МЕТОДЫ АНАЛИЗА ДАННЫХ
(Вычислительные системы)
1985 год Выпуск III

УДК 519.685.I

ПРОБЛЕМНО-ИНСТРУМЕНТАЛЬНАЯ ТЕХНОЛОГИЯ ПОСТРОЕНИЯ ПРОГРАММНЫХ СИСТЕМ

Ю.Г.Косарев, А.А.Москвитин

I. Введение. Успехи в науке и технике и, прежде всего, в микроэлектронике позволили приступить к массовому выпуску дешевых, малогабаритных, надежных и быстродействующих ЭВМ. Появилась возможность обеспечить каждого желающего персональным компьютером, способным решать многообразные, в том числе и достаточно сложные, задачи. Данная ситуация наступила так внезапно, что возник разрыв между необходимой и фактической подготовкой основной массы потенциальных пользователей.

Традиционно от пользователя требуется не только знание возможностей ЭВМ, круга решаемых ею задач но и умение нормально записать процесс решения своей задачи на одном из алгебраических языков. То есть пользователю нужно знать не только ЧТО он хочет, но и уметь точно описать КАК желаемое ЧТО должно реализоваться. Это КАК и является главной преградой на пути массового применения ЭВМ.

Подавляющее большинство пользователей относится к так называемым "непрограммирующим профессионалам" [I]. Именно они, представители всех сфер человеческой деятельности, и являются поставщиками задач для ЭВМ. Их часто именуют "конечными пользователями" [2] (чтобы отличить от программистов, выполняющих роль посредников между ними и ЭВМ). Естественно, что когда конечным пользователем станет каждый желающий, то никакой армии программистов-посредников не хватит. Заманчиво, конечно, обучить всех программированию, возведя его в ранг "второй грамотности". Но этот путь вряд ли приведет к успеху в обозримом будущем, поскольку он возлагает основную тяжесть на сами многомиллионные массы конечных пользователей.

Более рациональным представляется переложить эту ношу на профессиональных программистов, поставив перед ними задачу создания таких средств общения с ЭВМ, которыми мог бы легко овладеть любой пользователь, не знакомый с программированием (но знающий, конечно, что именно может дать применение ЭВМ в его профессиональной деятельности).

Основная цель предлагаемой работы - показать реальность этого второго пути на примере конкретной технологии построения проблемно-ориентированных программных систем, рассчитанных на непосредственное взаимодействие с конечным пользователем, не владеющим программированием.

Предлагаемая технология направлена на создание программных систем, ориентированных на информационное обслуживание работ, связанных с циркуляцией потоков символьной информации со сложной логической структурой (редакционно-издательская деятельность, информационно-справочные службы, деловая переписка, служба русского языка, управляющие части пакетов прикладных программ, интеллектуальные базы данных и знаний и т.п.). Эти системы должны пропускать через себя большие информационные потоки, преобразовывать, упорядочивать и хранить массивы данных, выявлять и накапливать обнаруженные закономерности и т.д. Как средства, работа которых имеет четко выраженный индустриальный характер, эти системы должны быть доступны широкому кругу специалистов из обслуживаемой области без каких-либо посредников и обладать рентабельностью, под которой понимается достаточно длительный жизненный цикл [3], а также довольно полное использование технических средств, и, как следствие, большая гибкость структуры, обеспечивающая легкость совершенствования системы по мере возникновения более эффективных методов и технических средств, видоизменения и расширения круга решаемых задач.

Короче, речь идет о создании программных систем, которые служили бы рабочим инструментом конечных пользователей, для информационного обеспечения соответствующих проблемных областей.

Для определенности назовем такие системы и технологию их построения ПИ-системами и ПИ-технологией (чем подчеркивается их проблемная ориентация и инструментальный характер)*).

---

*) Далее в статье для краткости - к-система, к-технология и т.п.

К π-системам и π-технологии предъявляются качественно новые требования, с которыми ранее не встречалась теория и практика программирования. Естественно пытаться справиться с такой сложнейшей задачей на пути следования общим принципам построения много - элементных программных систем [4].

