Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Л.В. БОЧКАРЁВА

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

по курсу

«Технология разработки и системы автоматизированного проектирования программного обеспечения»

для студентов специальности
«Программное обеспечение информационных технологий»

Минск 2002

УДК 681.3.061(075.8)
ББК 32.973я73
Б 86

Бочкарёва Л.В.

Учебно-методическое пособие по курсу «Технология разработки и сис-
Б 86 темы автоматизированного проектирования программного обеспечения» для студентов специальности «Программное обеспечение информационных технологий» / Л.В. Бочкарёва. – Мн.: БГУИР, 2002. – 52 с.: ил. 19

ISBN 985-444-387-6

В пособии рассмотрены технология разработки и автоматизированный синтез ПО, основные этапы и методы создания программных средств, способ автоматизированного синтеза и ориентации в программных средах САПР на основе модели представления данных и знаний - семантического полигона. Предложен лабораторный практикум по курсу “ТР и САПР ПО”.

УДК 681.3.061(075.8)
ББК 32.973я73

BN 985-444-387-6

СОДЕРЖАНИЕ

1. АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ ПРОГРАММНЫХ СРЕДСТВ

1.1. Цели и задачи систем автоматизированного проектирования программного обеспечения

Создание систем автоматизированного проектирования программного обеспечения можно представить как сочетание двух развивающихся перспективных направлений, а именно, технологии конструирования программных средств и автоматизации проектирования.

Программные средства (ПС) являются исполнительными элементами многих компьютерных систем различного назначения. В отличие от программ, создаваемых в единственном экземпляре для решения частных исследовательских задач, моделирования, обработки результатов экспериментов и не имеющих массового применения, ПС должны использоваться многократно. Для обеспечения повторного использования ПС или программных модулей (ПМ) необходимы стандартизация их оформления, написания, испытания, а также гарантии качества. Стандарты разных уровней обычно регламентируют технологию разработки, постановки и эксплуатации программ, правила структурного построения ПС в целом и их отдельных компонентов, состав и формы документации. Это позволяет значительно сократить дублирующие разработки, внедрить сборочное программирование и организовать на предприятиях накопление высококачественных программных продуктов. Для того чтобы программное обеспечение (ПО) отвечало всем вышеперечисленным требованиям, процесс его проектирования должен включать следующие этапы:

1. Анализ технических требований – определение функций создаваемой системы.
2. Проектирование – переход от спецификаций функций системы к описанию того, каким образом она будет их выполнять.
3. Реализация – создание реального программного продукта.
4. Аттестация – проверка корректности программной системы.
5. Верификация – формальная проверка того, что созданный программный продукт корректен и удовлетворяет поставленным техническим требованиям.
6. Сопровождение – фиксация технических решений для тех, кто будет сопровождать систему, и создание учебных материалов и инструкций для эксплуатационщиков.
7. Управление проектом – планирование, слежение и руководство при разработке ПО.

В настоящее время эти этапы выполняются самыми разными способами, что находит отражение в широком спектре существующих инструментальных средств САПР ПО. Хотя ПС, создаваемые сегодня, весьма разнообразны, выделился ряд типичных вопросов, которые приходится решать при разработке любого программного продукта. Одна из таких проблем – технология разработки ПС [1-2]. Выбор технологии становится ощутимым, когда объем ПС исчис-

ляется сотнями тысяч команд, что характерно для современных систем. В результате приходится решать следующие задачи:

– планирование и организация всего технологического процесса проектирования ПС;

– разработка математических моделей алгоритмов и других компонентов системы на всех стадиях их проектирования;

– создание алгоритмов, включающих задачи автоматизации самого процесса разработки, унификации типовых частей программ и т.д.;

– отладка программ с различными методами их контроля, обнаружения, диагностики ошибок и способами корректировки ПМ;

– проведение испытаний программных компонентов и всего комплекса;

– автоматизация изготовления необходимой документации.

При создании ПО компьютерных систем одним из первых вопросов, встающих перед разработчиками, является вопрос сложности ПО. Аспект сложности ПО включает две составляющие. Первая из них зависит от того, насколько сложна решаемая задача, а вторая связана с процессом проектирования. Автоматизация проектирования, если использовать этот термин в широком смысле слова, введена для того, чтобы помочь справляться со всеми аспектами сложности. Независимо от области, к которой относится задача, и от уровня абстракции система, предназначенная для автоматизации проектирования, должна поддерживать выполнение ряда функций:

1. Ввод описания проекта – обеспечение ввода проекта в систему.

2. Просмотр-обход – предоставление пользователю возможности выбора нужных фрагментов проекта и формирования необходимых запросов.

3. Декомпозиция проекта – разбиение его на части с целью представления в виде удобных для обработки частей.

4. Моделирование – представление проекта на таком уровне абстракции, который позволяет лучше понять его поведенческие характеристики.

5. Синтез – преобразование проекта из одного вида представления в другой, как правило, с переходом на более низкий уровень абстракции.

6. Управление процессом разработки, которое помогает разработчикам проекта оставаться в рамках выбранной инженерной методологии.

Значительное большинство сегодняшних инструментальных средств САПР ПО для анализа и синтеза поддерживают только часть из перечисленных функций. Поэтому считается, что создание высококачественных комплексов для разработки ПО является длительным итеративным процессом. Подобные системы позволят разработчикам определить те виды экономически оправданных технологий, которые обеспечат выполнение необходимых функций и помогут потенциальным заказчикам выработать приоритеты в некоторых случаях противоречивых требований.

1.2. Модульный подход к разработке программных средств

Отвечая на вопрос, связанный с выбором технологии проектирования программных продуктов при создании САПР ПО, остановимся на синтезе ПО, основанном на модульном принципе [3-5]. Современные ЭВМ позволяют хранить в базах данных практически неограниченные объемы информации, для чего необходимо выполнить оформление программ в виде модулей, отвечающих некоторым стандартным соглашениям. Это обеспечивает возможность соединения ПМ для решения более крупных задач. Модульный принцип сочетается с синтезом программ на основе ориентированных графов. Заданный маршрут на графе ассоциируется с решением поставленной задачи, при этом следует учитывать вопросы передачи параметров между ПМ, поскольку для их совместного использования необходима программная стыковка модулей. Алгоритм построения и изображение графов могут быть самыми различными, но смысловая нагрузка остается неизменной: порядок вызова программных модулей, входящих в синтезируемую программу. Кроме этого, для решения задачи составляется управляющая программа, которая в надлежащем порядке вызывает все программные единицы на выполнение. Построение управляющей программы требует детальных знаний свойств модулей, и прежде всего знания функций, реализуемых ПМ. С ростом количества информации, хранимой в БД, выбор ПМ усложняется. Поэтому с целью облегчения подбора модулей они снабжаются паспортами, в которых описываются их характеристики: идентификатор, назначение, язык программирования, способ обращения, входные и выходные данные с указанием их формата, технические сведения, параметры настройки.

Модульное программирование подразумевает умение разбить задачу на некоторое число самостоятельных подзадач, широкое использование стандартных модулей путем их параметрической настройки, автоматизацию сборки готовых модулей, создание и управление библиотеками готовых модулей. В модульной технологии широко применяется техника макрогенераций. Принцип модульности требует, чтобы программное обеспечение удовлетворяло жестким требованиям стандартизации, унификации, а CASE-средства содержали компоненты, позволяющие производить разбиение алгоритмов на модули, проверять эффективность модуляризации, находить оптимальную степень интеграции модулей и их иерархичность.

В качестве характеристик модуля используются число информационных входов m, число информационных выходов n, коэффициент заполнения модуля k^l.

$$k^l = \frac{N^l}{N^l_{\max}},$$

где $N^l_{\max}$ – максимальное число элементов модуля. Также учитываются число связей элементов внутри модуля z и число межмодульных соединений K. Об-

щее число элементов обозначается T, число модулей уровня l обозначается I^l.

Задача модуляризации алгоритма состоит в разбиении его на непересекающиеся модули различных уровней таким образом, чтобы $\Phi \to \min$ для всех l из множества {1,2,..,L}, причем

$$D_i^l \text{ I } D_j^l = \varnothing,\ i \neq j,\ \sum_{l=1}^{L} \sum_{i=1}^{I^l} N_i^l = T,$$

где Φ – критерий качества модуляризации рассматриваемой схемы алгоритма; D_i^l, D_j^l – множества элементов i-го и j-го типов, принадлежащих уровню l; N_i^l – число элементов этого уровня; L – число уровней; T – число элементов программного средства.

Степенью интеграции модуля уровня l называется величина

$$s^l = \lg N^l,$$

где N^l – число его элементов.

Задача модуляризации относится к задачам комбинаторного типа, и ее решение может быть получено путем перебора всех возможных вариантов. Однако их число R при возрастании числа компонентов T в ПС быстро растет. Учитывая это, разбиение схемы на модули методом перебора возможно лишь в простейших случаях. Особенностями задачи модуляризации ПС в отличие от задачи разбиения в конструкторском проектировании являются иерархия модульных уровней и детерминированность связей между элементами блока. Поэтому оптимизация при модуляризации ПС осуществляется как по числу уровней модулей, так и по числу модулей, объему и характеру межмодульных соединений, числу входов-выходов на каждом уровне при условии жестких ограничений, определяемых функциональной законченностью модулей библиотек программ. В процессе анализа решаемых задач в них выявляются наиболее часто используемые ПМ или группы ПМ, они принимаются в качестве регулярных структур ПС. Затем делается попытка покрытия регулярными структурами всего ПС. Формальный поиск регулярных структур на графе ПС организуется по матрицам входных и выходных связей модулей графа или матрице смежности. Модули графа ПС обозначаются:

$$M_i^l = M_i^l(N_i^l, k_i^l, z_i^l, m_i^l, n_i^l),$$

где $l = \overline{1, L}$, $i = \overline{1, I^l}$; N_i^l – число элементов модуля; k_i^l – коэффициент заполнения модуля; z_i^l – число внутренних связей; m_i^l, n_i^l – соответственно число входов и выходов модуля i-го типа уровня l.

Если принять, что J_i^l – это количество модулей типа *i* уровня *l,* тогда сумма модулей в *L*-уровневой модуляризованной структуре ПС определяется как

$$T_V^L = \sum_{l=1}^{L} \sum_{i=1}^{I^l} J_i^l,$$

а общее число межмодульных соединений равно

$$T_S^L = \sum_{l=1}^{L} \sum_{i=1}^{I^l} n_i^l + m_{вх},$$

где $m_{вх}$ – число входов ПС. При оптимизации структуры требуется, чтобы $T_V^L \to \min, T_S^L \to \min,$ а также учитываются такие характеристики, как N_i^l и k_i^l. В результате приходится решать задачу многокритериальной оптимизации. Для упрощения алгоритма выделения изоморфных подграфов на графе ПС обеспечивается упорядоченное кодирование вершин графа ПС, например растровое кодирование, при котором на графе ПС образуются изоморфные клеточные формы, и задача модуляризации ПС сводится к поиску клеточных форм различных уровней иерархии.

В процессе решения задачи знания пользователя о ее предметной области, а также данные из паспортов программных модулей объединяются в вычислительную модель (ВМ), описывающую семантику решаемой задачи [6] . Следовательно, если для расчета некоторого объекта имеются модули, то может быть построена его абстракция в виде ВМ. Примерами объектов, которые представлены таким способом в разных системах программирования, являются: турбина, атомный реактор, случайные величины с заданными распределениями. Рассмотрим метод синтеза ВМ на основе управляющего графа, что представляется оптимальной формой, поскольку графы – это универсальные математические модели.

В результате программные средства, создаваемые на основе технологии модульного проектирования, соответствуют основным требованиям, предъявляемым САПР к программному обеспечению.

1.3. Основные этапы разработки программных средств

На пути к достижению комплексного подхода при разработке программных средств широкое применение получили CASE-средства, обеспечивающие поддержку многочисленных технологий проектирования информационных сис-

тем, охватывая всевозможные средства автоматизации и весь жизненный цикл программного обеспечения. Диапазон CASE-средств очень велик, и сегодня практически каждое из них располагает мощной инструментальной базой. Весь процесс создания прикладных систем при помощи комплексов инструментальных средств поддержки технологий автоматизированного проектирования (ИСПТАП) можно разбить на следующие основные этапы: составление технического задания на проектирование системы; построение декомпозиционной модели системы; концептуальное проектирование; синтез вычислительной модели; построение программного кода на заданном языке программирования.

