ТЕОРИЯ ВЫЧИСЛИМОСТИ И ЯЗЫКИ СПЕЦИФИКАЦИЙ
(Вычислительные системы)
1991 год Выпуск 139

УДК 519.682.5

ИНСТРУМЕНТ ОТБОРА СТРАТЕГИЙ ИСПОЛНЕНИЯ МОДУЛЕЙ (ПЕРВАЯ ВЕРСИЯ)

В.С.Костин, А.А.Москвитин

Одной из специфических черт этапа программирования является необходимость решения крупномасштабных задач. Характерной особенностью такого класса задач является их многовариантность, означающая то, что удовлетворительные решения (с точки зрения пользователя) могут быть получены разными способами. Естественно, что поиск таких способов и отбор наилучших вариантов требует соответствующих вычислительных экспериментов, что, в свою очередь, приводит к необходимости создания соответствующего инструментария. При разработке такого инструментария важно учесть, что выбор "наилучшего" решения предполагает учет нескольких факторов: адекватность решения задачи требованиям пользователя, с одной стороны, и ресурсные ограничения, с другой; знания пользователя о характере вычислений (априорные сведения о задаче, различные гипотезы, экспертные знания и т.п.); особенности обрабатываемых данных (различные способы представления информации: числовое, символьное, аналоговые сигналы; различные шкалы измерений и т.п.); способы обработки данных (последовательные, параллельные, потоковые, распределенные и др.); индивидуальные психологические особенности пользователя и многое другое. Наиболее показательными здесь могут быть информационные, экспертные системы и системы принятия решений.

Мы будем рассматривать задачу отбора стратегий исполнения как задачу выделения быстроисполнимых функций, осуществляющих направленный перебор элементов модели и наиболее адекватно отражающих преследуемую цель [1].

§1. Схема работы PI-конструктора

Рассмотрим один из инструментов отбора стратегий исполнения модулей, реализованный на персональном компьютере IBM PC/AT в операционной системе MS DOS 3.30.

Данный инструмент отбора стратегий исполнения модулей реализован методами PI-технологии [2] и является составной частью технологического комплекса решения задач СИГМА [3,4]. Его можно рассматривать как изолированно, так и при разработке программных систем и решении задач методами, отличными от семантического программирования.

Воспроизведем вкратце схему решения задач в PI-технологии. Пользователю предоставляется возможность подобрать структуру программной системы под его задачу и в соответствии с его пожеланиями. При этом он пользуется некоторой базовой системой программных средств, системой справочников; средствами конст - руирования и помощью со стороны PI-технологического комплекса.

Конструирование программной системы осуществляется в несколько этапов:

1 этап - постановка задачи и определение архитектуры будущей программной системы, а также доработка исполнительных модулей, если это необходимо по условию задачи;

2 этап - ввод априорных сведений о задаче, выдвижение и фиксация гипотез пользователя о возможных результатах процесса решения задачи;

3 этап - анализ входной информации и настройка программной системы на потоки данных;

4 этап - анализ результатов вычислений и переопределение архитектуры системы, либо ее базификация в виде программы-стратегии исполнения модулей в базе данных и программ.

Реализация перечисленных выше этапов осуществляется в PI-технологии выделением,переопределением и фиксацией управляющих и настроечных параметров в специальном управляющем массиве, а не написанием и отладкой каждый раз новых управляющих программ, как это делается в традиционных схемах построения программных систем. Заметим сразу же, что при такой организации первоначально зафиксированная архитектура конструируемой программной системы может видоизмениться в процессе решения задачи как под влиянием потока входных данных,так и под влиянием полученных результатов исполнения модулей.

В данной схеме вычислений (в отличие от традиционной) всю тяжесть управления программной системой берет на себя управляющий массив. Это же обстоятельство позволяет упростить процесс конструирования, настройки и исполнения системы, сведя всю работу по конструированию программной системы к простой смене некоторого фиксированного числа параметров в управляющем массиве. Именно за счет простоты и оперативности конструирования программных систем удается быстро и качественно достичь требуемого результата.

Данная схема конструирования программных систем рассчитана на манипулирование объектами различной природы, но при этом существенно зависит от развитости технологической среды окружения.