В согласии с этими принципами в основу π-технологии была положена идея базовой системы - прародительницы, в недрах которой рождаются π-системы для конкретных областей применения. Система-прародительница по своим свойствам это также π-система. Создаваемые ею дочерние π-системы полностью наследуют ее структуру и получают "в качестве приданного" ту часть содержательных свойств, которых им должно хватить на определенный период самостоятельного существования.

Ясно, что для выполнения своего предназначения π-система-прародительница должна обладать способностью накапливать опыт создания и совершенствования всех дочерних π-систем в удобной для использования форме, а также уметь применять этот опыт для синтеза вновь создаваемых π-систем.

Иными словами, в π-системе-прародительнице должны сочетаться свойства как экспертных, так и технологических систем.

2. Функциональные массивы π-систем. Структурные и некоторые содержательные свойства (типовые) π-систем являются слепком с π-системы-прародительницы, поэтому все, что далее специально не оговаривается, будет относиться как к прародительнице, так и к любой из π-систем.

Употребляемые далее обозначения структурных элементов π-систем и данных собраны в табл.I.

Все π-системы состоят из пяти систем: анализа и обработки данных, диалоговой, синтеза управляющих программ, операционной и интеллектуальных баз данных и знаний.

Все системы одинаковы как по структуре, так и по составу образующих их функциональных массивов. Этих массивов также пять: массив модулей, управляющий, конфигураций, состояний и дескриптор системы.

2.I. Массив модулей состоит из специальным образом организованных программных модулей. Из тела этих модулей вынесены все переменные величины, определяющие свойства циклов, режимы работы модуля, условные переходы и модули-преемники, привязку к данным, тип, структуру, размеры этих данных и т.п. Эти величины заменены адресами соответствующих полей управляющего массива.

Таблица I

Структурные элементы данных π-систем

Проблемная область	Все объекты проблемной области	Набор объектов	Объект	Свойство объекта	Конкретное значение свойства
Представления конечного пользователя	Все данные о проблемной области	Данные о наборе объектов	Данные о свойствах объекта	Атрибут	Значение атрибута
Представления проблемного программиста	Набор файлов	Файл	Запись	Элемент	Значение элемента
Представления системного программиста	База данных	Набор данных	Запись	Элемент	Значение элемента

Структурные элементы π-систем

π-система-прародительница- π-система - подсистема - массив - элемент - поле
\ S-модуль - переменная величина

Такие модули-фреймы или скелетные модули далее называются S-модулями. S-модули бывают двух основных типов: полностью структурированные (SM-модули), из тела которых удалены переменные величины и которые пишутся на машинно-ориентированных языках программирования, и частично структурированные (SP-модули), которые пишутся на проблемно-ориентированных языках. В них могут остаться те переменные, вынесение которых затруднительно при применяемых средствах программирования или при конкретной реализации, когда, например, S-модуль создается путем адаптации готовой программы.

Применение SP-модулей может существенно ускорить создание или реконструкцию π-систем. Сравнительно просто создавать SP-модули из программ, написанных на ФОРТРАНЕ. Их адаптация аналогична вынесению всех передаваемых параметров в COMMON-блок. Именно так создавалась первая версия π-системы ЛАДА [5]. Естественно, что применение SP-модулей там, где это может существенно повлиять на качество π-системы, может допускаться лишь как временная мера. Все ответственные S-модули желательно строить как SM-модули. Это, конечно, требует от программиста бо́льших усилий и более высокой квалификации, но окупается эффективностью и гибкостью как исходных SM-модулей, так и SM-модулей более высокого порядка, получаемых из имеющихся SM-модулей путем их объединения по определенной методике [4].

2.2. Управляющий массив представляет собой совокупность элементов, каждый из которых взаимно-однозначно связан с одним из S-модулей массива модулей. Элементы состоят из полей, куда могут заноситься значения переменных, вынесенных из соответствующего S-модуля. Число полей каждого из элементов совпадает с числом переменных величин в соответствующем S-модуле (заметим, что каждая переменная может использоваться в S-модуле неоднократно).

Количество и состав полей элементов управляющего массива указываются в дескрипторах этих элементов. Кроме того, там же помещаются имя соответствующего S-модуля, тип дескриптора и два адреса: S-модуля и первого поля элемента.

Повторяющиеся варианты заполнения всех или какой-либо части полей управляющего массива накапливаются в базе данных в виде специального массива трафаретов. Использование трафаретов аналогично применению стандартных процедур в традиционных методах программирования.

