Конструирование управления геоинформационной системой на основании формальных моделей вычислительного процесса.

Ю.Г. Васин, С.Г. Кузин, А.С. Малыгин

Аннотация

Предлагается формализация понятия вычислительного процесса и модели вычислительного процесса, а также методология разработки управляющей компоненты сложного программного комплекса на основании формальных моделей вычислительного процесса. Приводятся примеры моделей вычислительного процесса для конкретной геоинформационной системы.

1. ВВЕДЕНИЕ

В настоящее время широко распространены методы автоматизированного конструирования информационных систем (CASE – технологии), основанные на концептуальных моделях потоков данных в предметной области (ER-диаграммы и т.п.). В этих методах основной упор делается на формальное вычисление схемы базы данных и запросов к базе данных (на основании концептуальной модели потоков данных в предметной области). Используются также функциональные модели, которые представляют схемы обработки данных и являются скорее моделями для управления разработками [1,2].

Конечным итогом подобной технологии является реляционная база данных, способная выполнять незапланированные запросы. Какая-либо навигация в реляционной базе данных отсутствует. Широкое распространение реляционных баз данных и различных технологий их конструирования обусловлено адекватностью подобного формализма реальным задачам компьютеризации человеческих технологий в сфере бизнеса, экономики, управления и т.д.

Однако далеко не все задачи информатизации человеческой деятельности вписываются в эту весьма эффективную, но далеко не универсальную парадигму:

- многие приложения состоят из большого числа взаимодействующих частей, каждая из которых обладает своим поведением и зависит от поведения других;
- многие системы должны обрабатывать большие объемы сложно структурированных данных;
- многие приложения осуществляют предсказуемый доступ к данным, поэтому навигационный доступ к данным в них является желательным;
- в большинстве приложений потребность в незапланированных запросах ограничена.

Вышеназванные соображения привели к созданию парадигмы объектно-ориентированных баз данных, которая является симбиозом реляционного подхода к моделированию данных и объектно-ориентированной технологии разработки сложных программных систем [3,4]. Можно говорить, что объектно-ориентированные базы данных решают проблему системной увязки структуры хранения данных и обрабатывающих функций. Но при этом за кадром остаются проблемы организации последовательности вызова обрабатывающих функций, как правило, в диалоговом режиме. Для этой цели не предлагается адекватной модели вычислительного процесса (схемы диалога) и реализация «прокрутки» функций отдается на волю создателя головной программы.

Многие программные комплексы (САПР, геоинформационные системы) требуют не только реализации сложно структурированных хранилищ данных, но также взаимодействия множества уникальных обрабатывающих программных модулей, порядок исполнения которых не может быть заранее запланирован. Более того, правомерна постановка проблемы конструирования на базе множества программных обрабатывающих модулей конкретных программных систем по заказу пользователя. В этом случае формализм модели вычислительного процесса (схемы диалога) посредством которого заказчик и архитектор программной системы находят консенсус, является не только желательным, но и обязательным.

С этой точки зрения, заслуживает внимания технология создания программного обеспечения «реактивных» систем, основанная на автоматном походе [5]. В этом случае в качестве модели вычислительного процесса используется система диаграмм переходов коллектива автоматов, состояниями которого являются состояния технического устройства. Это позволяет проектировать управление программной системой на содержательном уровне, используя для реализации управления универсальный алгоритм функционирования коллектива автоматов.

Проблеме управления в сложных программных системах посвящена работа [6], в которой также используется графовая модель информационной увязки программных модулей, на которую проектируется граф сценария диалога.

В Научно Исследовательском Институте Прикладной Математики и Кибернетики в течение ряда лет ведется работа по созданию методологии автоматизированного конструирования геоинформационных систем на основании специфических графовых моделей [7,8]. К числу таких моделей относятся: информационно-алгоритмическая модель функционального процессора, а также модель вычислительного процесса. Настоящая статья посвящена дальнейшему развитию этой методологии и ее иллюстрации на примере проектирования конкретной геоинформационной системы ГрафКарт.[1]

Предлагаемая нами технология конструирования больших программных систем ориентирована на **вычислительные системы**, спецификой которых является:

- большое разнообразие программных модулей обработки данных;
- необходимость конструирования разнообразных технологических цепочек, реализующих сложные вычислительные процессы на базе этих модулей;

[1] Работа выполнена при финансовой поддержке РФФИ, грант поддержки ведущих научных школ № 00-15-96108, и ФЦП "Интеграция", проект К 03392.