Согласно идеологии PI-технологии среда окружения технологического комплекса должна состоять из пяти подсистем: диалоговой, синтеза, операционной, анализа и обработки данных,а также подсистемы управления базой данных и программ. Естественно, что круг решаемых задач будет определяться не только насыщен-

ностью базы данных и программ, содержащей стратегии исполне - ния, но и качественностью реализации всех пяти подсистем.

Для наполнения базы данных и программ стратегиями испол - нения программных модулей и создан описываемый ниже программ - ный комплекс, который по своим функциональным возможностям позволяет решать и другие задачи, отличные от задач отбора стратегий исполнения модулей. Такими задачами могут быть: задачи анализа данных и отбора закономерностей; задачи обработки информации; вычислительные задачи с многопараметрическим выбором; задачи конструирования различных отладочных комплексов и т.п.

Для обработки схемы конструирования PI-систем и отбора стратегий исполнения модулей с целью создания базы данных и программ базового уровня технологического комплекса СИГМА реализована первая версия PI-конструктора. Конструирование программных систем производится из элементарных задач, оформлен - ных в виде загрузочных модулей.

База данных и программных средств конструктора PI-систем первой версии содержит: некоторое количество загрузочных модулей, справочников решаемых задач, файлов данных и связей и нацелена на пополнение новыми программами пользователя, написанными на языках СИ и ассемблере.

Схема работы в PI-конструкторе следующая. Пользователю предъявляется справочник микрозадач, реализованных в виде мо - дулей, выполняющих логически-завершенные функции,и содержащихся в базе данных и программ. Из этого набора необходимо выбрать те микрозадачи, набор которых позволяет решить основную задачу. Далее, из выделенного набора микрозадач надо выбрать исходный модуль и запустить его на выполнение. Кроме этого,для работы конструктора необходимо указать файлы входных данных и место размещения результатов вычислений.

В случае, когда для решения задачи требуется подключение модулей,не указанных пользователем при конструировании,PI-конструктор указывает на их отсутствие и предлагает включить их в состав конструируемой системы. При полностью сформированной схеме вычислений конструктор свою работу заканчивает и выдает созданной системе указание на выполнение вычислений.

Выбирая различные схемы решения задачи, указывая разные исходные модули и подключая различные входные данные,пользователь может подобрать для своей задачи наиболее оптимальный вариант стратегии исполнения.

Ниже приводится пример сценария конструирования фрагмента редактора текстов (рис.1,2).

