ВЛШЮП.

Разумеется, данная заметка не претендует на полноту описания такого крупного явления, как ВЛШЮП, но я надеюсь, что мне удалось выразить свои ощущения духа и атмосферы школы, которые я бережно храню вот уже более 20 лет.

М. Ю. Колодин

УРОКИ ЗАОЧНЫХ ШКОЛ ПРОГРАММИРОВАНИЯ

В 1979–1982 гг. в журнале «Квант» в разделе «Искусство программирования» публиковались уроки Заочной школы юных программистов (ЗШП) и статьи по компьютерной тематике. Эта школа прошла успешно несмотря на значительные организационные и издательские сложности, выпустила множество квалифицированных специалистов и дала основу для всесоюзного школьного предмета «Основы информатики и вычислительной техники». Были организованы филиалы ЗШП в нескольких крупных центрах (Ленинград, пр.), проводились Летние школы юных программистов (ЛШ). Были выпущены методические и справочные пособия, учебники информатики. Однако ЗШП преподала множество уроков не только школьникам, но и ее организаторам. Автор, ученик ЗШП 1979–1982 гг., рассказывает о процессе, результатах, подводит некоторые итоги, как они видятся именно со стороны ученика и, впоследствии, преподавателя школы.

Однажды, в один из самых обычных дней сентября 1979 года, на самый обычный урок вошёл наш учитель математики, Михаил Сергеевич Маховер. В руках у него был свежий номер журнала «Квант». «В этом журнале появился новый раздел, — сказал он, — Я не очень в этом разбираюсь, но кажется, что это интересно, это может стать вашей будущей специальностью». И вместе с несколькими одноклассниками, а я тогда был в пятом классе, я стал изучать уроки Заочной школы, писать ответы в Новосибирский ВЦ СО АН. Этот день определил мою профессию и, в немалой степени, всю мою жизнь. И я благодарен Михаилу Сергеевичу за это. Равно как и моим родителям, поддержавшим это начинание и помогавшим в освоении учебного материала; я тогда был «чистым математиком» — занимался в пяти математических кружках сразу, а вот ЭВМ я впервые увидел и потрогал на работе у мамы, заведующей группой ИВЦ СЗРП. Одновременно со мной стали отвечать на задания ЗШП несколько одноклассников; однако до конца Школы добрались, увы, немногие.

В следующем (1980) году появились филиалы ЗШП, в том числе и в Ленинграде, в Институте авиационного приборостроения (ЛИАП, позже СПбГААП (Академия), а ныне СПбГУАП, Государственный университет аэрокосмического приборостроения). А ещё через год в ЛИАПе открылась очная школа. Впоследствии я поступил в тот же институт, чему школа программирования была непосредственной причиной. Основными ведущими в

теперь уже очной школе стали энтузиасты школьной информатики Николай Николаевич и Галина Николаевна Бровины, под покровительством зав. кафедрой 44 (ЭВМ, ныне кафедра Вычислительных систем) М.Б. Игнатьева.

Но вернёмся к ЗШП. Её уроки публиковались почти в каждом номере журнала «Квант». Если не было уроков (декабрьские и летние номера), обычно были какие-либо обзорные статьи по разным темам, а в 12-х номерах — рассказы о Летних школах юных программистов в г. Новосибирске. Полный список материалов ЗШП приведён в приложении к этой статье.

Сейчас понятно, с каким трудом давались эти уроки. Сложность набора, нетипичного для физико-математического журнала, попытки вместить довольно-таки большой учебный материал, более удобный для полноформатных учебников, в несколько журнальных страниц — всё это давало не самый удачный результат. Было множество опечаток, причём в том числе и в правилах, и в примерах, что сильно мешало пониманию материалов. Я, например, иногда списывал конструкции своих программ с правил в журнале, подставляя нужные значения и переменные (собственно, так и предполагалось на первых порах), и однажды получил такой ответ от преподавателя: «Ответы этого урока не проверяются из-за ошибок в тексте урока в журнале». Таким образом сами учителя видели, что в учебном материале много неточностей, которые не дают возможности правильно понимать тему.

Впоследствии «Квант» больше не смог повторить такую Школу. Более того, ни один журнал, в том числе «Информатика и образование», не подняли проект такого масштаба. Представляется, что сейчас это невозможно: программирование стало слишком большим и сложным, а ограниченный объём журнальных публикаций, длительные периоды между отдельными выпусками журнала (типичный срок — месяц — слишком долог для поддержания постоянного обучения, теоретического и практического, для обеспечения активного роста ученика и его взаимодействия с учителем) не давали возможности правильно его преподать. Другие современные методы, например, веб-сайты, рассылки, тоже не решают проблему обучения. По-видимому, наиболее продуктивный на сегодня метод — дистанционное обучение через Интернет с помощью специальных программных средств. Однако тогда таких средств не было. Как и, практически, самого Интернета в России.

Тем не менее, было множество оригинальных и правильных находок. Прежде всего, весьма обоснованным, педагогически правильным, был подбор материала, его последовательность, выбор примеров. Многие уроки и примеры давались в игровой, весёлой форме, что облегчало освоение не-

обычного материала. «Позовите, пожалуйста, кота Матроскина к телефону. — Кот Матроскин подойти к телефону не может, он очень занят, он на печи лежит...» — примеры предписаний и утверждений. «Если слово “караул” хорошенько подредактировать, получится “ура”» — эпиграф к обработке строк и т.п. При явной бедности графических устройств того времени графики и рисунки появились у нас с первых же уроков. А наглядность работы, вид результата для школьников очень важны. Принципиален тот момент, когда программист, особенно юный, видит: «Вот я написал, и вот оно ожило, оно работает».

Важно, что первая же статья А.П. Ершова и Г.А. Звенигородского посвящалась обоснованию, мотивированию программирования: «Зачем надо уметь программировать?» Таким образом, с самого начала мы жили с сознанием того, что занимаемся чрезвычайно полезным делом.

Удачным был выбор синтаксических диаграмм для объяснения синтаксиса изучаемого языка (поначалу — Робика и Рапиры, затем — Паскаля). Более того, в отличие от Н. Вирта, преподаватели ЗШП использовали овальные очертания для изображения переменных элементов и прямоугольные — для постоянных. Это просто и наглядно, лучше соответствует смыслу понятия.

Выбор языков программирования также был, безусловно, очень удачным. У нас не пошли по стандартным западным вариантам, а, тщательно проанализировав положение с языками, доработали их или разработали свои. Свои оказались даже лучше иностранных.