• необходимость управлять в режиме диалога "прокруткой" этих модулей, причем, порядок "прокрутки" модулей существенно зависит от конкретной решаемой задачи.

При этом необходимо решать следующие проблемы:

• проектирование модулей обработки данных;

• системную увязку этих модулей по данным;

• формирование сценария диалога;

• реализацию сценария в виде управляющего процессора.

В отличие от [5], предлагаемые модели управляют не устройствами, а действиями по преобразованию данных, что обеспечивает большую их универсальность. В отличие от [6], предлагается использовать две модели вычислительного процесса - модель информационно-алгоритмической увязки и модель управления вычислительным процессом, что позволяет более четко проектировать как взаимодействие обрабатывающих модулей, так и схему диалогового управления [7,8].

2. ФУНКЦИОНАЛЬНЫЙ ПРОЦЕССОР ПРОГРАММНОГО КОМПЛЕКСА

Допустим, что для решения класса однородных, с некоторой точки зрения, прикладных задач **T** создается множество $\mathbf{M^T}$ программных модулей обработки данных, которое определяется проблемной ориентацией этого класса прикладных задач. Под программным модулем $M^T_i \in M^T$ будем понимать программу/подпрограмму обработки данных:

$$M^T_i(In_0,In_1,\ldots,In_p;Out_1,\ldots,Out_q); \qquad (1)$$

которая может запускаться функциональным процессором при поступлении на его управляющий вход соответствующей команды. Здесь, $In_1,\ldots,In_p$ –входные операнды; $Out_1,\ldots,Out_q$ –выходные операнды; In_0 – параметрический операнд, определяющий режим функционирования модуля. В частном случае, программный модуль оформляется в виде exe – файла, т.е. может запускаться непосредственно операционной системой. Операндами такого программного модуля служат файлы и, таким образом, с другими модулями множества M^T модуль M^T_i имеет связь по данным через файловую подсистему операционной системы.

Такой программный модуль идентифицируется полным именем файла в файловой подсистеме операционной системы. Каждому модулю M^T_i ставится в соответствие команда σ^T_i - текстовая строка, которая обозначает действие модуля на "человеческом" языке. Отношение команда – модуль фиксируется в таблице команд **ТАВСОМ^T**, каждая строка которой имеет вид:

$$<\sigma_i, \text{ПОЛНОЕ\_ИМЯ}(M^T_i)>. \quad (2)$$

Множество команд, идентифицирующих программные модули, образует систему команд Σ^T функционального процессора FP^T (Рисунок 1).

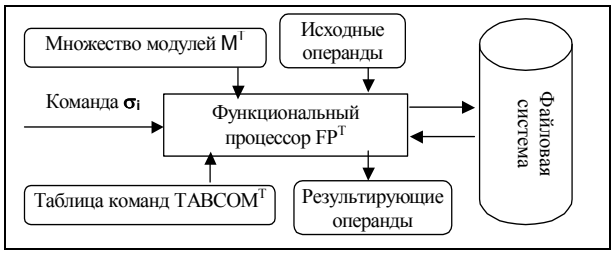

Рисунок 1. Функциональный процессор программного комплекса

В качестве примера рассмотрим функциональный процессор управления прикладной геоинформационной программой ГрафКарт: решение задач на графах, погруженных в электронную карту [9-15]. Здесь в качестве программных модулей используются подпрограммы-функции, связь по данным реализуется стандартным способом через механизм формальных/фактических параметров.

Исходной информацией для ГрафКарт служат картографическая база данных в формате интегрального файла и файл классификатора [9-12]. ГрафКарт позволяет: отобрать необходимые графы, погруженные в электронную карту; экспортировать их в архив графов; выбирать графы из архива с целью решения прикладных геоинформационных задач.

Таблица команд функционального процессора ГрафКарт приведена ниже (Таблица 1). Название столбцов:1- команда, 2- полное имя модуля; 3- семантика модуля, 4- входные операнды, 5- выходные операнды.