2.3. Массив конфигураций устанавливает последовательность работы S-модулей. Массив состоит из дескриптора и двух последовательностей полей: текущей, которая управляет работой к-системы в данный момент, и формируемой в процессе работы, которая может использоваться на следующем этапе или в дочерней к-системе (в случае к-системы-прародительницы). Каждое из полей содержит: имя S-модуля, адрес соответствующего ему элемента управляющего массива, а если выбор S-модулей-преемников неоднозначен, то еще и условие и адреса полей массива конфигураций, соответствующих выполнению и невыполнению условия.

Заполнение полей массива конфигураций аналогично по своим функциям составлению головной программы при использовании процедур. По своему исполнению такой способ программирования достаточно прост и может осуществляться автоматически системой синтеза. Здесь так же, как и для управляющего массива, могут широко использоваться трафареты.

Включение в массив конфигураций, кроме текущей, еще и формируемой конфигурации позволяет предлагать пользователю некоторые виды видоизменения в постановке задачи в случае неуспеха или неэффективности прежней. Это вполне оправдало себя в к-системе ЛАДА [5].

2.4. Массив состояний фиксирует все характеризующие ход процесса работы системы параметры: имя работающего в данный момент S-модуля, текущие значения переменных циклов, содержимое стеков, регистров и т.п. Состав и структура соответствующих полей массива состояний указывается в его дескрипторе.

Выделение изменяющихся в ходе работы параметров в отдельный массив облегчает отладку, слежение за процессом вычислений, его разбивку на кванты, прерывание и возобновление счета и т.п.

2.5. В дескрипторе системы сосредоточены адреса входов в дескрипторы и рабочие части остальных массивов, все используемые внешние адреса, а также все необходимые сведения о структуре системы.

3. Свойства систем. По своему назначение структурные элементы к-систем (системы, массивы, модули и т.п.) можно подразделить на функциональные (проблемно-ориентированные) и на операционные (инструментально-ориентированные). Функциональные элементы определяют индивидуальные особенности данной к-системы, а операцион-

ные отражают особенности π-технологии. Они составляют часть функциональных элементов π-системы-прародительницы, наследуемую всеми дочерними π-системами. В этом смысле они являются типовыми.

Функциональные различия систем проявляются прежде всего в составе S-модулей, а также в составе и особенностях используемых массивов.

3.1. Система анализа и обработки данных предназначена для реализации алгоритмов решения задач из данной проблемной области. Ее основу составляет массив функциональных S-модулей. Могут использоваться S-модули обоих типов (SM- и SP-), что позволяет сочетать эффективность и гибкость π-систем с экономичностью их создания и реконструкции.

Это основная система, роль остальных систем сводится к обеспечению тех или иных сторон ее функционирования.

Главным содержанием работ по созданию новой π-системы и является такое расширение возможностей данной системы у π-системы-прародительницы, которое позволяет реализовать все заданные функции вновь создаваемой π-системы. Далее остается только выделить все элементы, участвующие в этой реализации, и сформировать из них систему анализа и обработки данных дочерней π-системы.

3.2. Диалоговая система обеспечивает возможность непосредственного общения с ЭВМ: конечному пользователю на его профессиональном языке - подмножестве естественного языка, программистам - на алгоритмических языках, которыми располагает операционная система.

От диалоговой системы требуется организовать активное и исчерпывающее полное получение от пользователя сведений о его задачах и затем представить эти сведения в формальном виде, исходном для системы синтеза управляющих программ. В ходе диалога должны быть выявлены свойства данных, постановки задач, характеристики промежуточных и конечных результатов.

Кроме того, на диалоговую систему возлагается обучение конечного пользователя правилам ведения диалога и ознакомление его с возможностями данной π-системы, а также ведение служебного диалога с программистами для настройки самой диалоговой системы на определенный тип диалога и входной язык пользователя.

Диалоговая система π-системы-прародительницы содержит достаточно представительный набор S-модулей, позволяющий реализовать все пять наиболее известных типов диалога: команды, подказ-

ки, меню, заполнение бланка, вопросы с ответами "да/нет" [2]; и настраиваться на широкий спектр входных языков - подмножеств естественного языка.