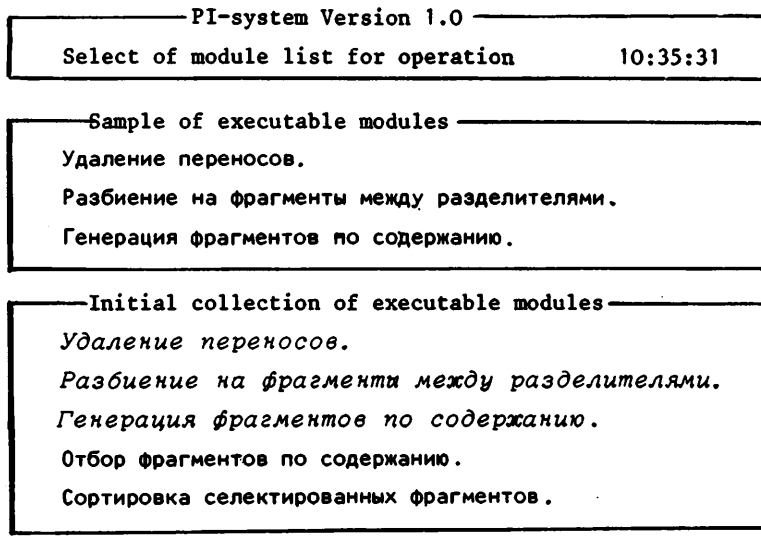

Рис.1. Отбор микрозадач из базы данных и программ

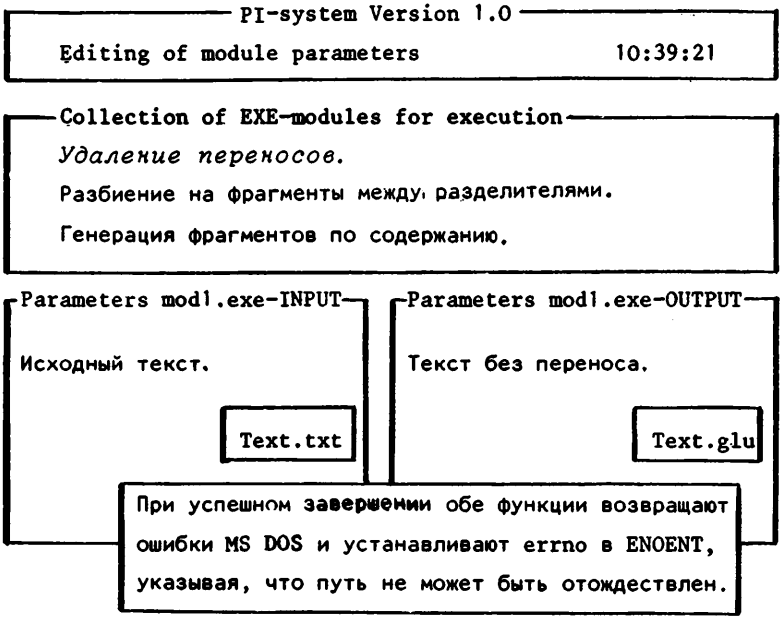

Рис.2. Фрагмент запуска PI-конструктора на выполнение

§2. Структура и состав PI-конструктора

База данных и программ (БДП) PI-конструктора состоит из: Файла-справочника задач (файл с именем PI.rpm); файла-справоч - ника данных (файл с именем PI.fpd); набора модулей исполнения (фэйлы с расширением exe); самого конструктора (файл с именем PRL.exe) и файлов данных (с именами из PI.fpd). Структура базы данных и программ приведена на рис.3.

Структура справочников PI.fpd и PI.rpm следующая:

1 строка - имя данного или модуля исполнения;

2 строка - значение данного или имя файла модуля.

Для включения исполнительных модулей в базу данных и программ PI-конструктора требуется их небольшая доработка, состоящая из следующих этапов.

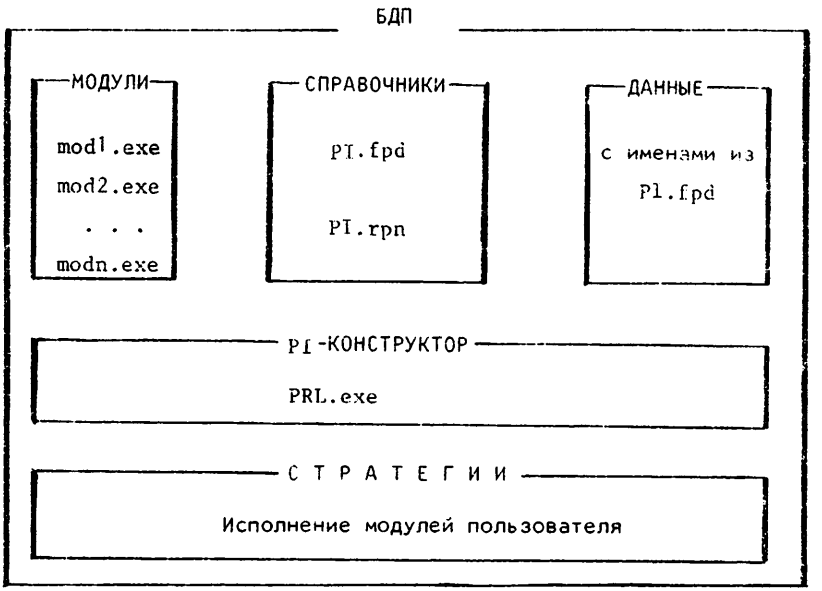

Рис.3. Структура PI-конструктора программных систем

Предварительный этап. Краткое описание задачи (одна строка) и согласование ее имени с уже существующими в базе данных и программ. Описание задачи и ее имя должны быть уни - кальными. Это необходимо для устранения возможных неоднознач - ностей, могущих возникнуть в процессе конструирования PI-сис- тем.

Этап 1. Дополнение файла PI.rpm парой (идентификатор задачи, имя exe-файла).