Таблица 1

1	2	3	4	5
σ_1	A1	Выбор карто-графич. базы данных	BDK, BKL	IF, KL
σ_2	A2	Отбор по классификатору объектов-вершин	KL	KKN
σ_3	A3	Отбор по классификатору объектов-дуг	KL	KKL
σ_4	A4	Выделение и экспорт графа в архив графов	IF, KKN, KKL	AG
σ_5	A5	Выбор графа из архива	AG	G
σ_{61}… σ_{6n}	A61 - A6n	Решение прикладных задач на графе	G	R
σ_7	A7	Визуализация	R	DSP

3. ИНФОРМАЦИОННО-АЛГОРИТМИЧЕСКАЯ МОДЕЛЬ ФУНКЦИОНАЛЬНОГО ПРОЦЕССОРА

Системная увязка алгоритмов (модулей) обработки и структур данных в пределах функционального процессора отображается иерархической информационно-алгоритмической моделью функционального процессора. Для случая ГрафКарт такая модель изображена на Рисунок 2.

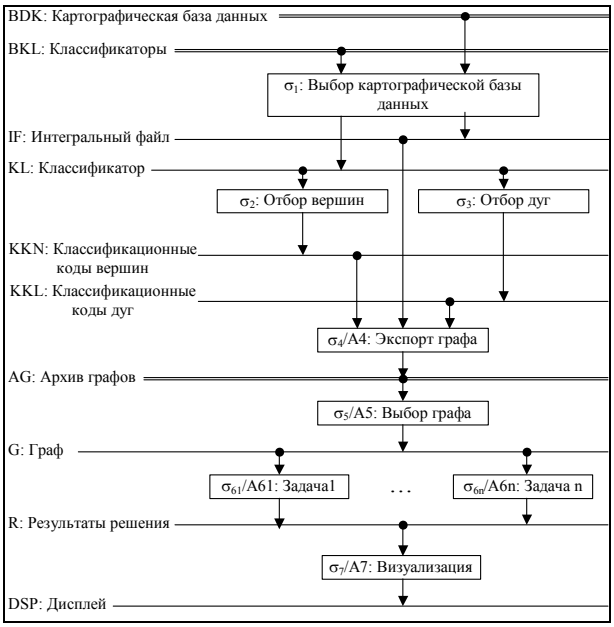

Рисунок 2. Информационно-алгоритмическая модель функционального процессора ГрафКарт

Операнды программных модулей на информационно-алгоритмической модели изображаются горизонтальными уровнями. Для каждого алгоритма (программного модуля) входные и выходные операнды определяют подключение этого алгоритма к соответствующим уровням.

4. СЕТЕВАЯ МОДЕЛЬ УПРАВЛЕНИЯ ВЫЧИСЛИТЕЛЬНЫМ ПРОЦЕССОМ.

Каждая прикладная задача $t \in T$, явно или неявно, предполагает существование некоторого семейства допустимых наборов исходных данных D^t и некоторого семейства результирующих наборов данных R^t. Таким образом, можно говорить о функциональном преобразовании прикладной задачи:

$$t: D^t \rightarrow R^t. \qquad (3)$$

Решение прикладной задачи заключается в вычислении функционального преобразования (3) в конкретной точке, т.е. в определении:

$$t(d) = r,\ d \in D^t \text{ и } r \in R^t. \qquad (4)$$

Алгоритм решения прикладной задачи – вполне определенный способ вычисления функционального преобразования прикладной задачи (3) в заданной точке. Понятие алгоритма является абстрактным понятием, которое конструктивно реализуется в виде вычислительного процесса P^t.

Траектория вычислительного процесса P^t - последовательность команд функционального процессора $\{\sigma\}$, вычисляющая функцию обработки данных в точке d. Заметим, что, за исключением тривиальных случаев, способ вычисления функции обработки данных зависит от конкретного $d \in D^t$. Таким образом, область допустимых исходных данных D^t разбивается на классы эквивалентности $D^t = \{D^t_\alpha\}$ таким образом, что все $d \in D^t_\alpha$ обрабатываются одной и той же траекторией:

$$\sigma^t{}_\alpha = [\sigma^\alpha{}_1, \sigma^\alpha{}_2, \ldots \sigma^\alpha{}_k]. \qquad (5)$$

Множество всех траекторий образует **спектр траекторий $S^t = \{\sigma^t_\alpha\}$ вычислительного процесса P^t**.

Очевидно, что должна обеспечиваться информационная совместимость образующих траекторию команд. Говоря другими словами, команды, предшествующие σ^i_α, должны вырабатывать значения всех входных операндов для σ^i_α. Также очевидно, что информационная совместимость команд траектории задается (контролируется) информационно – алгоритмической моделью функционального процессора.