Вначале — краткое введение на небольшом учебном языке Робик. На нём дано общее понимание понятий программирования, первые конструкции. К моменту, когда ученик начинал испытывать нехватку выразительных средств, уже через четыре месяца, ему давался более мощный инструмент — «учебно-производственный язык Рапира».

Вообще, разработка этого языка и создание его реализаций на разных платформах -- отдельный и особенный этап в развитии нашей учебной информатики. Школьники полноправно участвовали в создании мощного средства программирования под руководством опытнейших системных программистов страны.

Сама Рапира развивалась и менялась. То, что было описано в «Кванте», отличалось от предварительных версий, а реализации на различных компьютерах в разное время — тем более. Не все обозначения были удобными. Например, «!» для операции возведения в степень выглядит нелогично. Введённые в язык записи были неудобны в работе. Ограничение имён шестью символами является слишком сильным и мешает методически (и тех-

нологически) правильно разрабатывать программы. Ограничение длины чисел 12-ю цифрами не обосновано. Педагогически верным представляется использование различных символов для сравнения на равенство и присваивания, запрет использования ключевых слов языка в качестве имён переменных. По операторам Рапиры тоже есть замечания. Прежде всего, специальный знак «::» в операторе цикла смотрится весьма непонятно. В Рапире того времени было два способа вызова параметров — классический, со скобками (пустыми при отсутствии параметров), и дополнительный, в форме «.СБРОС» или «5.КРУГ», для процедур, имеющих не более одного параметра; представляется, что вторая форма была избыточной и только вносила путаницу в юные умы. Сокращения слов типа «КНЦ» для «КОНЕЦ» также не помогали: много букв на этом всё равно не сэкономить, а понимание страдает; поскольку человек воспринимает текстовую информацию не буквами, а словами, правильнее использовать привычные взгляду конструкции, распознаваемые целиком, как единое целое. Мощные структуры данных, кортежи и множества, делали язык весьма сильным. Помнится, даже были обсуждения, нужны ли в учебных языках мощные конструкции, отсутствующие в «реальных» языках, поскольку привыкший к такой роскоши выпускник может столкнуться с трудностями в последующей учёбе и работе.

Общий план изучения всех языков был примерно одинаковым: от простого к сложному. В то же время не хватало справочников, где вместе сводились бы правила языка. Фразы типа: «Сообщаем шестиклассникам эту формулу...» (например, для корней квадратного уравнения) создавали впечатление сложности используемого неизученного материала, особенно с учётом того, что на уроки ЗШП отвечали и более младшие школьники. От номера к номеру несколько менялся стиль записи программ, текстов уроков, заданий: они оформлялись по-разному, до курса по Паскалю практически никогда не использовалась структурированная запись «лесенкой», а иногда в текстах программ допускались переносы ключевых слов со строки на строку, что совершенно недопустимо. Хорошо, что при изучении геометрии давались и декартова, и полярная система координат, при рассмотрении хранения информации — и адресный, и ассоциативный поиск. Рассматривались приближённое решение уравнений (метод половинного деления, метод перебора с возвратом), работа с графами (простые, нагруженные графы, представление их в памяти ЭВМ; хотя изучение их было существенно неполным, но некоторое представление о теме давало).

После Рапиры был Паскаль — уже не просто как учебный язык, а как серьёзное средство для изучения важнейших концепций и конструкций

программирования. На нём обучение шло до конца ЗШП и далее — во втором, дополнительном, курсе «Фразеология программирования (стандартные приёмы программирования)», который вёл замечательный преподаватель, интереснейший человек — Леонид Штернберг. Полезными оказались не только разбираемые в уроках примеры, но и чётко сформулированные правила программирования и организации работы.

Выбор Паскаля был сознательным, оправданным, чётко проработанным. Вообще, отношение к выбору методов и средств обучения в ЗШП и, в дальнейшем, в ЛШ было принципиальным, многократно серьёзно изученным теоретически и практически. В работах Г.А. Звенигородского и коллег рассматривались имеющиеся и требуемые характеристики языков, предназначенных для обучения и дальнейшей профессиональной деятельности, и по множеству критериев были созданы или доработаны несколько языков. Для них были разработаны соответствующие программы изучения и преподавания с должной методической и технической поддержкой.

Программы на Паскале публиковались более корректно, хотя были ошибки и в самом материале, например, при описании распечатки значений перечислимого типа (урок 16, «Квант» 11/1983, с. 43). В целом, качество и интенсивность уроков повысились, но меньше стало адаптированности к школьному уровню, всё изложение стало несколько более формальным и перечислительным. Опечатки остались; порой они описывались на той же странице урока, что и сама опечатка (например, «Квант» 1/1982, с. 52). Файлы вводились весьма сложно, использовали авторский «буферный элемент», и, насколько я видел, с использованием их у новичков было много проблем. То же относится и к динамическим структурам: типы указателей и примеры их использования давались с трудом. Параметры-процедуры и параметры-функции вводились без должного предварительного объяснения, а ведь сам факт возможности передачи программного элемента в качестве параметра в другой программный элемент — достаточно сложный для понимания момент. Здесь причина, по-видимому, в том, что писал это опытный программист, для которого такие конструкции были давно понятны, и не учитывался малый опыт учеников.

Понятие об отладке программ было введено в одном из начальных уроков. Речь шла в то время только о ручной прокрутке. Например, в пятом уроке (язык Рапира) использовались «таблицы имён», куда вписывались значения, присваиваемые переменным, при этом зачёркивались прежние значения. Этот момент, отличающий работу с именами в математике и программировании, был дан сразу и корректно. А вот методов и средств орга-

низации и автоматизации более полного тестирования, а тем более — доказательства правильности программ в базовом курсе не было совсем.

После цикла уроков по языкам в «Кванте» были выпущены методические пособия с более подробным и проработанным курсом, в частности, языка Рапира. Были и методические разработки, и статьи в других журналах, и книги — в частности, учебники программирования, основывавшиеся, к сожалению, не на Рапире, фактически реализованной на многих программно-аппаратных платформах (в том числе на ПЭВМ Агат и Ямаха, на ЕС ЭВМ и БЭСМ-6, и пр.), а на «алгоритмическом языке» (прозванном «Ершолом»), который так и не получил серьёзной практической поддержки.

Важнейшей характеристической особенностью Школ, как очных, так и, фактически, заочных, было привлечение самих школьников к проработке и разработке программных средств и, после прохождения первоначального обучения, к преподаванию. Школьники были и программистами, и консультантами.