Вследствие того, что комплексы автоматизированного проектирования ориентируются на коллективную работу проектировщиков, инструментальные средства поддержки процессов программирования содержат компоненты, позволяющие разработчикам использовать коллективные базы данных, осуществлять взаимодействие на уровне обмена библиотеками, управления работой. При организации вычислительных процессов актуальными являются вопросы диспетчеризации, распараллеливания обработки программ, синхронизации, адаптации к аппаратной среде и операционной системе. В результате инструментальные средства поддержки технологий программирования образуют достаточно сложную инфраструктуру.

Рассмотрим комплекс автоматизированного проектирования программного обеспечения (КАПР ПО), поддерживающий выполнение вышеперечисленных этапов разработки программных средств. КАПР ПО представляет собой вычислительный комплекс иерархической структуры, обладает развитыми инструментальными средствами [7-11]. В его состав входят (рис 1.1):

- вычислительные устройства, обеспечивающие обработку и хранение поступающей информации;
- внешние устройства ввода-вывода информации в удобной для проектировщика форме, в качестве вычислительных средств могут использоваться вычислительные системы или локальные вычислительные сети;
- операционная система (ОС), осуществляющая управление вычислительными процессами и организующая работу проектировщика с вычислительными средствами КАПР ПО;
- прикладное программное обеспечение общего назначения (ПО ОН);
- система административного управления процессами автоматизированного проектирования (САУ ПА);
- ИСПТАП – набор инструментальных средств, включающих монитор пользователя (МП), процессор формирования заданий на проектирование (ПФЗП), процессор декомпозиции (ПД), процессор формирования концептуальных моделей (ПФКМ), процессор формирования вычислительных моделей

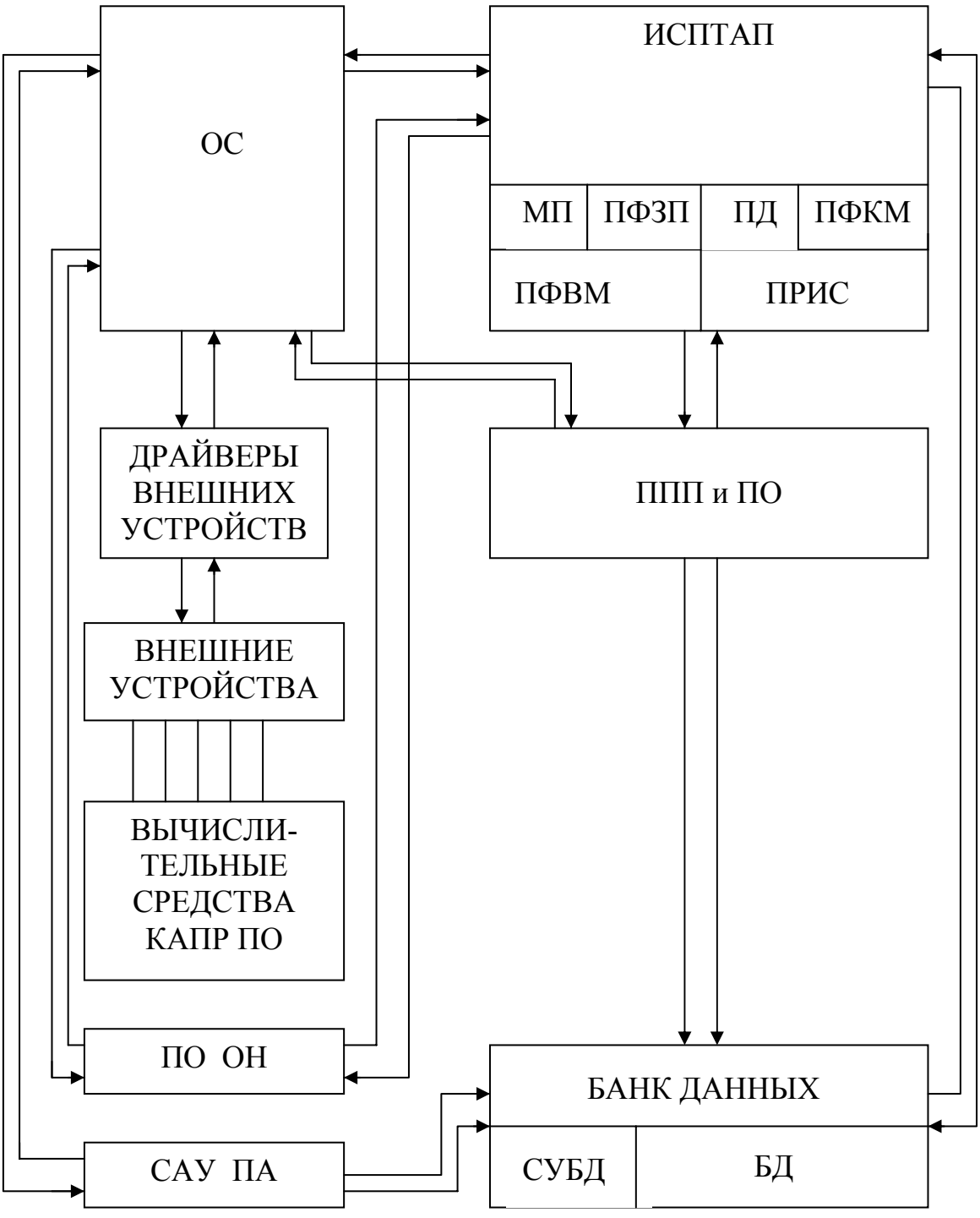


Рис.1.1. Структура ядра КАПР ПО

(ПФВМ), процессор-исполнитель (ПРИС);

- пакеты проектных процедур и проектных операций (ППП и ПО);
- база данных (БД), система управления базой данных (СУБД) и банк данных.

В КАПР ПО допускается одновременная работа нескольких проектировщиков, предусматривается многопользовательская защита данных от несанкционированного доступа, предоставляется развернутая информационно-справочная система (типа HELP). При выполнении сложных программ, требующих значительных объемов памяти, процессор-исполнитель позволяет осуществлять распределенную обработку информации, привлекая сетевые средства.

База данных КАПР ПО представляет собой совокупность личных баз данных разработчиков и системной базы данных. Такое разделение обусловливается различными функциями, реализуемыми каждой отдельной базой данных. Процесс автоматизированного проектирования программных систем носит итерационный характер с неоднократным изменением структуры ПО, ее различных моделей на разных стадиях разработки. Это требует наличия определенной базы данных с небольшим кругом конечных пользователей. Информация, хранящаяся в базе данных разработчика (БДР), не является окончательным продуктом до завершения разработки заданной системы, но необходима при работе процессоров ИСПТАП. С ее помощью организуется технологическая цепочка проектных процедур и операций, при выполнении которой происходит преобразование информации от исходного технического задания на создание ПО до проведения вычислительных экспериментов и анализа получаемых результатов. Выходная информация каждого процессора ИСПТАП помещается в БДР, откуда она извлекается по мере необходимости следующими процессорами на разных стадиях разработки ПО. База данных проекта (БДП) включает информацию о полностью разработанных элементах, входящих в ПО. Информация в БДП заносится проектировщиками из собственных БДР по мере разработки программных единиц и сопровождаемой документации. Логическая модель БДП включает в себя все компоненты БДР, за исключением объектов, отражающих связь ИСПТАП с инструментальной системой поддержки заданной технологии программирования. Система управления базами данных функционирует совместно с ОС ЭВМ, предоставляет средства интеграции и исключения избыточности в базах данных, средства для обеспечения независимости программ от вида хранения и способов организации данных на физических носителях, универсальные средства доступа к данным, контроля сообщений, анализа ошибок и их обработки.

База данных ядра КАПР ПО является составной частью банка данных, в который также входят специальные языковые, программные, организационные, технические средства, предназначенные для централизованного формирования, накопления, ведения и коллективного использования данных. Допустимые стратегии распределения данных определяются архитектурой КАПР ПО и программным обеспечением системы управления базами данных. Статус поль-

зователя определяется администратором комплекса и может изменяться в процессе работы.

Создание вычислительных моделей разрабатываемых объектов занимает центральное место среди остальных стадий. На этом этапе осуществляется переход от алгоритмов к построению программ на заданных языках программирования. Инфраструктура синтеза вычислительных моделей (ИСВМ) реализуется процессором формирования ВМ (на рис. 1.2 показана последовательность синтеза ВМ), который предоставляет пользователю выполнение следующих функций:

- выбор концептуальных моделей проектируемых единиц;
- поиск и выбор аналогов ВМ, создание библиотеки аналогов;
- построение управляющих графов синтезируемых объектов;
- выделение последовательных и параллельных процессов из управляющих графов;
- замена концептуальных модулей на программные единицы;
- сопряжение программ;
- синтез программ;
- проверка корректности синтезируемых объектов с дальнейшим синтезом вычислительных моделей;
- графическое представление вычислительных моделей и их редактирование;
- каталогизация вычислительных моделей;
- работа с базами данных;
- документирование.

Входными данными ПФВМ являются:

- перечень заданий на разработку вычислительных моделей;
- концептуальные модели программных модулей;
- вычислительные модели-аналоги;
- данные спецификаций программных модулей;
- технологические сценарии процессов формирования и редактирования вычислительных моделей;
- значения параметров директив пользователя;
- ответы пользователя на запросы ПФВМ.

Выходными данными ПФВМ являются:

- запросы и сообщения ПФВМ пользователю;
- вычислительные модели, размещаемые в БДР, БДП и подлежащие исполнению;
- вычислительные модели, размещаемые для хранения в распределенной БД;
- исходные тексты программ.