Используя введенные выше определения, проблему управления вычислительным процессом можно сформулировать следующим образом: ***для заданного*** $\mathbf{d \in D^t}$ ***и заданного функционального процессора*** $\mathbf{FP_\Sigma}$ ***определить траекторию вычислительного процесса*** $\boldsymbol{\sigma^t_\alpha}$, ***исполнение которой функциональным процессором обеспечит вычисление функции обработки данных в заданной точке, т.е. получение результата*** $t(d) = r$.

Будем использовать также понятие **языка управления вычислительным процессом**, которое введем следующим образом;

- предложением этого языка будем называть последовательность символов, однозначно обозначающих какую либо траекторию вычислительного процесса;
- множество L^t обозначений всех траекторий S^t назовем формальным языком управления вычислительным процессом.

Примечание. Смысл использования, наряду со спектром траекторий, формального языка управления вычислительным процессом заключается в следующем. Траектория определяется как предложение машинно-ориентированного языка, "понятного" функциональному процессору, тогда как предложение языка управления должно быть ориентировано на "легкое" понимание его человеком.

Учитывая тот факт, что мощность спектра траекторий вычислительного процесса, в общем случае, является большой или даже бесконечной, решать проблему управления вычислительным процессом непосредственно на спектре траекторий не представляется возможным. Для этого привлекается понятие модели вычислительного процесса, а также модели языка управления вычислительным процессом.

Модель спектра траекторий вычислительного процесса P^t мы определяем как конечное компактное описание спектра S^t траекторий прикладной задачи $t \in T$. Из определения следует, что модель спектра траекторий вычислительного процесса конструируется для конкретной прикладной задачи t из класса T.

Примечание. Очевидно, что для прикладной задачи t может существовать множество моделей вычислительного процесса, отражающие различные способы ее решения. Это тем более справедливо для всего класса прикладных задач T.

Модель языка управления вычислительным процессом P^t мы определяем как конечное компактное описание формального языка L^t, каждое предложение которого однозначно обозначает некоторую траекторию из S^t.

Модель управления вычислительным процессом (MCTRL^t) включает в себя следующие компоненты:

- модель спектра траекторий S^t вычислительного процесса;
- модель языка управления L^t вычислительным процессом;

- взаимно однозначное отображение $S^t \leftrightarrow L^t$.

В работе [7] предложено использовать в качестве модели управления вычислительным процессом автоматную сетевую модель. Она представляет собой специфический направленный граф, имеющий одну начальную и одну конечную вершины (Рисунок 3).

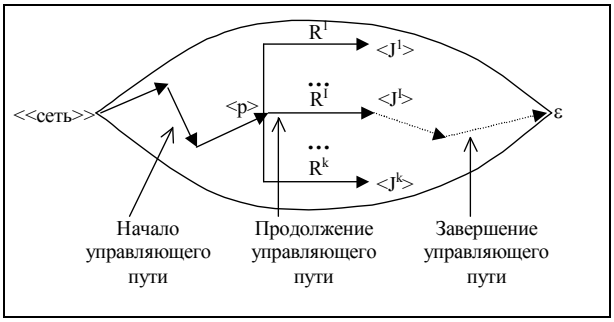

Рисунок 3. Автоматная сетевая модель вычислительного процесса

Через каждую вершину сети проходит хотя бы один путь, ведущий из начальной вершины в конечную вершину. Дуга сети имеет "верхнюю" метку R^I – лексему языка управления вычислительным процессом. Также дуга сети может иметь "нижнюю" метку $/\sigma^I/$ – команду функционального процессора.

Любой путь в сети порождает некоторую траекторию вычислительного процесса - последовательность "нижних" меток образующих путь дуг. Так же этот путь порождает обозначающее эту траекторию предложение языка управления вычислительным процессом - конкатенация "верхних" меток образующих путь дуг.

Очевидно, что множество всех путей в сети порождает как спектр траекторий S^t, так и множество предложений языка управления L^t. Также очевидно, что имеется взаимно однозначное отображение $S^t \leftrightarrow L^t$. Все сказанное выше позволяет рассматривать автоматную сеть как модель управления вычислительным процессом.