Ещё один успешный результат ЗШП: надъязыковый подход. Мы практически с самого начала стали работать на нескольких языках: после быстрого прохождения Робика перешли к Рапире (т.е. уже знали, что языков несколько, много, и каждый для чего-то хорош, а для чего-то плох), а затем и к Паскалю. В дальнейшем на ЛШ была относительно большая свобода в выборе программных средств: можно было пользоваться разной техникой и разными системами программирования. Это очень важно; уверен, что нашим последующим прогрессом мы в немалой степени обязаны умению выбирать, которое в нас заложили с самого начала. Тот, кто пишет только на Бейсике (а это не редкость и в современных школах), заведомо не сможет достичь таких гибкости, производительности и качества работы, как это получалось на наших Школах.

Одним из результатов ЗШП и ЛШ стало приобщение выпускников и даже старших учеников к преподаванию информатики — в той же школе, в своих кружках. Например, автор этих строк, начав изучение программирования в пятом классе теоретически, а в седьмом — практически, с девятого класса стал вести свой кружок программирования в 11 школе Василеостровского района г. Ленинграда. И у меня было много учеников, с которыми до сих пор сохраняются добрые отношения как личные, так и профессиональные.

По результатам ЗШП и очных местных школ было выпущено немало учебной и методической литературы. Возможно, в будущем и об этом напишут историки.

Сейчас появляются статьи и реплики, где говорится о «провале информатизации школы по Ершову». Насколько обоснованы такие утверждения? Мне представляется, что такое резкое мнение принципиально неправильно. Конечно, сейчас понятно, что можно было бы построить программу лучшим способом. Но — в истории этого «бы» нет. Тогда этого сделать было нельзя. На понимание, на развитие нужно время и опыт. ЗШП и ЛШ как раз и были тем самым опытом, без которого невозможно было бы достичь требуемых результатов, по крайней мере, так, как они понимались. Сейчас пересмотру подвергаются базовые принципы тогдашней школьной информатики, в частности, знаменитое правило А.П.Ершова «Программирование — вторая грамотность». Сейчас, в XXI веке, это понимается иначе. Но тогда сделали всё, что могли, причём сделали хорошо. Указанная «информатизация по Ершову» начиналась с нуля; у нас в стране ещё не было реального масштабного опыта преподавания информатики в школах, не было учительских кадров, организационных, методических, учебных материалов, техники; всё делалось впервые. Так что и судить нужно не с позиций сегодняшнего дня, который своим существованием и обязан в значительной степени той информатизации, а сравнивая то, что было до Ершова и его коллег, и то, что стало после их работ. И Заочная школа — существенный, важнейший элемент этого большого вклада в развитие школы и общества.

Справочная информация

Авторы статей и уроков: А. Ершов, Г.Звенигородский, Н. Юнерман, Ю. Первин, А. Рар, В. Касьянов, А. Салтовский, Б. Бабаян, Е. Кузнецов, Л. Штернберг.

Выпуски журнала «Квант»:

09.1979 Представляем новый раздел «Искусство программирования».
А. Ершов, Г. Звенигородский. Зачем надо уметь программировать?
ЗШП. Урок 1. Законы программирования. Правила записи предписаний на языке Робик. (Г. Звенигородский)
ЗШП. Урок 2. Гибкие системы предписаний, синтаксические диаграммы и переменные поля. (Г. Звенигородский)

10.1979 ЗШП. Урок 3. Работа с памятью. Имена и их значения. (Г. Звенигородский)

11.1979 ЗШП. Урок 4. Арифметические предписания языка Робик. Условные и циклические предписания. (Г. Звенигородский)

12.1979 IV Всесоюзная летняя школа юных программистов

01.1980 ЗШП. Урок 5. Основные операторы учебно-производственного языка Рапира. (Г. Звенигородский)
А. Салихова, Н. Соколова. Графическая система Шпага.
02.1980 ЗШП. Урок 6. Описание и вызов процедур на Рапире. (Г. Звенигородский)
03.1980 ЗШП. Урок 7. Функции, графика, локальные имена и тексты на Рапире. (Г. Звенигородский)
ЗШП. Урок 8. Множества и кортежи на Рапире. (Г.Звенигородский)
Олимпиада по программированию. (Оргкомитет)
04.1980 Ю. Первин, А. Салтовский. Память ЭВМ.
05.1980 Ю. Первин, А. Салтовский. Как работает процессор.
06.1980 А. Салтовский. Организация ввода и вывода в ЭВМ.
07.1980 ЭВМ на Олимпиаде.
Ю. Первин. Обработка протоколов соревнования по прыжкам в высоту.
09.1980 ЗШП. Урок 9. Циклы. (Ю. Первин)
10.1980 ЗШП. Урок 10. Программирование задач в полярной системе координат. (Н. Юнерман)
11.1980 ЗШП. Урок 11. Обработка текстов на ЭВМ. (Г. Звенигородский)
12.1980 V Всесоюзная летняя школа юных программистов.
01.1981 Ю. Первин. Зачем и как детей учат программированию?
Конкурс работ по программированию.
ЗШП. Урок 12. Ассоциативный поиск. (Ю. Первин, Н. Юнерман)
02.1981 ЗШП. Урок 13. Приближенное решение уравнений с помощью ЭВМ. (Н. Юнерман)
03.1981 ЗШП. Урок 14. Работа с графами. (Г. Звенигородский)
04.1981 Ю. Первин. Трехадресные, одноадресные и... безадресные машины.
05.1981 Г. Звенигородский, Е. Кузнецов. Что такое мини-ЭВМ?
07.1981 А. Салтовский. ЕС ЭВМ — семейство универсальных вычислительных машин.
08.1981 Б. Бабаян. Многопроцессорный вычислительный комплекс «Эльбрус».
09.1981 А. Рар. Какие бывают языки программирования.
Конкурс машинных рисунков.
10.1981 ЗШП. Урок 15. Основные понятия языка Паскаль. (Н. Юнерман)
11.1981 ЗШП. Урок 16. Условный оператор и оператор выбора. Операторы цикла. Процедуры и функции в Паскале. (Н. Юнерман)
01.1982 ЗШП. Урок 17. Массивы и записи в языке Паскаль. (А. Рар)