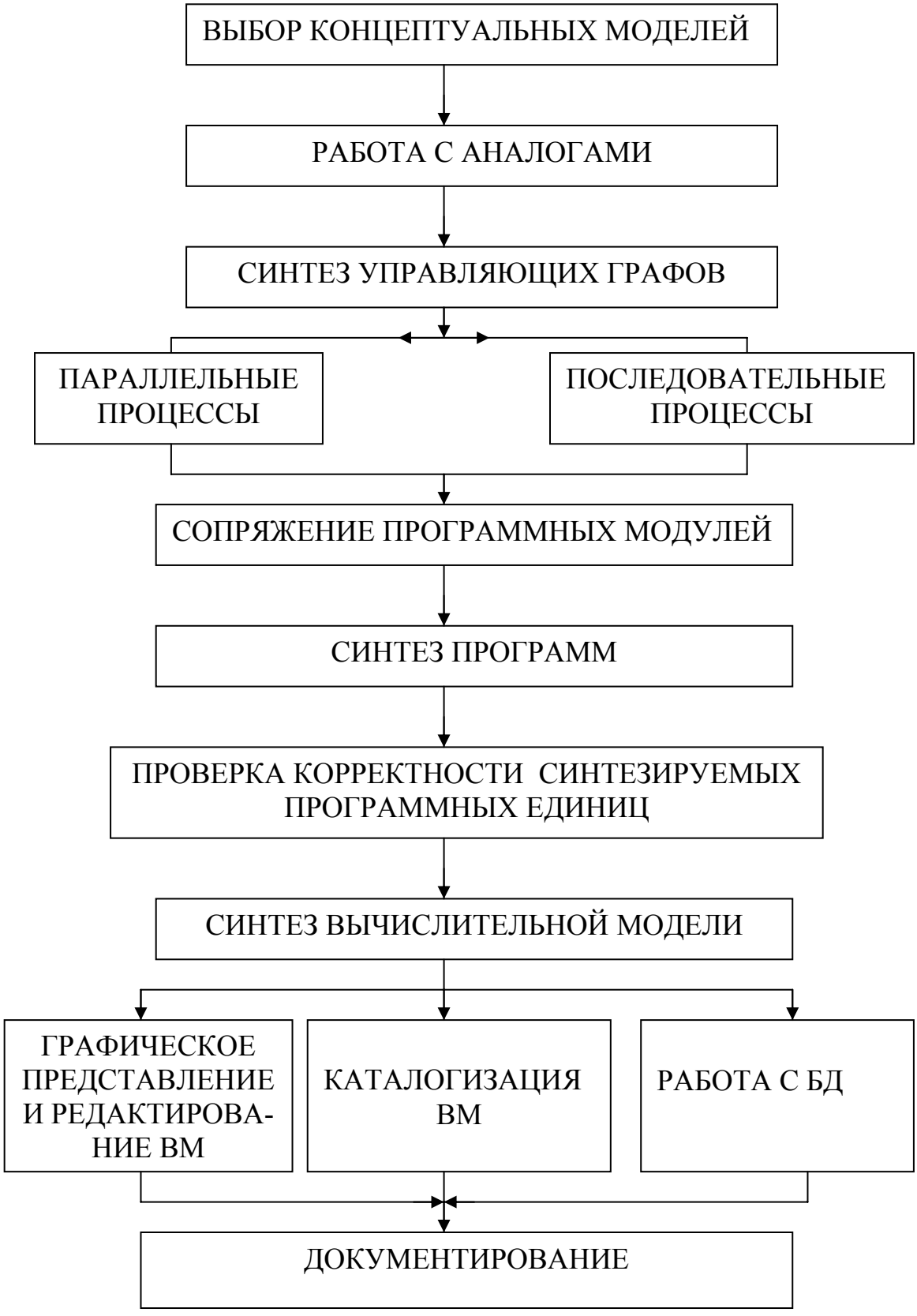


Рис. 1.2. Технология синтеза ВМ средствами ПФВМ

Вычислительные модели определяют асинхронные параллельные действия, которые должны выполняться для вызова требуемых программных модулей и корректной передачи данных. Данные со значениями любого из выходных параметров программного модуля (или элемента концептуальной модели) могут использоваться одним или несколькими другими модулями (или элементами концептуальной модели). Так как порядок обработки выходных параметров многими пользователями при асинхронной реализации параллельных действий исполнения вычислительной модели заранее предусмотреть невозможно, необходима буферизация промежуточных результатов вычислительного эксперимента. Удобно выделить отдельные ячейки для различных выходных параметров, что позволит ставить в соответствие типам значений входных и выходных параметров, связанных с одной ячейкой, тип данных, хранимых в этой ячейке, и тем самым решать задачу согласования типов данных программных модулей. Включение в вычислительную модель именованных буферов для хранения промежуточных данных вычислительного эксперимента превращает ее в сеть с двумя типами узлов: узлами-модулями, в которых определены действия по реализации вычислительного эксперимента, и интерфейсными узлами буферизации данных. Такая сеть может рассматриваться как одна из разновидностей расширенных интерпретированных сетей Петри.

При отображении в ВМ методики решения задачи исследовательского проектирования может возникнуть необходимость отобразить альтернативные варианты развития процесса решения задачи в зависимости от значений ее промежуточных данных. В случае рассмотрения нескольких вариантов реализации одного или нескольких компонентов сложного технического объекта (СТО) вероятно понадобится включить в одну вычислительную модель несколько альтернативных частных моделей или альтернативных вызовов программных модулей. При построении ВМ возможна ситуация, когда вычислительный эксперимент будет иметь несколько различных вариантов развития в зависимости от кода завершения работы вызванного программного модуля. Для быстрого реагирования на появление исключительной ситуации в ВМ предусмотрены альтернативные выходы из узлов-модулей, это влечет за собой необходимость описания альтернативных входов в узлы ВМ с целью сокращения размерности модели. Композиция вычислительных моделей из частных ВМ, соответствующих моделям отдельных компонентов СТО или ВМ проектных процедур, с интерпретацией этой совокупности моделей в рамках одного вычислительного эксперимента предполагает рассмотрение таких конструкций, как иерархии с вложением частных моделей в более общие. На верхних уровнях указанной модели ссылка на частные модели нижних уровней делается в специальных узлах вызова ВМ. На рис. 1.3 приведена обобщенная инфраструктура синтеза вычислительных моделей. Управление ИСВМ осуществляется системой мониторов. Монитор верхнего уровня представляет собой процессор формирования вычислительных моделей, связанный с рядом подсистем, а также запускающий монитор функциональных систем и систему выбора вычислительных моделей как из БДР, так и из СРБД.

Рис. 1.3. Инфраструктура синтеза вычислительных моделей

Мониторы функциональных систем построены аналогично монитору верхнего уровня и управляют работой подсистем выбора заданий (ПВЗ) вычислительных моделей проектных процедур и операций (ВМПП), формирования и поиска аналогов (ПФиПА) ВМПП, каталогизации (ПК) ПМ, автоматического синтеза (АС) модели, полуавтоматического синтеза (ПАС) модели, контроля синтеза (ПКС) системы, печати и защиты от несанкционированного доступа.

Подсистема выбора заданий предоставляет пользователю перечень имеющихся заданий, из которых он должен выбрать то, по которому будет осуществляться синтез системы. При желании он может вызвать концептуальную модель системы и описание ее ПМ.

Подсистема формирования и поиска аналогов обеспечивает оперативный просмотр файла паспортов имеющихся программных модулей и построение их каталога, после чего система поиска аналогов осуществляет поиск и вывод подходящих аналогов ВМПП. В ПФПА предусмотрены формирование аналогов ВМПП, поиск аналогов по идентификаторам и ключевым словам, граф-схем алгоритмов, структур программных систем, описаний аналогов и программных единиц, удаление аналогов из БД.

Подсистема каталогизации программных модулей выполняет построение каталогов, которые формируются на основании паспортов ПМ и их технических заданий. ПК ПМ поддерживает возможности просмотра каталога ПМ и вывода заданных описаний ПМ. Подсистема автоматического синтеза заданных объектов выполняет построение программного кода системы на основе управляющего графа ППП, сформированного при помощи графа связей ПМ прикладных программ. Синтез может осуществляться по следующим критериям: минимум объема памяти ЭВМ, минимум времени решения задачи, минимум исходных данных. Предусмотрен режим многокритериальной оптимизации.

В подсистеме полуавтоматического синтеза построение управляющих графов решаемой задачи осуществляется по требованиям, задаваемым пользователем. Модуль сопряжения ПМ обеспечивает согласование взаимодействующих программных модулей по входам и выходам. Модуль формирования остова синтезируемой задачи автоматически строит скелет программы на алгоритмическом языке. Модуль формирования исходной программы автоматически вырабатывает программу, пригодную для компиляции в заданной системе программирования.

Подсистема контроля синтезированной программы обеспечивает полный контроль корректности работы построенной программы или контроль по отдельным параметрам: логике программы, семантике переменных при передаче информации, типам данных.

Существует еще ряд подсистем (мониторов), входящих в КАПР ПО, работа которых влияет на синтез вычислительных моделей проектируемых систем. Подсистема создания и редактирования ВМПП предназначена для поддержки процессов построения и коррекции ВМПП. Её функциональные части выполняют формирование графов ВМПП, описаний программных модулей и ВМПП, удалений ВМПП из БДР, печать требуемых ВМПП. Подсистема ра-

ботает с данными, которые доступны также и системе поиска аналогов. Информация для функциональных систем КАПР ПО поступает из БДР, в которой имеются файлы описаний концептуальных моделей ПС, структур ПС, управляющих графов вычислительных моделей, паспортов программных модулей. Заполнение файлов осуществляется в инфраструктуре синтеза вычислительных моделей. По желанию пользователя любая системная информация может быть перенесена в БДР. Подсистема выбора вычислительных моделей обеспечивает вывод на экран дисплея перечня ВМПП для конкретного пользователя и всех необходимых элементов ВМПП (общего описания, графа, структуры ВМПП, описаний программных модулей). Подсистема планирования вычислительных процессов включает узлы для построения последовательных и параллельных процессов, развивающихся в вычислительной сети. В системе планирования осуществляются упорядочение программных модулей, образующих остов ВМ, поиск возможных маршрутов для развития вычислительных процессов. В результате в специальный файл записывается матрица вычислительного процесса. Монитор системы управления вычислительными процессами построен по принципу монитора верхнего уровня и также допускает формирование гибких технологических линий моделирования сложных технических систем. Монитор содержит подсистемы активизации заданного вычислительного процесса (ПАВП), исправлений, останова и прекращения вычислительных процессов, подсистемы графического отображения, печати и распределенной обработки (РО). РО становится задействованной при проведении экспериментов с использованием ресурсов вычислительной сети. Информацию о характере запускаемого процесса РО получает из ПАВП и временно хранит в файле моделей распределенной обработки. Данные, необходимые для программных единиц исполнительной модели, поступают от проектировщика непосредственно при вычислениях или по его указанию извлекаются из соответствующих узлов комплекса автоматизированного проектирования. РО может быть связана с системами распределенной обработки остальных узлов системы, обеспечивая тем самым запуск и выполнение программ в удаленных от пользователя узлах КАПР. Отличительными особенностями ИСВМ в РО являются управление запуском и исполнением программ в удаленных узлах вычислительной сети и возврат результатов пользователю. При этом допускается как выполнение других программ в узле-источнике, так и информирование в фоновом режиме о ходе эксперимента в узлах-приемниках.

Непосредственное управление вычислительным экспериментом осуществляется процессором-исполнителем (ПРИС). ПРИС имеет двухуровневую архитектуру, причем монитор верхнего уровня управляет системой выбора вычислительных моделей, системой планирования вычислительных процессов, монитором системы управления вычислительными процессами при проведении математических экспериментов, системами каталогизации исполнительных моделей, пересылки и распределенной обработки.

1.4. Методы разработки программных средств

Рассмотрим два подхода (метода) к созданию прикладных программных систем и условно назовем их алгоритмический и ориентированный на жесткую спецификацию данных.

Алгоритмический метод включает следующие этапы: разработка технического задания, построение декомпозиционной модели (ДМ), построение концептуальной модели (КМ), синтез вычислительной модели (ВМ), построение исполнительной модели (ИМ).

Разработка технического задания начинается с изучения и анализа области решаемой задачи, после чего оговариваются требования к будущей разработке, происходит описание и уточнение исходных данных проектирования. Этап завершается составлением спецификаций, согласованием их с заказчиком и документированием технического задания на проект.

Декомпозиционное проектирование связано с изучением разрабатываемого объекта, поэтому оно не обходится без построения его математической модели. В качестве математического аппарата для моделирования в работе выбрана динамическая модель представления данных и знаний – семантический полигон (СП), позволяющий всесторонне исследовать объект проектирования. Далее будут рассмотрены определение, построение и возможности СП. На этапе декомпозиции системы прорабатывается ее структура. По завершении декомпозиции система представлена на таком уровне абстракции, который включает наиболее общие особенности исследуемого объекта.

Концептуальное проектирование предполагает проработку системы до уровня программных модулей, в результате чего строится графовая модель, которая описывается для каждого уровня декомпозиционного дерева. Подсистема алгоритмизации задач предоставляет пользователю следующие возможности: поиск аналогов и их использование при решении задач алгоритмизации, редукция граф-схемы алгоритма. Графовая модель синтезируется в виде сети Петри и содержит множество реализаций искомого алгоритма, причем некоторые из них могут оказаться неподходящими в соответствии с техническим заданием или вообще неверными. Поэтому в данном случае наиболее важным этапом для разработчика является выбор корректного решения. Выбор осложняется анализом ряда альтернатив, и поскольку моделирование ориентируется на алгоритм, а не на данные, могут возникнуть проблемы с входными и выходными параметрами ПМ при их стыковке. Это является основным недостатком алгоритмического метода при создании программных средств.

Второй из указанных методов содержит следующие этапы проектирования: разработка технического задания, концептуальное моделирование, в случае необходимости декомпозиция концептуальной модели, синтез вычислительной модели, построение исполнительной модели.