Этап 2. Дополнение файла PI.fpd парой (идентификатор переменной, значение по умолчанию для всех новых параметров).

Этап 3. Модификация текста программы с целью согласования интерфейса ввода/вывода:

3.1. # include "pi.h" - подключение файла с описанием структур;

3.2. включение операторов extopen(...) и extclose(...);

3.3 #define _N_ 5 - задание размера структуры описания параметров (5 - общее количество параметров);

3.4 char *mfunc[2]={"описание задачи", "размер окна экрана для вывода текста"};

3.5. описание параметров модуля:

struct io extpar [_N_]=

={описатель 1, описатель 2, описатель 3, "имя зада - чи",...};

где описатель 1 ={INPUT_, OUTPUT_};

описатель 2={TEXT_,INT_,DOUBLE_};

описатель 2={FILE_,PARAMETER_};

3.6 main(narg,arg)

char *arg[ ];int narg; - включение аргументов,передаваемых командной строкой MS DOS из PI-системы;

3.7 FILE *f[_N_]]- описание массивов для передачи параметров через int ipar[_N_], double dpar[_N_], char *tpar[_N_];

3.8 extopen(narg,arg,_N_,extpar,f,ipar,dpar,tpar,mfunc);- прием параметров из системы через командную строку и extclose (_N_,extpar,f,ipar,dpar,tpar); - передача выходных параметров обратно в систему и закрытие файлов.

Ниже приведен пример адаптированного модуля фрагмента создания редактора текстов:

```
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <process.h>
#include "pi.h"
#define _N_ 3
  /* mod2.exe */
  char *mfunc[2]={"Разбиение на фрагменты между разделителями", "23,20"};
  struct io extpar[_N_]=
            {
            INPUT_,TEXT_,FILE_,"Список разделителей",
            INPUT_,TEXT_,FILE_,"Текст без переносов",
```

```
            OUTPUT_,TEXT_,FILE_,"Фрагменты - текст без разде-
                                 лителей",
            };
int rd[256];
main(narg,arg)
int narg;
char *arg[ ];
{
FILE *f[_N_];
int ipar[_N_];
doumle dpar[_N_];
char *tpar[_N_]}
    int c;
    int i;
extopen(narg,arg,_N_,extpar,f,ipar,dpar,tpar,mfunc);
    for(i=0;(c=getc(f[0]))!=EOF;rd[c]=++i);
    for(i=0;(c=getc(f[1]))!=EOF;)
       {
           if(rd[c]==0) i++;fputc(c,f[2]);putchar(c);}
           else if(i!=0){fputc((unsigned int)'\n',f[2]);
           putchar((unsigned int)'\n');i=0;}
        }
extclose(_N_,extpar,f,ipar,dpar,tpar);
}
```

В заключение отметим, что описанный выше конструктор PI-систем (первой версии) успешно применяется при разработке различных задач анализа данных, при создании систем обработки изображений, а также при отладке информационно-справочных систем.

Дальнейшее развитие данного инструмента направлено на разработку схемы взаимодействия модулей (в исходных или объектных кодах),согласно методам и средствам PI-технологического комплекса построения программных систем и направлено на реализацию базисного уровня технологического комплекса решения задач СИГМА.

Литература

1. ГОНЧАРОВ С.С., СВИРИДЕНКО Д.И. Σ^{+}-программы и их семантики //Логические методы в программировании. - Новосибирск, 1987. - Вып. 120: Вычислительные системы. -С. 24-51.

2. МОСКВИТИН А.А. Проблемно-инструментальная технология построения информационно-логических систем: Автореф.дис...канд. техн.наук: 05.13.11. - Новосибирск, 1986. - 17 с.

3. ГОНЧАРОВ С.С., СВИРИДЕНКО Д.И. Σ-программирование //Логико-математические проблемы МОЗ. - Новосибирск,1985.-Вып.107: Вычислительные системы. -С. 3-29.

4. СВИРИДЕНКО Д.И. Проектирование Σ-программ. Σ-оцениваемость //Логические вопросы теории типов данных. - Новосибирск, 1986. -Вып. 114: Вычислительные системы. -С. 59-83.

Поступила в ред.-изд.отд.

12 июля 1991 года