02.1982 ЗШП. Урок 18. Файлы. (А. Рар)
Добавление к уроку 18. Сортировки по текстовым ключам. Слияния. (Ю. Первин)
03.1982 ЗШП. Урок 19. Динамические структуры. (А. Рар)
04.1982 ЗШП. Урок 20. Параметры процедур и функций. (В. Касьянов)
05.1982 ЗШП (завершение 3-летнего цикла). Сортировки по текстовым ключам. Включение и распределение. (Ю. Первин)
06.1982 VI Всесоюзная летняя школа юных программистов. (Г. Звенигородский)
08.1982 Ю. Первин. Однажды вечером в семье программиста.
10.1982 Стандартные приемы программирования (Фразеология программирования).
Урок 1. Счет по рекуррентным формулам. (Л. Штернберг)
Вокруг ЭВМ.
11.1982 Стандартные приемы программирования (Фразеология программирования).
Урок 2. Линейный поиск. (Л. Штернберг)
12.1982 Стандартные приемы программирования (Фразеология программирования).
Урок 3. Переработка массивов на месте. (Л. Штернберг)
01.1983 Стандартные приемы программирования (Фразеология программирования).
Урок 4. Синхронная и асинхронная обработка массивов. (Л. Штернберг)
02.1982 Стандартные приемы программирования (Фразеология программирования).
Урок 5. Сколько будет 2x2 на ЭВМ, или Машинная арифметика и как с нею бороться. (Л. Штернберг)
03.1982 Стандартные приемы программирования (Фразеология программирования).
Урок 6. Настройка программы по параметрам. (Л. Штернберг)
Дополнительно:
Машинная графика, например, Ю. Котов. Фантазии ЭВМ.
Юмор (Квант улыбается).

С материалами этих статей можно сейчас ознакомиться, пожалуй, только на веб-сайте журнала «Квант», поскольку бумажные версии давно стали библиографической редкостью.

Интересные материалы по данной теме можно посмотреть также на сайте автора `http://myke.da.ru`.

М. Ю. Колодин

МЕСТНЫЕ ШКОЛЫ ПРОГРАММИРОВАНИЯ

Заочная школа программирования в журнале «Квант» — важное событие в развитии как советской информатики, так и советской школы. Началась она в 1979 году. Но не менее важными оказались менее звучные события — вскоре в разных городах Советского Союза открылись филиалы ЗШП.

Вот фрагмент из «Кванта» 9/1979: «Ровно год назад в “Кванте” были напечатаны первые уроки Заочной школы программирования, организованной редакцией нашего журнала и Вычислительным центром Сибирского отделения АН СССР. Сегодня в нашей школе занимаются свыше двух тысяч ребят разного возраста — от третьего до десятого класса — из всех республик Советского Союза и некоторых зарубежных стран. В работе Школы участвуют более 40 групп “Коллективный ученик”, создано два региональных центра (в Ленинграде и Свердловске). Летом 1980 года активисты Заочной школы были приглашены на VI Летнюю школу юных программистов, проходившую в новосибирском Академгородке... Школьники, проживающие в прибалтийских республиках, в Псковской, Новгородской, Ленинградской, Архангельской и Вологодской областях, в Карельской и Коми АССР, направляют свои работы по адресу: 190000, Ленинград, ул. Герцена 67, ЛИАП, кафедра вычислительной математики и АСУ, Северо-Западный филиал Заочной школы программирования. Школьники, проживающие в Свердловской, Пермской, Челябинской и Оренбургской областях и Башкирской АССР, пишут по адресу: 620219, Свердловск, ул. К.Либкнехта 9, Педагогический институт, кафедра вычислительной математики и программирования, Уральский филиал Заочной школы программирования. Жители остальной территории СССР и зарубежные читатели направляют работы по прежнему адресу: 630090, Новосибирск 90, проспект Науки 6, ВЦ СО АН СССР, отдел информатики, Заочная школа программирования...»

О Михаиле Борисовиче здесь нужно сказать несколько слов. Два важнейших качества, столь полезные для руководителя учебного и научного подразделения, присущи этому человеку. Прежде всего — предвидение. Ещё в конце 70-х годов прошлого века он осознал важность учебной информатики и оказал ей поддержку в своём подразделении. Второе качество, не менее важное, можно простым языком выразить так: «ему не жалко». Во

многих случаях, когда от него что-то зависело, и даже если это было не так, от него можно было получить помощь, будь то выделение аудиторий, машинного времени (весьма дефицитного) или оформление пропусков для школьников (для режимного вуза это всегда было проблемой). То же относится и к поддержке новых, необычных идей, направлений как в школьных, так и позже в студенческих работах.

Основной движущей силой нашей, ленинградской, школы были Н.Н. и Г.Н. Бровины. Занятия были преимущественно практическими. Основными языками стали не Рапира или Паскаль, а Фортран и Ассемблер. Главные компьютеры — М-6000, СМ-1 с не самыми полезными для глаз мониторами СИД, несколько позже — Агаты, а в дальнейшем — ЕС ЭВМ (ЕС1045 в ВЦ ЛИАПа) и СМ-2. Одной машины, даже с системой разделения времени, на всех не хватало. Особенно это было заметно к концу занятий, когда все почти одновременно запускали свои задачи на трансляцию и выполнение. Тем не менее программы работали и опыт рос.

В «Кванте», да и в других журналах, больше Заочных школ не было, и мы занимались с тех пор, в основном, в ЛИАПе. Для многих из нас эти занятия стали профессией: мы поступили в ЛИАП на ту же кафедру 44 по специальности 0608 (ныне 2201).

Тем временем получилась дружная активная компания школьников-специалистов-преподавателей: Олег Архангельский, Сергей Фризюк, Сергей Гавриленко, Вера Корчагина, Светлана Иванова, Анна Обловацкая и другие, вместе с автором этих строк.

Важно, что мы очень скоро стали сами преподавателями программирования: после небольших школьных кружков, часто «в безмашинном варианте», мы стали заниматься со многими группами параллельно, плановым, а чаще явочным порядком занимая учебные аудитории в институте и по очереди выходя на машины.

В ЛИАПовской школе было несколько попыток разработки учебного программного обеспечения, в частности, хотели сделать ту же Рапиру на М-6000/СМ-1. Увы, эти эксперименты по большей части не удались. Был Форт, было много игровых программ, активно разбирались с системным ПО. Но в то время поставить серьёзную разработку системного программного обеспечения учебного назначения, в общем, не удалось.

Были конференции, раз в год, на которых школьники рассказывали о выполненных работах. Пожалуй, не менее половины из них действительно были неплохо сделанными программами. Возможно, кто-нибудь когда-нибудь, подняв архивы конференций тех лет, сделает обзор и этого важного направления в учебной информатике. Начинать нужно будет с первой кон-

ференции в декабре 1981 года; пожалуй, наиболее интересными её участниками были акад. А.П. Ершов и М.М. Ботвинник.