3.3. Система синтеза управляющих программ настраивает к-систему на выполнение конкретных функций.

Исходную информацию для синтеза поставляет диалоговая система, которая настраивается первой. Для этого в ходе служебного диалога к-системе задается тип диалога и входной язык конечного пользователя.

Для настройки на желаемый тип диалога достаточно ввести в массив конфигураций диалоговой системы соответствующий трафарет. Эти трафареты стандартны и содержатся в массиве трафаретов базы данных.

Для настройки на конкретный язык конечного пользователя необходимо задать словарь этого языка и список соответствий между конструкциями входного языка и стандартными S-модулями организации диалога. Система синтеза строит по этим данным массив-транслятор и засылает его имя и адрес в соответствующие поля управляющего массива диалоговой системы.

У дочерних к-систем функциональные части систем баз данных, операционной и самой системы синтеза пока задаются жестко самой к-технологией. Но в их структуре заложена принципиальная возможность такой настройки. В случае возникновения необходимости в их проблемной или машинной ориентации к ним могут быть применены те же методы синтеза, что и для остальных двух систем.

Основное назначение системы синтеза - настройка функциональной части системы анализа и обработки данных на задачи конечного пользователя.

Диалоговая система поставляет системе синтеза два рода сведений о задачах: характеристики исходных данных и характеристики результирующих данных. По этим характеристикам система синтеза строит управляющую программу, помещает ее в массив конфигураций и настраивает S-модули на данные, занося их параметры в управляющий массив.

Этот процесс заполнения полей управляющего массива и массива конфигураций можно трактовать как специальный вид программирования для гипотетической вычислительной машины, состоящей из собственно ЭВМ и массива S-модулей. При заполнении массива конфигураций под структуру задачи программируется структура этой машины

(состав операций и очередность их выполнения). При заполнении управляющего массива под свойства данных программируются содержательные свойства этих операций.

Такой подход находится в полном согласии с принципом программной изменяемости свойств многоэлементных систем [6].

Более того, к числу переменных, выносимых из S-модулей, относятся и параметры циклов над данными. Это позволяет в согласии с [7] эффективно настраиваться на количество предоставляемых S-модулю процессоров и таким образом реализовывать гармоничные параллельные процессы вычислений [8].

Работа системы синтеза π-системы-прародительницы по своему характеру аналогична описанной выше и отличается только более широкими возможностями, предоставляемыми ей базой данных и общением с программистами и конструкторами π-технологии.

3.4. Операционная система обеспечивает остальным системам доступ к трансляторам, редакторам, отладчикам, загрузчикам и т.п., как разработанным в рамках π-технологии, так и содержащимся в операционной системе ЭВМ.

В первых π-системах (ЛАДА и РЕДАКЦИЯ) используется операционная система ОС РВ СМ ЭВМ. Связь с нею осуществляется с помощью косвенных командных файлов, необходимый набор которых содержится в данной системе на правах S-модулей. Их настройка через управляющий массив и обеспечивает взаимодействие любой из систем π-системы с ОС ЭВМ.

Для программистов операционная система предоставляет (через служебный диалог) средства для создания S-модулей. Такими средствами служат π-язык и π-транслятор с этого языка.

По типу π-язык близок к Р-языку [9]. Программирование на нем ведется от данных. Выделяется логическая схема данных, и используется графическое представление программ. Но в отличие от РТК интерпретация заменена компиляцией программного продукта в виде S-модуля, типы памятей - специальными типами файлов, табличная организация синтермов - вложенной (группировка термов в синтермы указывается значениями заданных разрядов кода терма).

3.5. Интеллектуальные базы данных и знаний предназначены для информационного обеспечения всех этапов применения, создания и совершенствования π-системы и каждой из ее подсистем.