Разработка ТЗ аналогична предыдущему методу, только особое внимание здесь уделяется описанию исходных и выходных данных проекта. Концептуальное проектирование заключается в построении графовой модели (У-сети) системы, содержащей алгоритм, по которому выполняется синтез ВМ. Синтез

У-сети начинается с анализа выходных данных проекта. Затем в БД готовых ПМ (примитивов) выполняется поиск тех из них, у которых выходные параметры аналогичны или совпадают с выходными данными проекта. Найденные модули составят первый уровень концептуальной модели. Далее анализируются входные параметры модулей первого уровня и из БД выбираются примитивы, чьи выходные данные могут быть входными параметрами вершин предыдущего уровня. Каждый из выбранных модулей состыковывается с соответствующим ему модулем первого уровня, в результате концептуальная модель имеет два уровня синтеза. Процесс поиска ПМ продолжается как минимум до тех пор, пока по какому-нибудь маршруту У-сети не будет выстроена связь между вершинами первого уровня синтеза и исходными данными системы, которые реализуются модулями последнего *n*-го уровня разработки. В том случае, когда *n*-й уровень синтеза не достигнут, и ни для одного из маршрутов не были найдены подходящие ПМ, требуется переходить к этапу декомпозиции концептуальной модели. Для этого анализируются построенные цепочки маршрутов, а затем один из них завершается путем условного соединения с исходными данными проекта. Дальнейший синтез КМ выполняется внутри звена условной связи. В процессе декомпозиции уровень абстракции условной связи доводится до стадии, когда ее можно реализовать примитивами из данной предметной области, стыкуя их с учетом входных и выходных параметров. В результате условная связь становится реальной. Если в процессе синтеза КМ получено несколько альтернативных маршрутов, то выбор того, по которому будет выполняться синтез ВМ, происходит на основании заданного критерия оптимизации.

В ходе построения КМ разработчик, находясь на произвольном уровне синтеза, имеет возможность выполнить анализ У-сети. Это помогает уже на ранних этапах синтеза КМ исключить из рассмотрения часть маршрутов и сделать построение КМ более качественным. Если необходимо выполнять декомпозицию КМ, то в результате анализа У-сети активными остается только часть альтернатив, что также упрощает дальнейший синтез КМ. Прежде чем переходить к построению ВМ, удобно провести окончательный анализ полученных маршрутов.

Синтез ВМ развивается в обратном порядке относительно КМ: обработка У-сети идет от *n* уровня к первому. По сравнению с алгоритмическим методом данный подход является более формализованным, что позволяет максимально автоматизировать построение всех моделей и приблизить алгоритм к требованиям технического задания. Изложенный метод позволяет специалисту, владеющему только предметной областью задачи, строить цепочки решений. Для этого необходимо разбить поиск решения на этапы и, зная, что должно являться результатом каждого из этапов, искать нужные ПМ в БД, читая их паспорта. И, наоборот, поиск решения может выполнять программист, осуществляющий связывание ПМ формально по их входным и выходным характеристикам. Сотрудничество пользователей двух типов необходимо в том случае, когда есть альтернативные ПМ для одной задачи или требуемые программные единицы не найдены.

1.5. Модель представления данных и знаний - семантический полигон

Семантический полигон – это среда, содержащая совокупность информационных стационарных и мобильных объектов, обладающих определенными свойствами и обеспечивающих решение поставленных задач. Возможности полигона используются для принятия согласованных и выбора оптимальных решений, исследования развивающихся на полигоне процессов во времени и в пространстве [12-14]. Для работы с семантическими полигонами введём несколько определений.

Определение 1. Абстрактным информационным стационарным объектом (СО) называется упорядоченное множество соединенных между собой информационных элементов вида B(W,R,S,A), где

B – идентификатор объекта; W – структура объекта с выделением множества его входов X и множества выходов Y; R – репозиторий; S – множество состояний; A – алгоритм функционирования СО.

Идентификатор задает имя объекта, под которым он функционирует на СП. Множество входов и выходов является упорядоченной совокупностью переменных, определяющих тип, формат, допустимый объем входной и выходной информации, доступной объекту и контролируемой им. Структура отражает информационные элементы объекта со связями между ними. Репозиторий служит для хранения необходимой информации об объекте. Он характеризуется объемом, средствами хранения и управления данными, обеспечивает доступ, обновление, анализ и визуализацию всей информации. Состояние СО – это определенный набор значений всех его информационных элементов. Переменная S отражает текущее состояние объекта. Информация о состояниях, в которых находился СО, хранится в репозитории, поэтому, проанализировав S и данные из R, можно спрогнозировать состояние объекта. Переменная A определяет алгоритм работы объекта B с учетом его состояния и входных данных.

Рассмотрим пример стационарного объекта, относящегося к области систем автоматического управления (САУ). На рис. 1.4 изображен объект, классифицирующий САУ по признаку “линейна-нелинейна”. Основные знания относительно этих понятий теории автоматического управления можно упорядочить следующим образом: b_1 – свойство гомогенности; b_2 – свойство аддитивности; b_3 – свойство линейности; b_4 – свойство нелинейности [15]. Алгоритм работы объекта B_1 следующий: осуществляется проверка на выполнение одновременно свойств гомогенности и аддитивности системы, если результат положительный, система – линейна, в противном случае САУ – нелинейна. В зависимости от входных данных информационные элементы СО принимают конкретные значения, что фиксируется как текущее состояние объекта, связывается с переменной S, а затем может быть помещено в репозиторий. Структура объекта B_1 включает следующие элементы $W = \{b_1, b_2, b_3, b_4, x_1^1, x_2^1, y_1^1, y_2^1\}$, множество состояний описывается двумя значениями, $S = \{0,1\}$, где 0 – признак линейности САУ, 1 – признак нелинейности системы.

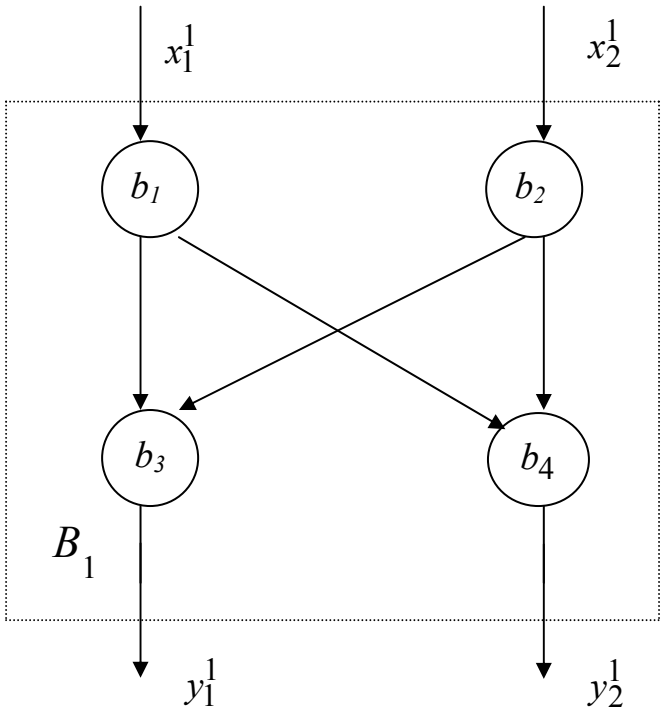


Рис.1.4. Стационарный объект

Определение 2. Абстрактным информационным мобильным объектом (МО) называется объект вида M(W, S, A, Q, U, Z), способный перемещаться в пространстве семантического полигона и взаимодействовать с его стационарными и мобильными объектами.

Символы W, S и A у МО обозначают те же характеристики, что и у СО. Кроме этого, мобильные объекты располагают: Q – органами автономного движения, U – системой управления движением, Z – системой поддержания работоспособности. Структура МО (мобиля) представлена на рис. 1.5, где ОД – система организации движения мобиля, МР – маршрутизатор, СПР – система принятия решения, СУ – система управления мобилем. Система организации движения осуществляет навигацию мобиля в базе данных (БД), просмотр файлов. Маршрутизатор обеспечивает поиск маршрутов достижения заданной цели в БД. Система принятия решений служит для анализа входных и синтезированных данных, а также выработки решений на их основе, система управления выполняет осуществление принятых решений. Посредством входных переменных вводятся цель движения, описание структуры полигона, описание окружения мобиля и другая необходимая информация для решения конкретной задачи. Роль мобильных объектов на СП выполняют специальные программы, которые воспринимают информацию об окружающих их объектах, анализируют ее и принимают соответствующие решения о дальнейших действиях. Одной из функций мобиля является нахождение возможных маршрутов достижения поставленной цели, т.е. возможных алгоритмов решения задачи.

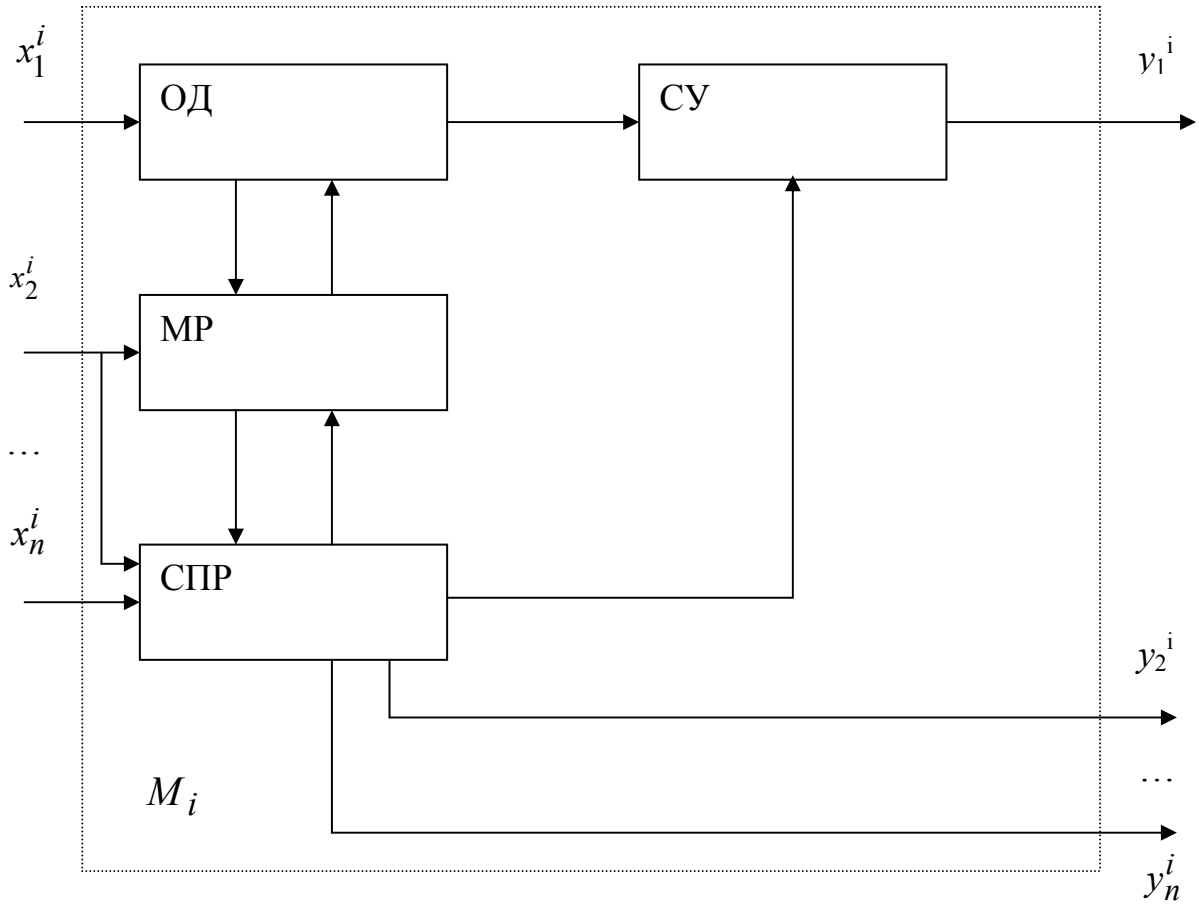


Рис.1.5. Мобильный объект

Определение 3. Семантическим полигоном называется среда, содержащая стационарные объекты и магистрали между ними со средствами управления передачей данных, а также мобильные объекты, способные перемещаться по магистралям и взаимодействовать в некоторой окрестности со своим окружением.

СП задается в виде P (X, Y, B, M, R, C), где X – множество входов полигона, Y – множество выходов полигона, B – множество стационарных объектов полигона, M – множество мобильных объектов, R – система регламентации функционирования СП, C – система центрального управления.