Не менее интересной и полезной была работа и в других институтах и с другими учителями. Прежде всего нужно назвать Андрея Николаевича Терехова, ныне профессора, зав. кафедрой системного программирования матмеха СПбГУ и одновременно руководителя успешной фирмы. С ним мне довелось поработать особенно активно в Ленинграде, после возвращения с ЛШ-1982, и этот опыт был чрезвычайно важным для профессионального развития. Несколько моментов оказались ключевыми. Первым из них был такой. Мы с приятелем (М.В. Годиным) появляемся на матмехе, нам вручается толстенный том библиотеки реализации Рапиры на Алголе-68, и со словами: «Вы с этим разберитесь и исправьте», — Андрей Николаевич выделяет нам рабочее место, приглашает на студенческие лекции (а мы ещё только в восьмой класс перешли) и откланивается. Мы ещё и Алгола-68 тогда толком не знали, тем более, что его реализация на матмехе, похоже, создавалась параллельно с работой. Но после консультаций с Андреем Николаевичем разобрались и что-то исправили. Второй момент возник при разборе программ, написанных на Летней школе. «Вот, — говорит Терехов, — что написали школьники. Очень удобная конструкция, всё сведено в простую таблицу, я сам сразу не додумался.» У меня лёгкое потрясение: ведущий специалист признаёт, что школьники сделали лучше него. Может, Андрей Николаевич и преувеличивал, очень может быть, но эффект был достигнут: я поверил, что и сам смогу что-то сделать существенное, большое и полезное, раз такие же школьники, как я, это могут, и это подтверждает ведущий системный программист. Там работа шла, в основном, на ЕС-1033 (на «старом матмехе», на 10 линии, дом 33). И когда мне в 1984 году понадобилась машина для реализации Форта, я снова обратился к Андрею Николаевичу. Получив на три июньских выходных место на ЕС-ке в Петергофе и сделав свою программу, я тем же летом рассказывал о ней на Летней школе. Так, разными способами и в разных местах, мы осваивали профессию — тогда мы уже не сомневались, что это станет делом всей жизни, хотя и не формулировали это так возвышенно.

Конечно, школы продолжались и позже. Когда мы поступили в институты и университеты, непременной составляющей нашей работы и учёбы были занятия со школьниками в кружках и школах программирования.

Но это уже совсем другая история...

М. Ю. Колодин

ЛЕТНИЕ ШКОЛЫ ЮНЫХ ПРОГРАММИСТОВ: ЗАЧЕМ И КАК

В самолёте Ленинград—Новосибирск школьники переговариваются между собой: то, куда и зачем они летят, для них впервые. Мы закончили Заочную школу и год или два занимались программированием в ЛИАПе, но предстоящая Летняя школа в новосибирском Академгородке — явление для нас совершенно новое. Ко мне оборачивается «знаток»:

—Вот, к примеру, М-6000 знаешь?

— Да.

— СМ-1 помнишь?

— Да.

— Ну вот, совсем не похоже...

Женя Забокрицкий знает, о чём говорит: он уже был в Новосибирске, и эта поездка для него — вполне понятное дело.

Так оно и оказалось. Всё было непривычным. И сам Академгородок, и Летняя школа, и Университет, и ВЦ. Но — в высшей степени замечательным.

Это была VII ЛШ. Последняя из тех, что проводились в самом Академгородке. Жили мы в общежитиях Университета на улице Пирогова. Там же, на стадионе, была и зарядка — в те времена к этому относились достаточно строго. А комплексный обед — в столовой на Морском проспекте.

Занятия проходили в Университете. Все школьники были разделены на три потока. Первый из них составляли новички. Занятия и само программирование были ориентированы, в основном, на Паскаль, а также на Бейсики первых для нас персональных машин Apple и Olivetti. Кстати, даже там, где, казалось бы, было не избежать примитивности Бейсика, Геннадий Анатольевич Звенигородский нашёл способ сделать лекцию полезной и в некотором роде надъязыковой: мы изучали сразу два языка, два разных Бейсика одновременно. Доска (и тетрадь) была разделена на два столбца, и в каждом шло описание языка одной из этих машин, с указанием на сходства и различия реализаций. На этих машинах мы, в основном, рисовали. Паскаль, а также уже давно, по крайней мере, теоретически, известная нам Рапира были на БЭСМ-6. Вместе с Рапирой была Шпага («школьный пакет графических процедур, адаптированный») — вкратце описанная в квантовских уроках графическая библиотека. Некоторые рисунки школьников были вполне удачными, прежде всего, те, в которых активно использовалась ма-

тематика, полярная система координат; некоторые обычные рисунки также смотрелись весьма мило. Первый поток вела, в основном, Нина Ароновна Юнерман. Второй и третий потоки на той ЛШ вели, соответственно, Геннадий Анатольевич Звенигородский из ВЦ СО АН СССР и Андрей Николаевич Терехов с матмеха ленинградского Университета. Они занимались реализацией языка Рапира, соответственно, на Агатах и на ЕС ЭВМ.

Работа с машинами была только на ВЦ. Самым страшным наказанием для нас было: «Оставим без машинного времени». Тогда Вычислительный центр был для меня невероятно большим зданием. Заблудиться в нём было проще простого. Приходилось консультантам, которые с нами работали, не только заниматься машинными делами, но и элементарно разводить нас по терминальным комнатам («терминалкам»). Консультанты — новосибирские школьники, которые сами относительно недавно закончили обучение, но уже имели немалый опыт практической работы и на ВЦ были полностью как свои.

Был Оргкомитет — А. Ершов, Г. Звенигородский, Ю. Первин, Н. Юнерман. Входили в рабочую группу Оргкомитета и школьники. Прежде всего, это Н. Глаголева, Л. Бараз, В. Цикоза, П. Земцов, Е. Налимов, А. Буднева, Е. Елинер, И. Мавлютов, Е. Краштан, Е. Музыченко, Е. Каленкович, А. Филатова, А. Ивания, Е. Боровиков, Л. Рабинович, А. Петров, С. Гавриленко, С. Терехов, Н. Погосян. В числе преподавателей ряда Школ были, кроме того, Н. Бровин, А. Терехов, А. Филиппов, А. Кривцов, О. Титов, А. Берс. В разработке ПО участвовали таже М. Зайцев, А. Грабарь и другие. На каждую ЛШ специально приглашались ведущие специалисты, прежде всего, по информатике для проведения семинаров, чтения лекций. Не забудем и тех, кто обеспечивал жизнь всех участников школы: например, врача.