Во многих случаях целесообразно использовать обобщение автоматной сетевой модели – рекурсивную сетевую модель вычислительного процесса. Такая модель представляет собой систему взаимосвязанных автоматных сетей, в которой выделена одна главная сеть и каждая сеть, кроме терминальной, ссылается на другую сеть этой системы.

Приведем пример рекурсивной сетевой модели управления вычислительным процессом решения геоинформационных задач, который реализуется в программной системе Граф-Карт (Рисунок 4).

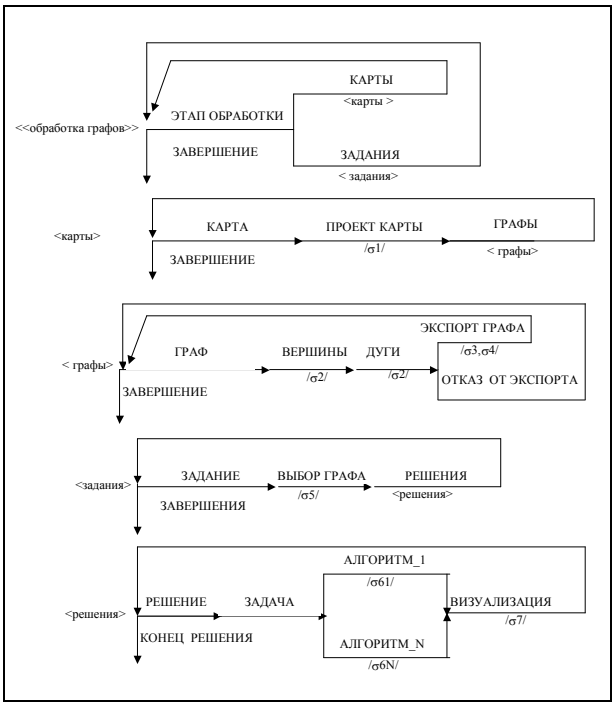

Рисунок 4. Рекурсивная сетевая модель управления вычислительным процессом

5. УПРАВЛЕНИЕ ВЫЧИСЛИТЕЛЬНЫМ ПРОЦЕССОМ

Управление вычислительным процессом на основании сетевой модели осуществляется **управляющим процессором**, который для заданного набора исходных данных $d \in D^t_\alpha$ и заданной модели управления вычислительным процессом $MCTRL^t$ выделяет в сети **управляющий путь**, порождающий траекторию вычислительного процесса σ^t_α, исполняя которую функциональный процессор FP_T преобразует d в t(d).

Пусть управляющий путь в сетевой модели (Рисунок 3) определен до вершины <p>. Если из этой вершины выходит одна дуга (безальтернативная вершина), проблемы с продолжением пути не возникает. Решение о продолжении пути должно приниматься в вершине <p>, из которой выходит несколько дуг (альтернативная вершина). Предполагается, что дуги, выходящие из альтернативной вершины, помечены различными "верхними" метками – лексемами языка управления (Рисунок 3).

В зависимости от способа принятия решения о продолжения пути в альтернативной вершине, существуют три основные стратегии управления вычислительным процессом.

1. Автоматическое управление. Управляющий процессор на основании анализа конкретных исходных данных и получающихся промежуточных результатов без вмешательства человека определяет продолжение управляющего пути в каждой альтернативной вершине сетевой модели вычислительного процесса (Рисунок 3).

Траектория вычислительного процесса, порождаемая управляющим путем, исполняется функциональным процессором. (Например, таким управляющим процессором является устройство центрального управления компьютера. В этом случае, в качестве функционального процессора выступает арифметико-логическое устройство).

Предложение языка управления, порождаемое управляющим путем, целесообразно выдавать пользователю как толкование вычислительного процесса.

2. Интерактивное управление подразумевает диалог человека с управляющим процессором, в результате которого принимается решение о продолжении управляющего пути в альтернативной вершине сетевой модели вычислительного процесса (Рисунок 3). Выделяется два способа интерактивного управления.

- Командное управление. Решение о продолжение управляющего пути реализуется в виде команды R^I, в качестве которой выступает "верхняя" метка дуги, выходящей из вершины <p>. (Пример – командный режим работы Norton Commander.)

- Менюшное управление. В каждой альтернативной вершине для пользователя выдается меню, строками которого являются "верхние" метки дуг этой вершины. Выбирая строку, пользователь определяет продолжение управляющего пути в этой вершине. (Пример – менюшный режим работы Norton Commander.)