Построение СП состоит из двух этапов, каждый из которых выполняется при помощи мобильных объектов определенного типа. По функциональному назначению различаются мобили первого и второго типов. Синтез СП начинается с определения магистралей (связей) между стационарными объектами. Для этого мобили первого типа анализируют свойства каждой пары стационарных объектов и принимают решение о том, возможны ли между ними связи, причем соединения могут быть не единичными, так как образовываются по ряду свойств. Затем запускаются мобили второго типа, которые перемещаются по синтезированным маршрутам полигона с целью выявления некорректных связей. Они хранятся в специальной таблице и представляют собой записи, содержащие недопустимые последовательности стационарных объектов. Эти мо-

били имеют радиус обзора больше единицы, благодаря чему они "видят" выстроенные последовательности СО, которые анализируют на предмет корректности. Чем больше видимость МО, тем более длинные цепочки СО они способны анализировать. МО второго типа могут запускаться неоднократно с целью всестороннего изучения модели, сбора статистической информации, поиска оптимальных маршрутов и т. д. В качестве примера рассмотрим модель семантического полигона, изображенного на рис. 1.6. Данный полигон отражает второй из вышеописанных этапов построения СП, когда связи между СО установлены. После того, как модель признана корректной, составляются матрицы входных и выходных связей стационарных объектов, отражающие во времени зависимости между ними. Предположим, что на рис. 1.6. все связи корректны и каждая цепочка, начинающаяся с входных данных и заканчивающаяся получением результатов, соответствует решению определенной задачи, относящейся к теории автоматического управления. Задачи, рассматриваемые на СП, будем называть вычислительными процессами. Реализация алгоритма одного стационарного объекта с учетом входных данных – это простейший вычислительный процесс, выполняющийся на СП. Таким образом, решение различных задач на СП организуется путем передачи управления и данных от одних объектов к другим в определенном порядке. Все обозначения на рисунке соответствуют ранее принятым соглашениям. Опишем информационное наполнение стационарных объектов: B_1 – классификация САУ по признаку "линейна-нелинейна"; B_3 – определение типа САУ (непрерывные или дискретные); B_4 – построение передаточной функции системы; B_5 – получение характеристического уравнения САУ; B_6 – тип исследования САУ для определения свойств устойчивости; B_2 – классификация корневых годографов; B_7 – синтез полиномов Харитонова для заданной САУ; B_8 – построение полей корневых годографов; B_9 – нахождение граничных траекторий полей; B_{10} – определение граничных траекторий устойчивости полей корневых годографов. На полигоне присутствуют два мобильных объекта, каждый из которых кроме своей системы управления располагает двусторонней связью с органом центрального управления всего СП. Организация модели позволяет добавлять и стационарные, и динамические объекты. При изменении числа СО требуется запустить мобили первого типа для переустановки всех связей. На СП имеется объект-диспетчер. Взаимодействие мобилей может осуществляться как напрямую, так и через диспетчера, который поддерживает функции информационной службы, контроля и управления. Мобили классифицированы как активные и пассивные. Активные мобили способны самостоятельно перемещаться по выбранным маршрутам. Получив задание, они принимают оптимальное решение и начинают его выполнять. При этом активные мобили наблюдают за обстановкой на СП в определенной зоне обзора, анализируют ситуацию, принимают решения в реальном масштабе времени и исполняют их. Пассивные мобили выполняют движение под управлением диспетчера, который выбирает для них оптимальные маршруты, анализирует текущую обстановку на СП, выдает оперативные указания и решает конфликтные ситуации.

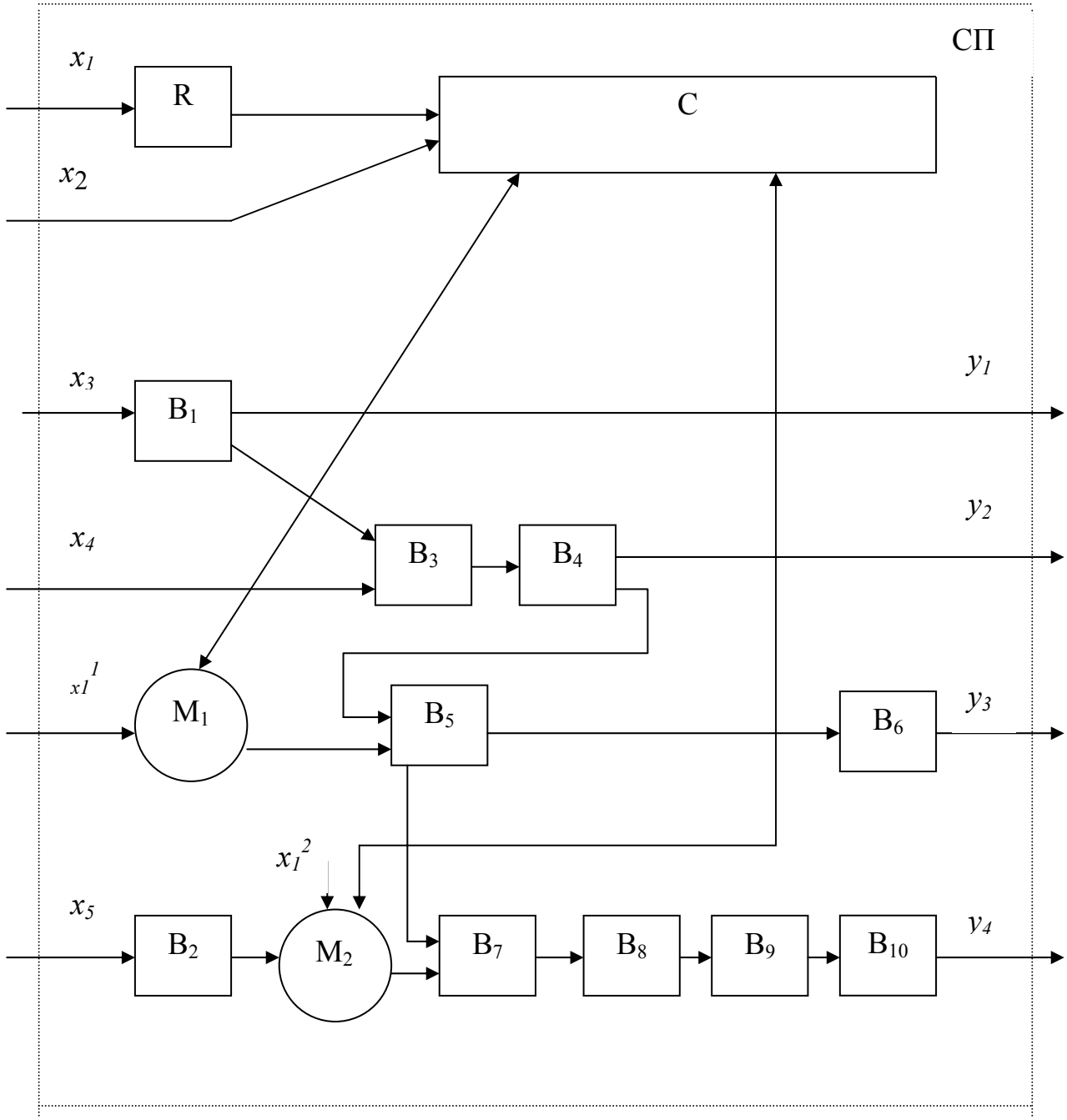


Рис. 1.6. Модель семантического полигона

Изложенная организация СП была выбрана с целью изображения его в виде абстрактной многоуровневой модели, отражающей структуру и протекающие во времени процессы. Значит, семантические полигоны представляют собой модели определенных областей знаний, разделов теорий, дисциплин, методически разделенных, как это принято в рассматриваемой отрасли науки. Причем от степени формализации знаний, точности алгоритмов СО, "интеллекта" мобильных объектов, совершенства инструментальных средств СП зависят эффективность работы полигона и качество решений, получаемых с его помощью.

1.6. Автоматно-временные функции семантических полигонов

При построении семантических полигонов представляет интерес исследование процессов во временной области. Алгоритм работы объектов, а следовательно, и вычислительного процесса, поскольку он состоит из объектов, можно изобразить в виде блок-схемы или управляющего графа (У-графа). Объединение У-графов будем называть У-сетью. Для изучения временных характеристик процессов на СП были предложены автоматно-временные функции (АВФ).

Определение 4. АВФ называется выражение в виде равенства, в левой части которого находится событие, обозначающее результат вычислительного процесса, а в правой части – сумма последовательных и альтернативных событий, протекающих во времени и приводящих к результату.

АВФ строятся по У-графам процессов и являются их отражением во временной области. В каждый момент времени из суммы событий, входящих в АВФ, только одно, соответствующее текущему моменту, отлично от нуля. Этим интерпретация АВФ схожа с принципами работы конечных автоматов.

Рассмотрим У-сеть, изображенную на рис. 1.7. Каждый вычислительный процесс имеет начальную и конечную точки. В данном случае можно получить четыре комбинации типа "начальная вершина – конечная вершина". Следовательно, в У-сети развиваются четыре процесса. Для описания процессов дискретных систем используется метод z-преобразований. В результате того, что процессы управляются логическими переменными, при коэффициентах z-функций присутствуют логические выражения. Операторные вершины У-сети обозначаются переменными Y, условные вершины – переменными X. Время выполнения оператора У-сети считается равным одному такту или T. Условные вершины срабатывают мгновенно, т.е. время на них не тратится. Номера процессов в сети обозначены верхними индексами, номера операторных и условных вершин – нижними индексами. Признаком цикла в процессе является наличие знака Σ в его автоматно-временной функции. В цикл входят операторы, на которые распространяется знак Σ. Время работы цикла определяется нижним и верхним пределами суммы. По определению АВФ строится таким образом, что в левой части равенства фиксируется вершина У-сети, которая является конечной точкой вычислительного процесса, а в правой части равенства – цепочка из последовательности операторов, приводящих к поставленной цели.

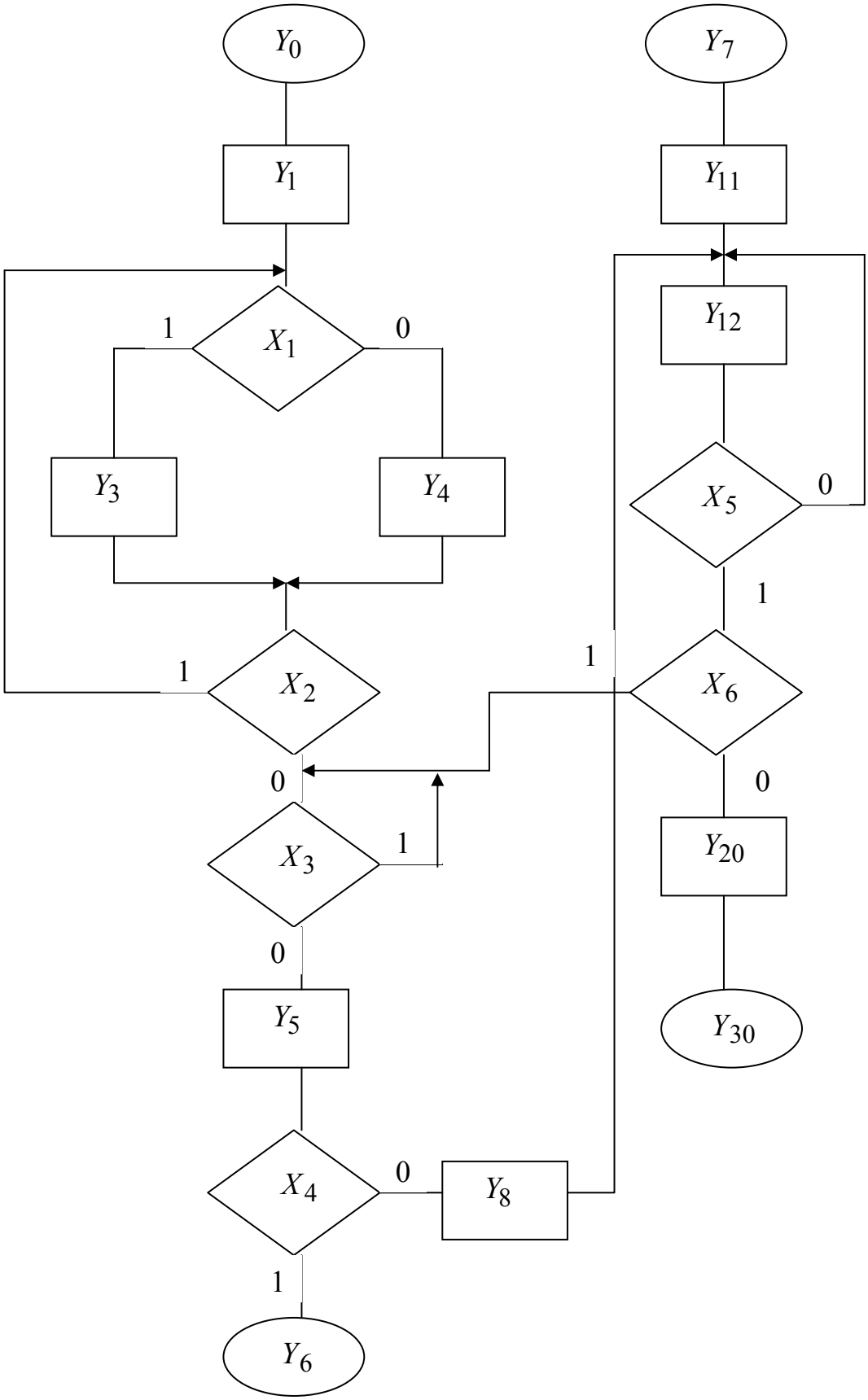


Рис. 1.7. У-сеть семантического полигона

АВФ первого процесса сети, изображенной на рис. 1.7, имеет следующий вид:

$$Y_6^1 = Y_0 z^a + Y_1 z^{a+T} + (X_1 Y_3 z^{a+2T} \vee \overline{X}_1 Y_4 z^{a+2T}) +$$

$$+ (X_2 \sum_{i=0}^{A-1} (X_1 Y_3 z^{a+T(3+i)} \vee \overline{X}_1 Y_4 z^{a+T(3+i)}) \vee \overline{X}_2 \overline{X}_3 Y_5 z^{a+T(3+A)}) +$$

$$+ (\overline{X}_4 Y_8 z^{a+T(4+A)} \vee X_4 Y_6 z^{a+T(4+A)}) + Y_{12} z^{a+T(5+A)} + (\overline{X}_5 \sum_{i=0}^{B-1} Y_{12} z^{a+T(6+A+i)} \vee$$

$$\vee X_5 X_6 \overline{X}_3 \sum_{i=0}^{C-1} (Y_5 z^{a+T(6+A+B+i(3+B))} + (X_4 Y_6 z^{a+T(7+A+B+i(3+B))} \vee$$

$$\vee \overline{X}_4 Y_8 z^{a+T(7+A+B+i(3+B))}) + Y_{12} z^{a+T(8+A+B+i(3+B))} +$$

$$+ \overline{X}_5 \sum_{j=1}^{B-1} Y_{12} z^{a+T(8+A+B+i(3+B)+j)})).$$

Здесь *A, B, C* – числа повторений циклов. Приведенная АВФ описывает процесс, выполняющийся в простейшем случае в течение четырех тактов времени. В эту АВФ не включен оператор X_3, поскольку с ним не связан ни один из *Y*-операторов. Следовательно, X_3 не влияет на построение АВФ. Процесс может иметь три цикла. Первый цикл начинается при выполнении условия X_2 через интервал времени $a+3T$ после запуска сети и повторяется *A* раз. Второй начинается, если значение переменной X_5 – ложно, через интервал времени $a+6T+AT$ после запуска сети и повторяется *B* раз. Так как первый цикл успевает завершиться к моменту начала второго цикла, то к времени его начала прибавляется число тактов первого цикла. Третий цикл начинается при выполнении условий X_5, X_6, и $\overline{X}_3$, повторяясь *C* раз. Последний цикл включает в себя три события, одно из которых – второй цикл. Рассмотрим составляющие третьего цикла и разберем подробнее их показатели времени. В показатель времени при операторе Y_5 входят слагаемые:

- *a, T* – для синхронизации процессов (*a* – момент начала функционирования процесса);
- $6+A+B$ – интервал времени от момента запуска сети до рассматриваемого события при условии существования первого и второго циклов;
- $i(3+B)$ – номер текущей итерации, умноженный на сумму, состоящую из количества операторов цикла и времени выполнения вложенного цикла (на первой итерации это слагаемое будет равно нулю).

Показатели времени у операторов Y_6, Y_8, Y_{12} составляются аналогично. В показатель времени при Y_{12}, когда этот оператор рассматривается как вложен-

ный цикл, добавляется переменная *j* для обозначения текущей итерации внутреннего цикла.

Автоматно-временные функции остальных процессов У-сети строятся аналогично. Для определения состояния процесса в заданный момент времени *t* находится $k^i = t - t_0^i$. Если $k^i < 0$, значит *i*-й процесс еще не начался. Если $k^i = 0$, то состояние процесса определяется первым слагаемым автоматно-временной функции. Если $k^i > 0$, то в момент времени *t* процесс описывается k^i первыми слагаемыми автоматно-временной функции.

Решение сложных прикладных задач на основе семантических полигонов требует наличия аппарата для изучения свойств, возможностей и зависимостей между процессами СП. Анализ процессов полигона включает исследование их структуры, нахождение различных характеристик процессов. Блок-схемы большинства реальных процессов включают циклы, поэтому исследование структуры процессов невозможно без изучения особенностей их циклов. Условия существования циклов описываются при помощи логических выражений, причем условие выполняется, когда значение данного выражения равно единице. Для первого процесса сети, приведенной на рис. 1.7, условие существования первого цикла определяется выражением

$$X_2 \sum_{i=0}^{A-1} (X_1 Y_3 z^{(t_0^1+3T+iT)} \vee \overline{X}_1 Y_4 z^{(t_0^1+3T+iT)}),$$

которое равносильно следующему выражению:

$$X_2 z^{(t_0^1+3T+iT)} \sum_{i=0}^{A-1} (X_1 Y_3 \vee \overline{X}_1 Y_4).$$

Значит, данный цикл выполняется, если

$$X_2 z^{(t_0^1+3T+iT)} = 1, \quad \forall i = \overline{0, A-1}.$$

Для выполнения данного цикла необходимо, чтобы функция X_2 принимала истинное значение в течение *A* тактов, начиная с момента t_0^1+3T. Из показателя z-функции $t_0^1+3T+iT$, характеризующего время выполнения цикла, найдем его период, т.е. количество тактов времени, необходимое для выполнения цикла. Период определим как разность между показателем z-функции на $i+1$ и i проходах цикла. Корректность выбранного способа доказывается методом математической индукции.

Доказательство:

1. Вычислим период цикла для $i=1$: $T_1 = (t_0^1 + 5T) - (t_0^1 + 4T) = T \Rightarrow T_1 = T$.

2. Предположим, что способ справедлив для $i=n$, и период данного цикла при указанном i равен T: $T_1 = (t_0^1 + 3T + (n+1)T) - (t_0^1 + 3T + Tn) = T \Rightarrow T_1 = T$, тогда

докажем справедливость метода для $i=n+1$.

Получаем: $T_1 = (t_0^1 + 3T + (n+2)T) - (t_0^1 + 3T + (n+1)T) = T \Rightarrow T_1 = T.$

Следовательно, период первого цикла равен T. Вследствие того, что время выполнения операторной вершины равно T, период первого цикла, выраженный через Y-операторы, равен одному оператору. Анализ остальных циклов выполняется аналогично.

Достижение эффективной и гибкой работы с семантическими полигонами потребовало обеспечить возможность анализировать и корректировать У-графы отдельных процессов, а следовательно, и всю управляющую сеть. Для этого были введены операции над процессами СП. Основными операциями являются объединение и пересечение процессов. Объединение процессов можно выполнять во времени и в пространстве. Объединение во времени обеспечивает включение в процесс в заданный момент времени нового процесса или его фрагмента. Объединение в пространстве представляет собой объединение У-сетей процессов. Для операции пересечения необходимо задать момент времени от начала функционирования первого из процессов. Пересечение процессов можно рассматривать во времени и в пространстве. Пересечение во времени позволяет выделять фрагменты АВФ, описывающие действия в заданный интервал времени. Пересечение в пространстве обеспечивает выделение одинаковых подграфов У-сети. Операции объединения и пересечения процессов на СП позволяют решать различные оптимизационные задачи и задачи, связанные с ориентацией в программных средах САПР в процессе принятия проектных решений.

1.7. Автоматизированный синтез и ориентация в программных средах САПР

Современные пакеты прикладных программ (ППП), предназначенные для работы в САПР, имеют большой объем, высокую сложность и требуют значительных затрат технических и материальных ресурсов. Ситуация сохраняется на протяжении всего времени существования САПР, что объясняется объективными потребностями промышленного проектирования. Кроме того, создание ППП САПР, осложняясь техническими и финансовыми возможностями организаций, не редко ведется разрозненными подразделениями для дальнейшего использования в конкретной прикладной области. В результате ППП, как правило, являются узкоспециализированными, с ограниченными возможностями, неудобными в эксплуатации. Однако положение может измениться, если будут использованы методы автоматизированного построения и исследования программных сред САПР. Это позволит определять их характеристики и применять одни и те же программные единицы при решении разных задач. Следовательно, возникают проблемы автоматизированного синтеза, ориентации и навигации в программных средах САПР и сопряжения систем, реализующих решение навигационных задач, с машинными системами синтеза программ САПР.

Процесс синтеза программной среды и ориентации в ней свяжем с построением семантического полигона. Поскольку СП позволяет представить среду в виде наглядной графовой структуры, на которой могут выполняться параллельные вычислительные процессы, кроме того, подобную модель удобно анализировать и корректировать. Мобильные объекты первого типа способны на основании входных данных и признаков стационарных объектов автоматически осуществлять связывание СО, а мобили второго типа – автоматически проверять корректность каждой образованной связи. Согласно предварительно заложенной программе, мобили второго типа анализируют связи между СО, воспринимают информацию об окружающих мобильных объектах, обмениваются с ними информацией, обрабатывают полученные данные и принимают решения. В результате их работы на полигоне должны образовываться маршруты, соответствующие решению поставленной задачи. При помощи АВФ удается учитывать влияние параметров объектов друг на друга и их взаимодействие на полигоне. Благодаря этому появляется возможность управления объектами, их адаптации к изменяющимся условиям задачи, разработки алгоритмов автоматизированного проектирования изделий в определенной предметной области, в частности, анализа и синтеза систем автоматического управления.

Будем строить семантический полигон по определению. Тогда получаем

$$P\ (X, Y, B, M, R, C),$$

где X, Y – конечные множества входных и выходных символов; B – конечное множество стационарных объектов СП; M – конечное множество мобильных объектов; R – система регламентации функционирования СП, C – система центрального управления. Входные символы используются для кодирования исходных данных расчетов и инициируемых вычислительных процессов. Выходными символами кодируются результаты расчетов, моделирования.

Семантический полигон, который должен стать моделью программной среды, представляет собой сложное образование. Это связано с большим количеством стационарных объектов и со сложной структурой каждого из них, поскольку пакеты прикладных программ могут насчитывать тысячи единиц. Поэтому определение связей между стационарными объектами полигона является трудоемким процессом, который целесообразно автоматизировать. В качестве входной информации для построения магистралей мобильные объекты первого типа получают сведения о том, по каким признакам необходимо выполнять связывание стационарных объектов. Радиус обзора этих мобилей равен одному шагу, так как в любом случае они проверяют каждую пару программных модулей, пытаясь их связать. Затем по полигону проходят мобили, анализирующие построенную управляющую сеть. Входными данными для них является множество, состоящее из сочетаний связей, признанных запрещенными на полигоне, при данной постановке задачи. Поэтому важен радиус обзора МО. Чем больше обзор мобильного объекта, тем больше информации он получит и тем лучше будет ориентироваться на полигоне.