Традицией тех и, частично, современных Школ были конференции. Обычно их было две — в начале и в конце Школы. На первой школьники рассказывали о своих работах, выполненных до ЛШ, дома, на второй — о том, что было сделано за время самой ЛШ. Надо сказать, что учёба и работа были весьма непростыми: за 2 недели нужно было вжиться в новую среду, учебную и человеческую, изучить некое новое средство, язык, машину, технологию, и сделать на них нечто полезное, по крайней мере, значимое на уровне школьника. Уже тогда эти конференции моделировали настоящие научные собрания. Часто возникали обсуждения, целые дискуссии. Даже простые задачи могли вызвать вопросы. Например, на той ЛШ несколько человек, в том числе и я, сделали программы определения дня недели по году, месяцу и числу — задача традиционная, но каждым она ко-

гда-то решается впервые. Так было и у нас. Я тогда заметил, что из-за смены дат в России с 1 на 14 февраля 1918 года на 13 дней, т.е. на неполное число недель, расчёт дней недели до 14.02.1918 необходимо уточнять с учётом этого факта. Мне кто-то возразил, началось обсуждение, и мне, впервые в жизни, пришлось отстаивать свою правоту в огромном зале — дело происходило в Институте геологии. Это было важное событие и большой опыт. И так оно было для многих. Плюс использование эпидиаскопа и указки — всё по-серьёзному.

Замечательной была и традиция костров и песен под гитару. А.Н. Терехов хорошо знал Битлов и умело это демонстрировал. Там же, на Пирогова, в лесочке, был устроен костёр, и песни и рассказы звучали допоздна. Уже тогда начало собираться «литературное обеспечение ЭВМ» («ЛО ЭВМ»).

Впрочем, значительно большее развитие ЛО ЭВМ получило на VIII и IX ЛШ, которые проходили в лагере «Сибиряк» бюро международного молодёжного туризма «Спутник», между Академгородком и Бердском. Там мы ставили «полнометражные» компьютерные оперетты, например, «Прекрасная Ада» — с ариями под гитару, костюмированным представлением на сцене, где, скажем, в роли барона Фортрана был Л.Ф.Штернберг, преподаватель Школы и мастер «Фразеологии программирования» (сейчас бы мы, скорее, говорили бы о технологии). Жизнь «в лесу» этому романтическому настрою весьма способствовала.

Хочется отметить важный, не только технический, момент. Тогда на ЛШ только появились «Агаты». Их было всего несколько штук. Надёжность техники оставляла желать лучшего, и только самоотверженными усилиями специалистов по вычислительной технике О. Титова, А. Кривцова, А. Филиппова и других энтузиастов её удавалось оживлять и использовать практически круглосуточно. Машинного времени всё равно не хватало. И тем не менее, сотня школьников успевала даже на паре персональных компьютеров сделать очень многое. Сейчас, мне кажется, при значительно лучших технических условиях, такая производительность не достигается.

Важно, что с самого начала школьники привлекались к созданию школьных же программ, к разработке системного программного обеспечения — операционной системы, редакторов, трансляторов. Это дало настолько большой профессиональный опыт, что многие из нас до сих пор вспоминают его с благодарностью и пользуются освоенными тогда приёмами.

Основная форма проведения большинства Летних школ — мастерские (при Г.А. Звенигородском они ещё так не назывались, но фактически смысл

был тот же). Они заключались в том, что мастер — специалист, который может и хочет вести некоторую тему, — предлагает её школьникам для изучения и разработки соответствующих программ. В начале Школы обычно проходит целая презентация мастерских. Школьники выбирают наиболее интересующую их тему и приходят к мастеру. В течение Школы они вместе занимаются и на выходе имеют программу, о которой докладывают на заключительной конференции. Большинство мастерских следует признать удачными. Тематика мастерских — самая различная: от простых учебных программ начального уровня до весьма сложных системных разработок (языки, операционные системы, редакторы), много было и игровых программ (сами и писали, и играли). Мои школьники справлялись со многими сложными вещами, например, реализациями языков Форт (на Ассемблере), Бейсик и Оккам (на Форте) и др. За время Школы успевали сделать наиболее существенные части работ, не всегда доводившиеся до полного совершенства. Иногда эти работы завершались участниками дома, после возвращения с Летней школы, или в самом Новосибирске. В последнее время большой интерес проявляется к графическим, сетевым, низкоуровневым и веб-приложениям. Игры были в почёте всегда; но, по-моему, не стоит увлекаться на Школах компьютерными игрушками, можно только их делать.

Иногда мастерские заявляются как студии или превращаются в таковые по ходу занятий, что означает чисто учебный характер мастерской, без обязательной выдачи программного продукта. В принципе, количество студий следует сокращать — они ведут к слишком высокой пассивности участников — и оставлять их для младших участников или новичков, которые ещё не умеют самостоятельно писать программы и только приобщаются к полноценной работе с компьютером. Например, студия «пресс-центр»: в ней могут участвовать самые младшие школьники и в сотрудничестве с другими мастерскими выдавать свой результат для всей Школы. Для них коллективная работа будет полезна и в рамках студии. Неудачной была попытка участия школьников сразу в нескольких мастерских; вопреки ожиданиям и обещаниям, это фактически приводило к неучастию ни в одной. В целом же идея мастерских очень удачна и жизнеспособна; её нужно развивать и далее.

Впоследствии летние школы перебрались в Политехникум (он же — Высший колледж информатики) у Шлюза (на Русской улице). С одной стороны, это было удобно: и жильё, и большая часть техники, и столовая, и конференц-зал находились рядом, и природа была в порядке... но что-то там было уже не то. У Политехникума была своя задача — привлечение

новых абитуриентов, и всё обучение, все постановки задач делались с упором на это. Увы, цели и лучшие традиции прежних ЛШ при этом были забыты. После нескольких лет существования школ там их уровень существенно понизился и считать их продолжением наших ЛШ было уже нельзя; хотя как школы колледжа они вполне уместны.

На несколько лет ЛШ ушли в пассивное существование: о них помнили, их продолжения многие хотели, но выполнить это в середине 90-х годов не удалось. И только на грани нового тысячелетия было решено и практически поддержано возобновить Летние школы — на старых традициях, но с новой технической (аппаратной, программной) и методической базами. И это получилось — первая новая школа прошла с большим успехом на Семинском перевале на Алтае летом 2001 года. Это ещё не история, и писать о ней, наверное, рано. Но самых тёплых слов заслуживают те, кто поднял всё это сложное дело, прежде всего, А.Г. Марчук, Л.В. Городняя, Т.И. Тихонова, А.А. Берс и другие. Как главный мастер той Школы должен подтвердить, что они сделали очень большую и важную работу. Не меньшей была и работа студентов, которые обеспечили возможность проведения Школы технически и сами проводили занятия. Появились и новые (и вернулись старые) преподаватели, которые вели мастерские и читали лекции. Вторая школа была в 2002 году в обновлённом, вернее, почти полностью развалённом за годы перестройки «Сибиряке». Сложности в проведении были, но опять же с ними справились; Школы могут преодолеть любые технические трудности, но не идейные и организационные.