3.5.1. В интеллектуальной базе данных сосредоточиваются все используемые в π-системе виды информации:

- наборы исходных данных (их первоначальная и преобразованная форма, паспорта);

- служебные наборы (вынесенные списки, трафареты языковых конструкций, дескрипторов, формы таблиц, анкет и т.п.; таблицы перекодировок; таблицы функций и т.д.);

- библиотеки программ (типовые и функциональные S-модули, трафареты управляющих массивов и массивов конфигураций и т.п.);

- средства диалога (словари, команды, меню, подсказки, вопросы и т.д.);

- документация (описание π-системы, инструкции, стандарты, характеристики технических и программных средств вычислений, информация о реконструкциях π-системы, сведения о π-технологии в ее последней редакции, дневник π-системы и т.п.);

- архивы (пользователей, проблемных и системных программистов, загрузки π-системы, решенных задач и т.п.);

- базы знаний (результаты анализа и обработки данных, выявленные закономерности в логической, конструктивной и количественной формах; терминология, аксиоматика, гипотезы, состояние данной проблемной области, библиография и т.д.).

Предполагается, что в базе данных π-системы-прародительницы дублируется вся информация, содержащаяся в базах данных дочерних π-систем (кроме, конечно, личных архивов и другой текущей информации).

3.5.2. Система Управления Базой Данных (СУБД) выполняет все традиционные для реляционных баз данных функции: сбор, структуризацию, хранение, поиск информации, выявление и формальное представление отношений (реляций) между элементами данных, общение с пользователем (через систему диалога) и т.д. [10].

В основу организации баз данных π-систем положены следующие две идеи: применение вынесенных списков для обнаружения, представления и поиска отношений между элементами файлов и классификация файлов по степени однородности их структуры для определения типов допустимых над этими файлами операций [11,12].

Суть первой идеи состоит в том, что вся служебная информация о реляционных свойствах элементов помещается не в самих записях файлов, как обычно, а выносится в специальные таблицы (вынесенные списки).

Вынесенный список представляет собой упорядоченное по первому члену множество пар номеров мест записей до и после (или после

и до) упорядочения файла по заданному ключу. Первый член пары представлен адресом машинного слова относительно начала файла, а второй - содержимым этого слова.

Такая форма представления списковых структур по сравнению с традиционным размещением списковых адресов в самих записях файла обладает следующими преимуществами:

- экономией оперативной памяти. В нее можно вызывать не все, а только необходимые списковые таблицы и не обязательно всю запись файла, а лишь соответствующие ключевые слова (это особенно удобно для инвертированных файлов);

- увеличением числа списковых структур, в которых участвуют элементы файла. Это число определяется уже объемом не оперативной, а вспомогательной памяти;

- ускорением поиска места нужной записи. Его можно теперь искать с помощью дихотомии, а не последовательным перебором (т.е. при n записях за $\log_2 n$ шагов вместо $n/2$);

- сохранением в неизменном виде файлов при включении их в новую списковую структуру.

В согласии со второй идеей выделяются четыре основных типа файлов (табл.2), различающихся возможными типами операций над ними (табл.3):

- стековые и другие типы операций, последовательно выполняющиеся над записями файлов (вагонные, бобслей и т.п.), допустимы для всех типов файлов;

- регистровые (сборка из записей файла новой записи) допускаются только для файлов с записями одинакового размера;

- списковые, в том числе и операции над вынесенными списками, допускаются для файлов, у которых ключевые слова (и списковые адреса для невынесенных списков) находятся на строго фиксированных местах, одних и тех же у всех записей;

- ассоциативные, под которыми понимаются табличные (с прямой адресацией), хеширование, упорядочение с линейной сложностью от числа записей, дихотомический поиск и т.п.*) , возможны лишь для двух последних типов файлов и то для U2 лишь в случае применения вынесенных списков.

В этих таблицах "0" означает "нет", "1" - "да", "-" - "безразлично".

---

*) По этому поводу см., например, [13-15].

Таблица 2

Основные типы файлов	Обозначение	Свойства	
		Однородны структуры начал записей?	Записи, однородны по размерам?
Неопределенный	U0	-	-
Однородный по размерам	U1	-	I
Частично-однородный	U2	I	-
Однородный	U3	I	I

Таблица 3

Основные типы файлов	Допустимые типы операций			
	Стековые	Регистровые	Списковые	Ассоциативные
U0	I	0	0	0
U1	I	I	0	0
U2	I	0	I	I*)
U3	I	I	I	I

*) Только для вынесенных списков.

Тип файла указывается в его дескрипторе. Для файлов типа U2 и U3 указываются число возможных значений ключевого слова, применяемые способы упорядочения и поиска.