До начала построения СП имеется множество разрозненных программных модулей (единиц), представленных в виде вершин графовой модели и выполняющих функции стационарных объектов. На первом этапе запускается один или несколько мобилей, которые, анализируя условия для связывания, соединяют пары объектов. В общем случае пара СО может располагать несколькими признаками, по которым следует выполнять связывание, в этом случае между указанными вершинами прокладывается количество дуг по числу совпадающих свойств. Атрибуты СО, по которым выполняется их связывание, характеризуются принадлежностью к входной или выходной группе признаков и имеют идентификаторы. Тогда условие для связывания можно определить следующим образом: если мобиль находит пару программных единиц, у которых есть совпадающие признаки, причем у одного СО данный признак относится к выходным свойствам, а у другого он принадлежит к входным характеристикам, то мобиль связывает эти объекты. Для образования причинно-следственных связей дуга направлена от СО, имеющего рассматриваемый признак в выходных характеристиках, к тому, у которого он является входным признаком. Пример СП после выполнения первого этапа синтеза приведен на рис. 1.8. Структура СП описывается матрицами связей входов и выходов его объектов. В большинстве случаев такая форма записи является наиболее экономичной.

Тогда матрицы связей входов (MSI), выходов (MSO), векторы номеров модулей N, времени обработки TO будут иметь вид

$$N=\begin{bmatrix}1\\2\\3\\4\\5\\6\\7\\8\\9\\10\\11\\12\end{bmatrix};\ MSI=\begin{bmatrix}X_{11}&4&0\\X_{12}&1&0\\X_{22}&2&0\\X_{21}&3&0\\1&0&0\\2&0&0\\3&0&0\\4&6&0\\5&7&10\\6&11&0\\7&12&0\\8&9&0\end{bmatrix};MSO=\begin{bmatrix}2&5\\3&6\\4&7\\1&8\\9&0\\8&10\\9&11\\12&0\\Y_{11}&12\\Y_{12}&9\\Y_{22}&10\\Y_{21}&11\end{bmatrix};\qquad TO=\begin{bmatrix}TO_1\\TO_2\\TO_3\\TO_4\\TO_5\\TO_6\\TO_7\\TO_8\\TO_9\\TO_{10}\\TO_{11}\\TO_{12}\end{bmatrix}.$$

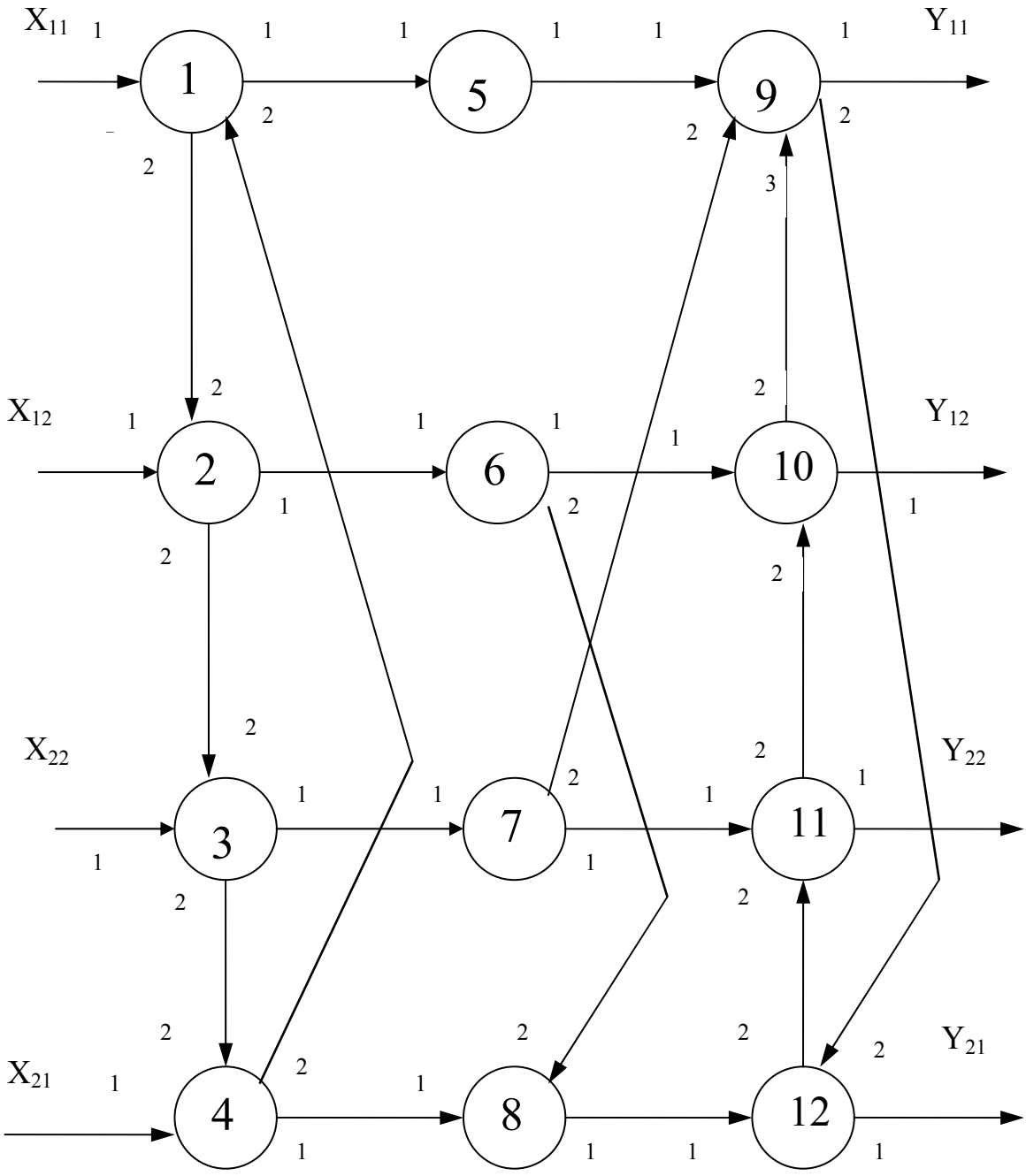


Рис.1.8. Представление У-сети СП.

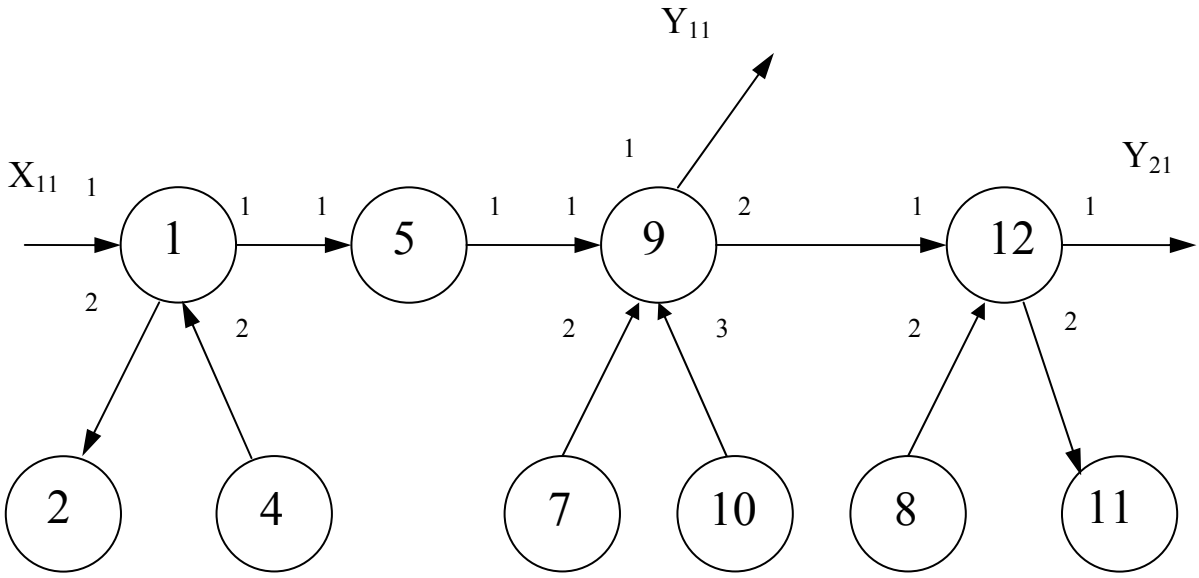
Перейдем ко второму этапу синтеза СП. Пусть мобильный объект $МО_1$ движется по маршруту

$$ММО_1[1,j]=[1\ 5\ 9\ 12],\ j=1,2,3,4,$$

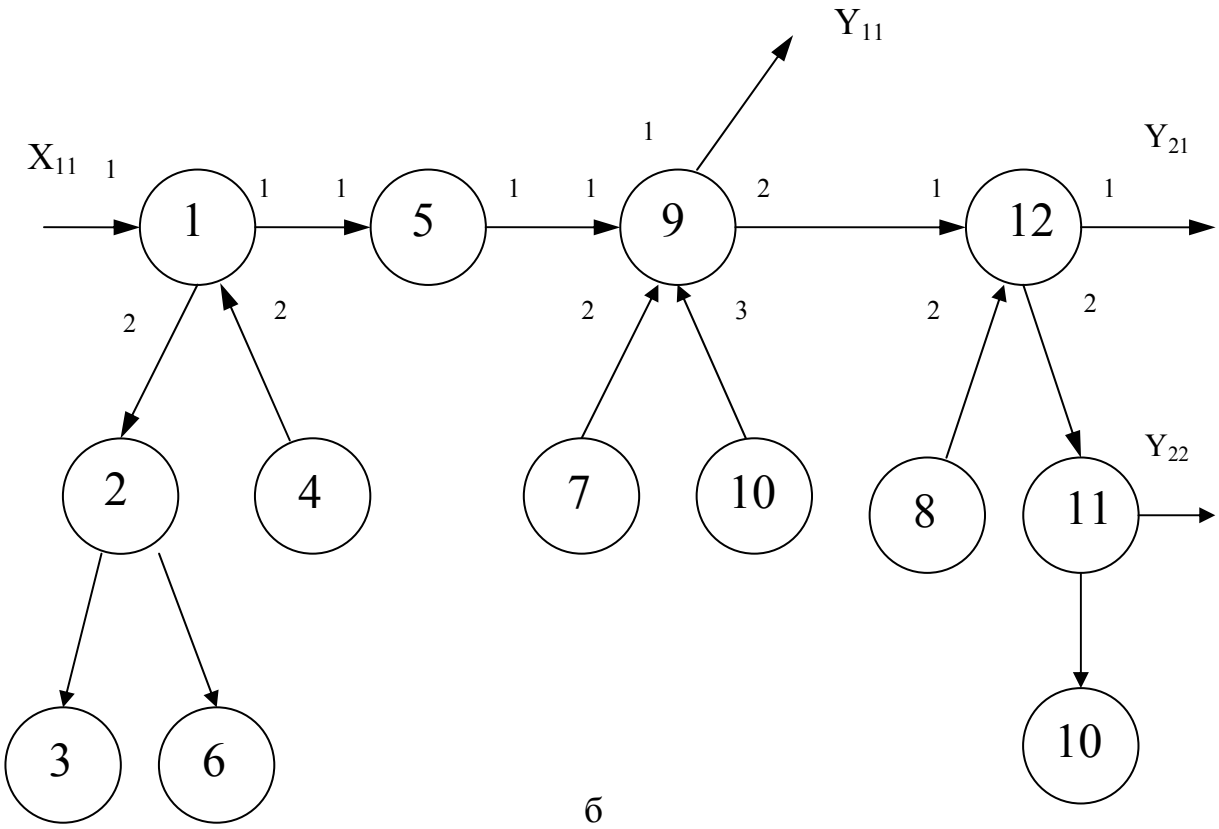
где элементами вектора-строки являются элементы вектора-столбца М[*i*], *i*=1,…,N, т.е. номера модулей У-сети СП. Если мобиль $МО_1$ имеет величину обзора на один шаг вперед (на одну дугу), то, как следует из структуры графа, он сможет раскрыть 13 элементов СП (рис. 1.9, а). Такой шаг означает, что МО пытается пройти по маршруту, не корректируя его, так как не добавляется новая информация. При величине обзора на 2 шага вперед обнаруживаются 17 элементов (рис. 1.9, б). Пусть теперь мобиль $МО_1$ двигается по маршруту