Фактически Летние школы прошли несколько этапов; даже места проведения школ в какой-то мере отражают эти этапы: первые, экспериментальные Школы; полномасштабные Школы в Академгородке и в «Сибиряке» при жизни Геннадия Анатольевича Звенигородского; Школы там же после его смерти, «школы учеников»; Школы в Политехникуме; возрождённые Школы. Каждый этап был чем-то значим, в каждой из этих Школ учились и жили люди, множество школьников, студентов, преподавателей, организаторов — для всех нас это важнейшая часть нашей жизни.

Сейчас очень важно, чтобы и об этих Летних школах мы не стали вспоминать уже завтра как о давно прошедшей истории. Их необходимо продолжать. То, что для этого нужно, уже известно и подробно описано. Самое главное — команда единомышленников с большой светлой идеей. Если это будет — будут и новые успешные школы.

Автор благодарит О. Архангельского и С. Гавриленко за помощь в подготовке материала.

Г.А. Сапрыкина

РАЗВИТИЕ ПРОГРАММНЫХ СРЕДСТВ ОБУЧЕНИЯ ПО МЕРЕ ОБНОВЛЕНИЯ ШКОЛЬНОЙ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

С появлением вычислительной техники в школе стали разрабатываться программные средства учебного назначения. В работе рассматривается динамика развития и совершенствования программных педагогических средств, создаваемых для применения в школьном образовательном процессе, в хронологической связи с разными типами школьной вычислительной техники.

Информатизация нашего общества как элемента международного сообщества явилась неизбежным историческим процессом. Этот процесс предусматривает использование средств коммуникации и вычислительной техники во всех сферах человеческой деятельности. Технической основой процесса информатизации общества является вычислительная техника. Для ее использования необходимы определенные навыки, умения, мышление, иначе говоря, необходима компьютерная грамотность. 80-е годы стали десятилетием технологической революции в образовании во всем мире [1].

Для обучения компьютерной грамотности необходимы были педагогические программные средства (ППС), представляющие собой программы разной сложности, структуры и назначения и методические рекомендации для их использования.

Термин “педагогические программные средства” выбран для обозначения программного продукта учебного назначения, которое может быть создано и освоено с помощью компьютера. При этом следует помнить, что в 80-е годы вычислительная техника, на которой использовались ППС в школе, имела небольшую оперативную память. Поэтому ППС могли создаваться только небольшими по объему. В этой связи они предназначались для решения определенных, достаточно узких учебных задач. Подобные ППС классифицировались и типологизировались по разным критериям и признакам. Так, Т.А. Сергеева и Т.А. Невуева [2] ввели типологизацию ППС по следующим признакам:

1) по предметному содержанию программ (тематический принцип) — математика, физика, история и т.д.;

2) по функции (принцип целевого назначения): диагностические, контролирующие (текущий и итоговый контроль, регистрация и хранение све-

дений об успеваемости), обучающие (демонстрационные, тренажеры, справочно-информационные);

3) по степени активности учащихся, определяемой структурой и характером деятельности (демонстрационные, конструирующие программы);

4) по уровню коммуникативности (предметно и коммуникативно-ориентированные — сетевая коммуникация);

5) по целевой группе пользователей — инструментальные педагогические средства (базы данных, редакторы, компьютерные журналы и конспекты).

Современные компьютеры позволяют создавать ППС, которые включают сразу все первые четыре типа. Такие ППС в настоящее время называют электронными учебниками (ЭУ).

Классифицировались ППС и по другим критериям: по цели конкретной разработки, ее внутренней структуре и содержанию. Такой подход исходит из точки зрения пользователя. К ППС с точки зрения пользователя можно отнести:

- программные средства (ПС) — отдельные компьютерные программы с методическими рекомендациями или инструкциями для пользователя;
- компьютерные (или компьютеризированные) курсы (КК) — предметно-ориентированные ППС, охватывающие тему или раздел изучаемого предмета;
- компьютерные учебные пособия (КУП) — предметно-ориентированные и интегрированные компьютерные курсы с методическими рекомендациями для пользователя;
- программно-методические комплексы (ПМК) — совокупность ПС, КК, КУП, объединенных в комплекс тематическим планом для достижения общей цели и методическое пособие для работы с ними. Иерархия ППС хорошо прослеживается на рис. 1.

Если соотнести последнюю классификацию ППС с указанной выше типологизацией, то ПС определяются принципом целевого назначения (пункт 2); КК определяется предметным содержанием (пункт 1); КУП определяются обоими принципами. Кроме того, КУП по физике, например, может содержать вспомогательный КК по математике, КК по истории, являясь интегрированным курсом. ПМК могут включать все виды и типы ПС, КК, КУП.

<table>
<tr><td colspan="6">ПМК</td></tr>
<tr><td colspan="3">КУП$_1$</td><td colspan="3">КУП$_2$</td></tr>
<tr><td colspan="2">КК$_1$</td><td colspan="2">КК$_2$</td><td colspan="2">КК$_3$</td></tr>
<tr><td>ПС$_1$</td><td>ПС$_2$</td><td>ПС$_3$</td><td>ПС$_4$</td><td>ПС$_5$</td><td>ПС$_6$</td></tr>
</table>

Рис.1

При знакомстве с конкретными разработками по каталогам [3-6] становится ясно, что было разработано крайне мало ППС типа КУП по общеобразовательным предметам; фрагментарность, тематическая разрозненность мешали постоянному использованию их в учебном процессе. И главный недостаток многих имеющихся программ состоит в неправомерном переносе традиционных форм и методов обучения в компьютерную обучающую программу. Конечно, это следствие того, что потребности практики не были еще достаточно обеспечены соответствующими теоретическими проработками и отсутствовали квалифицированные специалисты-постановщики компьютерных курсов.

В начале процесса компьютеризации школ лейтмотивом деятельности образовательных учреждений в этом направлении была поставка в школы страны любой вычислительной техники. И в школы попала разнообразная ВТ: БК-0010, Корвет, УК-НЦ, Ямаха, Агат и многие другие виды. И в то время Сибирским институтом образовательной технологии Российской академии образования на первые четыре из указанных типов школьных компьютеров был разработан программно-методический комплекс для подготовки руководящих и педагогических кадров в области информатики (36-часовой курс). Прочие разработки относились к одному виду техники.