3.5.3. Заметную роль для π-систем и особенно для самих баз данных играют библиотеки S-модулей, в которых, кроме самих S-модулей, содержатся все необходимые сведения для синтеза соответствующих элементов управляющего массива и настройки S-модулей; и библиотеки управляющих трафаретов, где накапливаются варианты заполнения полей управляющего массива и массива конфигураций всех систем.

В этих библиотеках содержится "золотой" фонд π-технологии, определяющий свойства трех основных функциональных массивов (модулей, управляющего, конфигураций), а следовательно, и "лицо" каждой из систем и π-системы в целом.

Именно с этим фондом связано совершенствование интеллектуальных возможностей СУБД - одной из главных проблем современной информатики.

4. Организация работ в π-системах. В основе разработки, совершенствования и математической эксплуатации π-систем лежит идея четкого разделения функций между конечным пользователем и программистами: пользователь решает свои задачи, непосредственно общаясь с ЭВМ, а роль программистов заключается в предоставлении ему этой возможности.

Соответственно у π-системы имеются один пользовательский и три программистских уровня общения.

4.I. Уровень общения с конечным пользователем. Конечный пользователь является не только потребителем, но и активным участником разработки и совершенствования π-систем.

На начальной стадии разработки новой π-системы для конечного пользователя организуется диалог непосредственно с π-системой-прародительницей (либо с какой-либо из дочерних π-систем), в ходе которого устанавливаются взаимопонимание и вырабатывается диалоговый язык. Затем выясняются (и соизмеряются с возможностями) запросы пользователя как заказчика, а также круг допустимых видоизменений постановок его задач для выявления возможности их сведения к уже решенным. В конце концов выясняется, какие функциональные средства имеются в готовом виде, а на какие надо дать заказ программистам. Аналогичная ситуация возникает и при появлении у пользователя новой задачи, не учтенной при разработке данной π-системы.

В диалоге с пользователем по определенному сценарию извлекается информация, необходимая для четкой формулировки задачи. Определяются типы данных, подлежащих обработке. Формируются "разумные", с точки зрения пользователя, ограничения, и определяются предполагаемые свойства результатов.

По данным, извлеченным в ходе диалога с конечным пользователем, выделяется логическая схема данных, которая затем отображается в логическую схему программы, аналогично тому, как это делается в Р-технологии программирования [9]. Функциональная часть системы отделяется от ее структуры. При этом все выделенные параметры (различных типов) концентрируются в управляющем массиве π-системы. Процесс синтеза начинается с попытки свести задачу или метод ее решения к уже известным методам или ранее полученным решениям.

В том случае, когда в базе данных имеется аналог решаемой задачи, то при необходимости сменой содержимого управляющего мас-

сива изменяются некоторые или все параметры, составляющие функциональную часть системы. Искомую задачу можно считать запрограммированной.

Когда такого аналога нет, предпринимается попытка изменить содержимое массива конфигураций и тем самым изменить структуру имеющейся программной системы (при сохранении набора входящих в нее модулей и функциональной части системы).

Если эта попытка не приводит к успеху, то задача разбивается на подзадачи и для каждой подзадачи вновь ищутся готовые или сводящиеся к ним трансформацией логической схемы решения.

Если и в этом случае не удается добиться успеха, то конечный пользователь обращается к проблемному программисту.

4.2. Уровень проблемного программиста. Для каждой π-системы предусматривается главный проблемный программист, который является ее хозяином. Он создает новые модули по заказу конечного пользователя, устанавливает соответствия между конструкциями входного языка и схемами программных модулей. В его распоряжении имеется π-язык и соответствующий ему транслятор, позволяющий в конечном итоге создавать S-модули и соответствующее семантическое напол - нение управляющего массива и их объединение. Он проводит адапта - цию включенных в библиотеку готовых программных модулей (SP-модулей), в согласии с принципами π-технологии.

По заказу проблемного программиста, не сумевшего справиться с заказом пользователя имеющимися средствами, выдается задание следующему уровню разработчиков - системному программисту, который расширяет возможности π-системы, π-языка, π-технологии.