3. Директивное управление. Пользователь априори определяет траекторию вычислительного процесса, необходимую для обработки конкретного набора исходных данных, посредством формирования соответствующего предложения языка управления. Управляющий процессор контролирует допустимость предложения, отображает его в соответствующую траекторию и передает последнюю для исполнения функциональному процессору. (Примером предложения языка управления вычислительным процессом служит bat-файл операционной системы DOS, который определяет необходимую траекторию вычислительного процесса и обрабатывается соответствующим процессором.)

При любой стратегии управления, проблему формирования управляющего пути, соответствующего конкретному набору исходных данных, решает управляющий процессор.

Управляющий процессор, в зависимости от стратегии управления, реализует ту или иную модификацию базисного алгоритма функционирования детерминированного конечного автомата [7].

Управляющий процессор может быть реализован в виде универсальной программы (Рисунок 5). В этом случае, модель управления играет роль входных метаданных, настраивающих универсальный управляющий процессор CP на управление конкретным вычислительным процессом t.

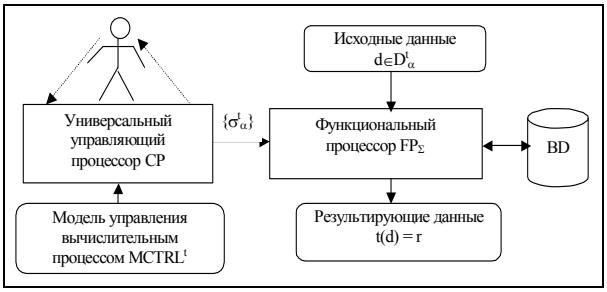

Рисунок 5. Универсальный управляющий процессор

Управляющий процессор может быть реализован в виде специальной программы, управляющей одним и только одним конкретным вычислительным процессом (Рисунок 6).

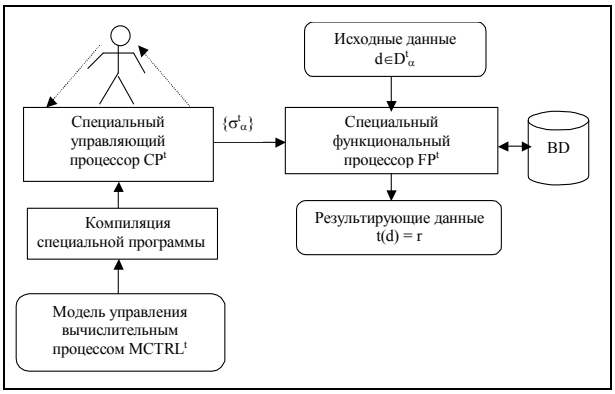

Рисунок 6. Специальный управляющий процессор

В последнем случае, модель управления вычислительным процессом $MCTRL^t$ выступает в качестве исходных данных для компиляции специальной управляющей программы. Процесс компиляции может быть как "ручным" (программирование) так и "автоматическим" (автоматическое программирование).

В представленной геоинформационной системе ГрафКарт управляющий процессор реализован в виде специальной программы.

Начало диалога ГрафКарт приведено на Рисунок 7.

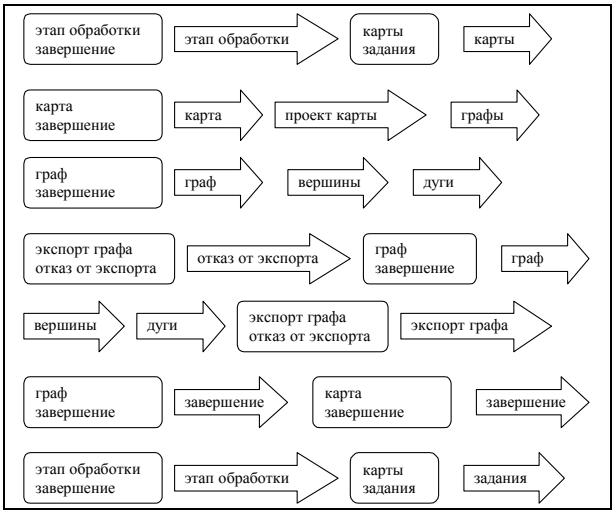

Рисунок 7. Пример диалога ГрафКарт. Прямоугольники – меню, стрелки – действия.