С началом школьной реформы нового тысячелетия в политике информатизации школ произошел качественный скачок. В настоящее время не выпускается школьная вычислительная техника. В школы поставляются современные модели персональных компьютеров. Стремительный процесс информатизации школ на основе современных компьютеров, поступающих в учебные заведения страны, открывает в образовании путь электронным учебникам. Этот термин в настоящее время наиболее устойчив, и к этому типу разработок относятся все в большей или меньшей степени целостные компьютерные курсы учебного назначения.

Для обеспечения многофункциональности при использовании и в зависимости от целей разработки электронные учебники могут иметь различную структуру. Например, для использования на уроках можно создавать электронный учебник, поддерживающий школьную программу по конкретному предмету, и учебный материал подавать согласно имеющемуся тематическому планированию. Можно разрабатывать электронный учебник без привязки к тематическому планированию, а просто следуя учебному плану по конкретному школьному курсу. Можно создавать электронные учебники по принципу вертикального изучения учебного материала. К примеру, функции и графики изучаются в школе с 7 по 11 классы. На бумажных носителях существует четыре учебника для соответствующих классов, в каждом из которых наряду с другими темами есть и учебный материал по функциям и графикам. Электронный учебник может объединить весь изучаемый материал по этой теме с 7 по 11 классы. Такой ЭУ можно использовать и для самостоятельных занятий, для подготовки к сдаче экзаменов, на уроках, для подготовки к сдаче курса экстерном.

В настоящее время издано много печатных учебников по разным школьным предметам. Но тем не менее, некоторые авторы отмечают их низкий уровень по отдельным предметам. Так, по мнению В.К. Совайленко (учитель математики) [9], в школе нет хороших учебников по математике. По химии также ведутся занятия по учебникам, не нашедшим положительной оценки у учителей [10]. Говоря о качестве школьных учебников, ректор МГТУ им. Баумана И.Б. Федоров [11] отмечает: «В отечественных школьных учебниках по математике и физике сквозь текст зачастую приходится “продираться”. При изложении материала, как правило, идут не от смысла, а от формы…». Поэтому оправдан постоянный поиск новых форм организации учебного материала в учебниках.

Отправной точкой в создании электронных учебников являются дидактические цели и задачи, для достижения и решения которых используются информационные технологии.

В зависимости от целей обучения электронные учебники могут быть следующих типов:

- предметно-ориентированные ЭУ;
- предметно-ориентированные электронные учебники для изучения отдельных разделов предметов общеобразовательного цикла при сквозном изучении учебного материала;
- предметно-ориентированные электронные тренажеры с наличием справочного учебного материала;

- электронные автоматизированные системы развития способностей.

Предметно-ориентированные ЭУ находят широкое применение в настоящее время в школах. При разработке электронного учебника необходимо первоначально выработать его строение, порядок следования учебного материала, сделать выбор основного опорного пункта будущего учебника. Рассмотрим, к примеру, электронный учебник «Школьный физический эксперимент» для общеобразовательной школы со сквозным изучением учебного материала [12]. Такой ЭУ состоит из следующих разделов школьного курса физики :

1) механика (7, 9 классы),
2) электромагнитные явления (8,10-11 классы),
3) молекулярная физика (7-10 классы).

Каждый раздел такого электронного курса по физике состоит из следующих компонентов:

- моделирование физических экспериментов для исследования и демонстрации физических законов, явлений;
- познавательный материал по разделам физики;
- задания, тесты для закрепления и контроля усвоения знаний;
- терминологический словарь или справочник физика;
- математический аппарат, необходимый для усвоения отдельных тем курса;
- историческая справка об открытии и исследовании конкретного физического явления.

Все разделы курса и их компоненты взаимосвязаны, находятся в общей программной оболочке. Каждый компонент в указанных разделах электронного учебника доступен для пользователя из любого другого компонента.

Изучение указанных тем из школьного курса физики ведется в нескольких классах по спирали, начиная с 7-го или 8-го. Затем многие темы в более развернутом виде изучаются в следующих классах: механика — в 9-м, молекулярная физика — в 8-м, электродинамика — в 10-м. И завершается изучение в 10-м или 11-м классах. Таким учебником, размещенным на одном компакт -диске, можно пользоваться с 7-го по 11-й классы.

СПИСОК ЛИТЕРАТУРЫ

1. Пелгрюм В.Й. Международные исследования в компьютеризации образования // Перспективы. — 1993(83). — № 3. — С. 100–110.

2. Сергеева Т.А., Невуева Т.А. Рекомендации по проектированию педагогических программных средств. М.: НИИ ШОТСО АПН СССР. — 1990. — 50 с.
3. Каталог программных средств для ПЭВМ, поставляемых в 1990 г. Казань. — 1990. — 65 с.
4. Каталог фонда алгоритмов и программ. Республиканский центр новых информационных технологий обучения. Омск. — 1990. — 81 с.
5. Каталог РОСЦИО компьютерных учебных программ. —М., 1992. — № 1. — 56 с.
6. Каталог РОСЦИО компьютерных учебных программ. — М., 1993. — № 1(2). — 85 с.
7. Советский энциклопедический словарь. — М.: «Советская энциклопедия», 1985.
8. Ретинская И.В., Шугрина М.В. IBM и Makintosh в сфере образования // Мир ПК. — 1994. — № 3.
9. Совайленко В.К. О содержании математического образования и качестве учебников (мнение учителя) // Педагогика. — М.: Педагогика, 2002. — № 3. — С. 35–39.
10. Материалы круглого стола Дистанционного научно-методического объединения учителей химии на сайте НООС www.websib.ru
11. Федоров И. Главное условие качественного образования — наука, но никак не наукообразие // Школьное обозрение. — М., 2002. — № 1. — С. 42–43.
12. Карпушова И.Б., Сапрыкина Г.А., Старцева Н.А. Технология разработки КУП по физике. Труды телеконференции 2000–2001 г. Новосибирск. СИОТ РАО. — 2002. — 328 с.

СОДЕРЖАНИЕ

НОВОСИБИРСКАЯ ШКОЛА ПРОГРАММИРОВАНИЯ.

Перекличка времен

Под редакцией
проф. И. В. Поттосина,
к.ф.-м.н. Л. В. Городней

Рукопись поступила в редакцию 27.09.04
Редактор З. В. Скок

Подписано в печать 30.12.04
Формат бумаги 60 × 84 1/16 Объем 13.8 уч.-изд.л., 15.1 п.л.
Тираж 100 экз.

ЗАО РИЦ «Прайс-курьер»
630090, г. Новосибирск, пр. Акад. Лаврентьева, 6, тел. (383-2) 30-72-02