4.3. Уровень системного программиста. Главный системный программист является конструктором π-систем. Если проблемному программисту приходится работать в рамках средств, уже имеющихся в π-системах, то системный программист способен создавать средства их наращивания, модификации и т.п. Иначе говоря, он является поставщиком необходимых средств проблемному программисту. В случае необходимости он оформляет заказ конструктору π-технологии.

4.4. Уровень конструктора π-технологии. Встреченные "узкие" места в работе системного программиста могут быть расширены работой конструкторов π-технологии по выданному им заказу. Этим группам разработчиков доступны средства, позволяющие модифицировать или отвергать принятую идеологию в создании π-систем. Взамен

предлагается новая π-технология, в которой учтены недостатки предыдущей.

По предложенной схеме разделения труда между различными группами пользователей π-систем, конечный пользователь является главным в определении всех видов производимых в π-системах работ. По его заказам работают проблемные программисты, задания которых, в свою очередь, являются программой работ для системных программистов. Замыкают эту цепочку конструкторы π-технологии.

При таком распределении функций программисты различной квалификации являются уже не посредниками между конечным пользователем и ЭВМ, а разработчиками соответствующих их профессиональной подготовке инструментальных средств, необходимых для того, чтобы конечный пользователь мог сам непосредственно решать на ЭВМ свои профессиональные задачи.

Такая организация соответствует реальной ситуации с трудовыми ресурсами, так как предполагает, что проблемных программистов требуется на I-2 порядка меньше, чем конечных пользователей, и что такое же соотношение сохраняется и при переходе от проблемных программистов к системным и от последних - к конструкторам-разработчикам технологии.

5. Заключение. Опыт применения π-технологии пока невелик. С ее помощью созданы и внедрены в ряд организаций страны две π-системы: система логического анализа данных ЛАДА [5] и информационно-справочная служба РЕДАКЦИЯ [I6].

Обе системы являются типовыми и служат базой для новых разработок. На основе первой из них ведутся работы по созданию π-системы для принятия решений, а также по реализации в рамках системы ЛАДА идей Σ-программирования [I7].

На базе π-системы РЕДАКЦИЯ начато проектирование π-системы для автоматизации библиотечных работ.

Продолжаются также работы над созданием π-системы ПАРИС [I8] для обеспечения Полным Автоматизированным Редакционно-Издательским Сервисом редакторов и авторов научных публикаций. В ходе многолетней работы над этой сложной проблемой и сложились основные черты π-технологии.

Указанные первые применения π-технологии позволяют сделать некоторые предварительные выводы о сроках разработки и качестве создаваемых систем.

Разработка первой очереди π-системы ЛАДА заняла около двух человеко-месяцев. Это время потребовалось: на преобразование в S-модули имеющиеся программные модули, реализованные на языке ФОРТРАН-4 стандартными средствами; на разработку языка общения с пользователем и таблицы соответствий входных конструкций последовательностям S-модулей; привязку входного языка пользователя к имеющимся средствам организации диалога; разработку четырех косвенных командных файлов для связи с операционной системой ОС РВ СМ ЭВМ, а также на синтез управляющих программ.

Разработка π-системы РЕДАКЦИЯ, занявшая около шести человеко-месяцев, включала в себя разработку языка общения с пользователем и таблицы соответствий входных конструкций последовательностям S-модулей; привязку входного языка к имеющимся средствам организации диалога; разработку некоторых функциональных S-модулей. Основное время было затрачено на формирование базы данных. На этой π-системе впервые отлаживалась организация реляционной структуры базы данных, реализованная на основе вынесенных списков и инвертированных файлов. Именно этим можно объяснить значительно бо́льшие затраты труда на разработку π-системы РЕДАКЦИЯ по сравнению с π-системой ЛАДА, где пока применяется простая файловая организация, обеспечиваемая стандартными средствами ОС РВ.

Обе созданные π-системы обладают следующими качествами:

- доступностью конечному пользователю без обычного трудоемкого периода обучения. Общение с ЭВМ ведется на русском языке с помощью определенного набора фраз (своего у каждой из π-систем), достаточного для диалога о свойствах задач из данной проблемной области;

- эффективностью настройки на задачу. Так в системе ЛАДА синтез соответствующих программных средств занимает несколько минут по сравнению с несколькими неделями, которые требовались на составление и отладку головной программы при библиотечной организации программ анализа данных. π-системе РЕДАКЦИЯ требуются на настройку доли секунды;

- простотой реконструкции. В соответствии с пожеланиями пользователей, появившимися в ходе эксплуатации, у обеих π-систем был расширен круг решаемых ими задач. Каждая такая реконструкция (заключавшаяся в основном в расширении возможностей диалоговой и синтезирующей систем, а также в разработке и добавлении некоторых S-модулей) потребовала 1-3 человеко-дня.