6. ЗАКЛЮЧЕНИЕ

В работе, на примере создания геоинформационной системы ГрафКарт, продемонстрирована технология конструирования сложных диалоговых программных систем с использованием графовых моделей. Для этого:

- построена модель функционального процессора, представляющая множество алгоритмов (программных модулей) и их системную увязку по данным;

- построена сетевая модель диалогового управления геоинформационной системой;

- управление геоинформационной системой ГрафКарт реализовано в виде программы специального управляющего процессора.

- реализация и опытная эксплуатация программной системы ГрафКарт показала адекватность используемых моделей проблеме управления, а также подтвердила эффек-

эффективность предлагаемой технологии конструирования управляющих компонент сложных программных систем.

7. ЛИТЕРАТУРА

[1] Вендеров А.М. Проектирование программного обеспечения экономических информационных систем. М. Финансы и статистик, 2000.

[2] Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя. М.: ДМК, 2000.

[3] С. Кузнецов. Объектно-ориентированные базы данных – основные концепции, организация и управление: краткий обзор. www.citrum.ru/database/articles/art_24.shtml

[4] Д. Харрингтон. Проектирование объектно-ориентированных баз данных. М., ДМК, 2001.

[5] А.А. Шалыто, Н.И. Туккель. SWITCH-технология: автоматный подход к созданию программного обеспечения "реактивных" систем. Программирование, 2001, №5, с. 45-62..

[6] Программные системы. Применение. Разработка. Обоснование. Под редакцией Бахмана. М.. Мир, 1998.

[7] Кузин С.Г., Кошелев М.В. Модели и способы управления вычислительным процессом// Вестник Нижегородского университета: Математическое моделирование и оптимальное управление. Н. Новгород: Изд-во Нижегородского университета, 1997. с.184-195.

[8] Васин Ю.Г., Кошелев М.В., Кузин С.Г., Смирнов А.Ф. Об одной технологии конструирования сложных программных комплексов// Вестник Нижегородского университета: Математическое моделирование и оптимальное управление. Н. Новгород: Изд-во Нижегородского университета, 1998, 1(18). с.213-226.

[9] Васин Ю.Г., Кобрин Р.Ю., Коротков В.М. База знаний по лесоустройству и ее реализация в автоматизированной картографической системе. Информационные процессы и системы, 1990, №9, с. 6-14.

[10] Васин Ю.Г., Ясаков Ю.В. Система управления базами видеоданных. Методы и средства обработки графической информации. Межвуз. сб. ГГУ, Горький, 1987, с. 153-162.

[11] Васин Ю.Г. Объектно-ориентированный топологический обменный формат интегрального файла. В кн. Тезисы докладов 3 конференции "Распознавание образов и анализ изображений: новые информационные технологии". РОАИ, Н. Новгород, 1997. ч. 2, с. 19-22.

[12] Васин Ю.Г., Ясаков Ю.В. Организация хранения и обработки информации в формате интегрального файла. В кн. Тезисы докладов 3 конференции "Распознавание образов и анализ изображений: новые информационные технологии". РОАИ, Н. Новгород, 1997. ч. 2, с. 43-45.

[13] Городецкая Н.И., Васин Ю.Г. Автоматическое планирование оптимального маршрута по дорожной сети. Автоматизация обработки сложной графической информации. Межвуз. сб. ГГУ, Горький, 1987, с. 153-162.

[14] Васин Ю.Г., Кузин С.Г., Малыгин А.С. Тематическая электронная карта ГРАФКАРТ для решения геоинформационных графовых задач. //В кн. VI Всероссийская конференция с участием стран СНГ "Методы и средства обработки сложной графической информации". Тезисы докладов. Н. Новгород, 2001.

[15] Кузин С.Г., Малыгин А.С. Формальные модели и алгоритмы экспорта графовой информации из интегрального файла электронной карты в базу данных табличного типа. // Вестник Нижегородского университета: Математическое моделирование и оптимальное управление. Н. Новгород: Изд-во Нижегородского университета, 2001, 1(23). с. 289-297.

Об авторах

Васин Юрий Григорьевич, директор НИИ Прикладной Математики и Кибернетики ННГУ, д.т.н., профессор, член-корр. АТН РФ.

Кузин Станислав Григорьевич, Нижегородский государственный университет, НИИ ПМК ННГУ, к.т.н., доцент.

Малыгин Алексей Сергеевич, аспирант ННГУ, НИИ ПМК ННГУ.