Таким образом, уже по первым шагам к-технологии можно судить о ее жизнеспособности.

Литература

I. Персональные компьютеры. Материалы Всесоюз.конф. "Диалог-82-Микро", Пущино,1983. - 72 с.

2. ДЕНИНГ В., ЭССИГ Г., МААС С. Диалоговые системы Человек-ЭВМ: адаптация к требованиям пользователя. - М.:Мир,1984.-112 с,

3. ФОКС Дж. Программное обеспечение и его разработка.-М.:Мир, 1985.- 368 с.

4. КОСАРЕВ Ю.Г. О путях развития математического обеспечения средств вычислений.- В кн.: Проблемы обработки информации (Вычислительные системы,вып.100).Новосибирск,1983,с.3-10.

5. ВИТЯЕВ Е.Е., МОСКВИТИН А.А. ЛАДА - программная система логического анализа данных . - Настоящий сборник, с.38-58.

6. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные универсальные вычислительные системы высокой производительности.-Новосибирск:Наука,1966.- 308 с.

7. КОСАРЕВ Ю.Г. Распараллеливание по циклам. - В кн.: Вычислительные системы. Вып.24.Новосибирск,1967,с.3-20.

8. КОСАРЕВ Ю.Г. О математической модели гармоничных систем. Часть I. - В кн.: Математическое обеспечение ВС из микро-ЭВМ (Вычислительные системы,вып.96).Новосибирск,1983,с.51-74. Часть 2. - В кн.: Анализ разнотипных данных (Вычислительные системы,вып.99). Новосибирск,1983,с.15-38.

9. ВЕЛЬБИЦКИЙ И.В. Технология программирования. - Киев: Техника,1984. - 280 с.

10. ДРИБАС В.П. Реляционные модели баз данных.- Минск, 1982.- 192 с. (Белорусск.госуниверситет).

11. МОСКВИТИН А.А. Специализированный инструментальный транслятор. - В кн.: Вычислительная техника и дискретная математика, Новосибирск,1983,с.54.

12. КОСАРЕВ Ю.Г., ЧУХАНОВА Н.А. О построении обобщенной памяти машины для задач обработки текстовой информации.-В кн.: Автоматизация проектирования в микроэлектронике.Теория.Методы.Алгоритмы (Вычислительные системы,вып.77).Новосибирск,1978,с.63-71.

13. КОСАРЕВ Ю.Г. Примеры использования таблиц для сокращения времени счета. - В кн.: Вычислительные системы,вып.30.Новосибирск, 1968,с.46-54.

14. Ассоциативное кодирование: реализация и применение /Величко В.М., Гусев В.Д., Косарев Ю.Г. и др. - В кн.: Вычислительные системы,вып.62. Ассоциативное кодирование.Новосибирск,1975,с.3-37.

15. КОРМИЛИЦЫН Н.С., КОСАРЕВ Ю.Г. Программа внутренней сортировки для ЭВМ "Минск-32". - В кн.: Вычислительные системы,вып.59. Стандартные программы обработки информации. Новосибирск,1974,с.78-83.

16. МОСКВИТИН А.А. Автоматизированная информационно-справочная служба РЕДАКЦИЯ.- В кн.: Автоматизированные системы перера - ботки текстовой информации (АСПТИ): Тез.докл.Республиканской науч.-техн.конф. Львов,1985,с.27-28.
17. СВИРИДЕНКО Д.И. Проектирование Σ-программ. Постановка проблемы. - Настоящий сборник, с. 108-127.
18. КОСАРЕВ Ю.Г., МОСКВИТИН А.А. Система широкого применения для автоматизации редакционно-издательских работ. - В кн.: Методы обработки информации (Вычислительные системы,вып.74).Новосибирск, 1978,с.3-20.

Поступила в ред.-изд.отд.
3 сентября 1985 года