$$ММО_1[2,j]=[1\ 2\ 3\ 4\ 8\ 12],\ j=1,\ 2,\ldots,\ 6,$$

причем величина обзора составляет три шага вперед. Тогда по мере перемещения по маршруту в его поле зрения попадает 36 элементов, хотя граф СП содержит только 20 элементов, т.е. некоторые элементы просматриваются неоднократно. В частности, модули 1, 11, 12, 2, 3, 10, 4 просматриваются дважды, модуль 6 – трижды, модуль 9 – четырежды, по два раза попадают в поле зрения выходные величины $У_{11}$, $У_{12}$, $У_{21}$, $У_{22}$. По существу, маршрут $МО_1[2,j]$ позволяет держать в поле зрения структуру всего полигона в отличие от маршрута $МО_1[1]$, но возникает избыточность информации, которая образуется при увеличении шага обзора модуля. Каждый модуль имеет определенное число входов *l* и выходов *m*. Максимальное число входов у всего множества модулей СП определяет число столбцов матрицы связей входов MSI (три в приведенном примере). Аналогично максимальное число выходов у модулей СП задает число столбцов матрицы связей выходов MSO (два в приведенном примере). При перемещении мобиля некоторые связи могут оказаться некорректными тогда, когда мобиль видит соединения, идущие друг за другом, каждое из которых возможно, а вместе они недопустимы. В этом случае он помечает соответствующие вершины и дуги как "нелогичные", а по результатам анализа перестраиваются матрицы связей. Удаление запрещенных связок из первоначальных матриц связей не выполняется, так как при иной постановке задачи некорректные данные могут оказаться подходящими. В процессе движения мобиля по СП устанавливаются номера входов и выходов модулей, с которыми взаимодействует мобильный объект (на рис. 1.9 – это цифры при дугах графа). Информация, полученная мобилем, позволяет обнаружить петли на графе. Например, двигаясь по второму маршруту, мобиль, пройдя вершины 1, 2, 3, 4, обнаруживает непосредственную связь модуля 4 с модулем 1, что свидетельствует о возможности появления цикла при развитии вычислительного процесса. В то же время не обнаруживается петля 9,10,11,12. Однако при обзоре на четыре шага вперед эта петля также была бы зафиксирована. Следовательно, при целенаправленной организации движения мобильных объектов на полигоне, а значит, и в пакете программ, можно установить всю структуру и связи объектов (программных единиц), для чего необходимо грамотно выбирать разведывательные маршруты мобилей и их характеристики. В процессе движения мобилей собирается и такая информация, как выявление циклов в



а



б

Рис.1.9. Проход по маршруту [1,j] при радиусе обзора на один шаг вперед (а) и на два шага вперед (б)

процессах, их длины. Одновременно возникают и дополнительные проблемы: устранение избыточности данных, достоверность выводов, зависящая от стратегии обследования полигона и наблюдаемости объектов.

В результате основные задачи, относящиеся к семантическому полигону, могут быть сформулированы следующим образом: найти методы синтеза структуры СП с помощью мобильных объектов первого типа и идентификации среды СП посредством мобильных объектов второго типа.

Для этого определяется матрица разведывательных маршрутов

$$MMO_k[i,j],\ i=1, 2, \ldots, I_k^m,\ j=1, 2, \ldots, J_k^m,\ k=1, 2, \ldots, K,$$

позволяющая построить матрицы MSI, MSO за минимальное число рейдов мобилей, если известны формализованные спецификации стационарных объектов полигона. Будем искать матрицу разведывательных маршрутов в приведенном виде,

где κ – номер мобильного объекта второго типа;

m – номер маршрута κ-го мобильного объекта;

I_k^m – число маршрутов;

J_k^m – наибольшая длина маршрута k-го мобильного объекта;

K – максимальное число используемых мобилей в задаче ориентации на СП.

Не менее важной задачей на СП является достижение заданной цели (вершины) из указанной внутренней вершины семантического полигона. Мобильные объекты имеют средства связи со службой центрального управления полигона и могут получить команду, находясь в определенной вершине, переместиться в другую указанную вершину. Поэтому мобильный объект должен четко сориентироваться на полигоне и выработать стратегию своего перемещения для достижения заданной цели. В ходе решения задач ориентации на СП существенно определение оптимальной "видимости" мобилей для нахождения минимального числа рейдов при параллельном движении минимального числа мобилей, обеспечивающих вскрытие структуры СП. Программные средства САПР могут не иметь формализованных спецификаций используемых программных единиц. В этом случае ориентации в ППП предшествует задача чтения и распознавания исходных программных модулей с целью составления необходимых спецификаций, что значительно усложняет процесс идентификации программных сред САПР. В основе алгоритма идентификации структуры СП лежат запуск произвольного процесса на СП и последовательное доведение расчетов до получения достоверных результатов. Поиск минимального I_k^m осуществляется путем коррекции шага обзора r_i мобильных объектов (i – номера используемых разведывательных мобилей) на величину δr_i в зависимости от разницы обнаруженного числа стационарных объектов, объема входной и выходной информации СП. Принятие коллективных решений при организованном движении мобилей по СП происходит в процессе выработки решений на верхних уровнях управления СП с помощью системы диспетчеризации. Допускают-

ся взаимная связь между мобилями СП и согласование их действий. По результатам анализа структуры СП с учетом поставленных целей проводится прогноз решения необходимых задач. В частности, для функций y_i, задающих результат работы вычислительного процесса, находятся вероятности достижения цели P_i и соответствующие им интервалы времени T_j

$$P_1(y_1) \rightarrow T_{y_1},$$
$$P_2(y_2) \rightarrow T_{y_2},,$$
$$P_3(y_3) \rightarrow T_{y_3},,$$

после чего определяются

$$\max(P(y_1), P(y_2), \ldots)$$
$$\min(T_{y1}, T_{y2}, \ldots).$$

Критические участки маршрутов мобилей находятся с помощью операции пересечения АВФ процессов, т.е.

$$S_{кр}(MO_1, MO_2, \ldots) = y_1 \cap y_2 \cap y_3 \cap \ldots$$

Общая стратегия достижения заданных целей на полигоне определяется посредством операции объединения АВФ процессов, т.е.

$$Y = y_1 \cup y_2 \cup y_3 \cup \ldots$$

Система центрального управления или диспетчеризации осуществляет синхронизацию процессов в модели и обеспечивает реализацию выработанной стратегии по критерию наибольшей вероятности достижения мобилями своих целей, учитывая независимость поведения мобильных объектов. При этом может выделяться предпочтительный маршрут

$$P=\max[P_1, P_2, \ldots,].$$

Коллективное решение, принятое всеми участниками событий в СП под руководством системы верхнего уровня, порядок достижения мобилями своих целей или согласованные маршруты под управлением системы диспетчеризации закладываются в программу работы мобильных объектов, после чего производится запуск вычислительных процедур в СП.

2. ЛАБОРАТОРНЫЕ РАБОТЫ

ЛАБОРАТОРНАЯ РАБОТА №1

РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ НА ПРОЕКТИРОВАНИЕ

Цель работы – построение подсистемы синтеза технического задания, входящей в состав инструментальных средств поддержки автоматизированной разработки программного обеспечения.

Теоретическая часть

Система построения заданий на проектирование предназначена для подготовки и редактирования технических заданий на разработку программных средств, анализа и использования ранее созданных аналогов. Техническое задание вводится с помощью шаблона. Составление ТЗ происходит путем заполнения ключевых полей, которые однозначно идентифицируют каждый разрабатываемый проект. На данной стадии происходят изучение, анализ и уточнение исходных данных, оговариваются требования к будущей разработке. Этап завершается составлением спецификаций, согласованием их с заказчиком и документированием технического задания на проект.

В качестве примера системы автоматизированного проектирования рассмотрим комплекс САПР ЛИДС, предназначенный для решения прикладных задач из области САУ, а именно, линейных интервальных динамических систем (ЛИДС).

Разработка прикладных систем в САПР ЛИДС начинается с составления технического задания на проект. ТЗ заполняется с помощью шаблона. На основании заполненного шаблона модуль формирования технического задания создает спецификацию проекта, используя заданные ключевые поля, которые однозначно отличают один проект от другого (рис. 2.1). При выделении очередного поля его название появляется в нижней части шаблона, после чего ему можно присвоить значения, набрав их в строке ввода. При составлении технического задания в распоряжении разработчика находится БД с перечнем готовых проектов и их спецификациями.

Порядок выполнения работы

1. Изучить теоретическую часть лабораторной работы.
2. Разработать шаблон технического задания и заполнить его поля.
3. Сформировать техническое задание на проект и оформить отчет по выполненной работе.

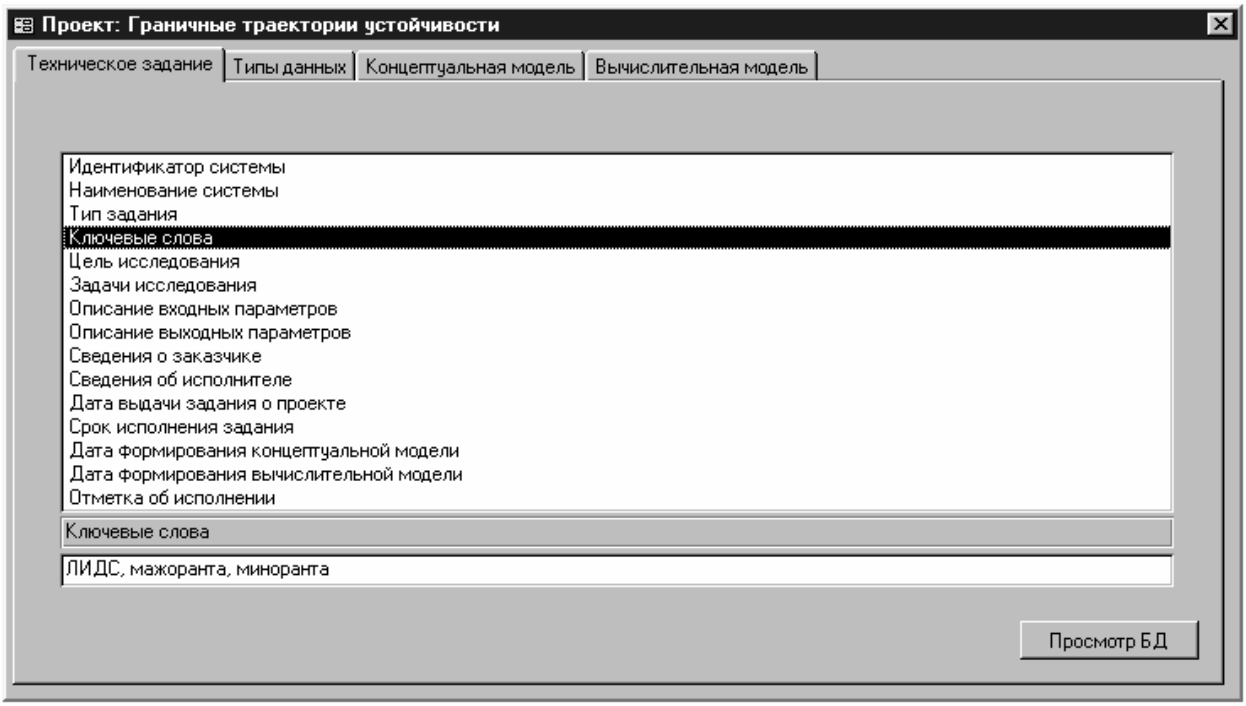


Рис. 2.1. Шаблон ТЗ

ЛАБОРАТОРНАЯ РАБОТА №2

РАЗРАБОТКА БАЗЫ ДАННЫХ ПРОГРАММНЫХ МОДУЛЕЙ

Цель работы – создание базы данных программных модулей для решения поставленных проектных задач.

Теоретическая часть

Одним из режимов работы, которые поддерживают комплексы автоматизированной разработки прикладных программных систем, является ведение базы данных программных модулей и проектов. Этот режим предусматривает доступ к запрашиваемым данным, их корректировку, удаление и создание. БД САПР ЛИДС выполнена с использованием языковых средств Visual Basic, для хранения данных используется формат БД Microsoft Jet. Это обусловлено нижеперечисленными особенностями. Указанный формат поддерживает средства, позволяющие быстро строить базы данных, а затем удобно поддерживать их надежную работу. В период между запросами файлы логически закрываются и становятся неуязвимыми. БД Jet имеет встроенные средства для работы с данными в сетевом режиме. Схема БД САПР ЛИДС приведена на рис. 2.2.

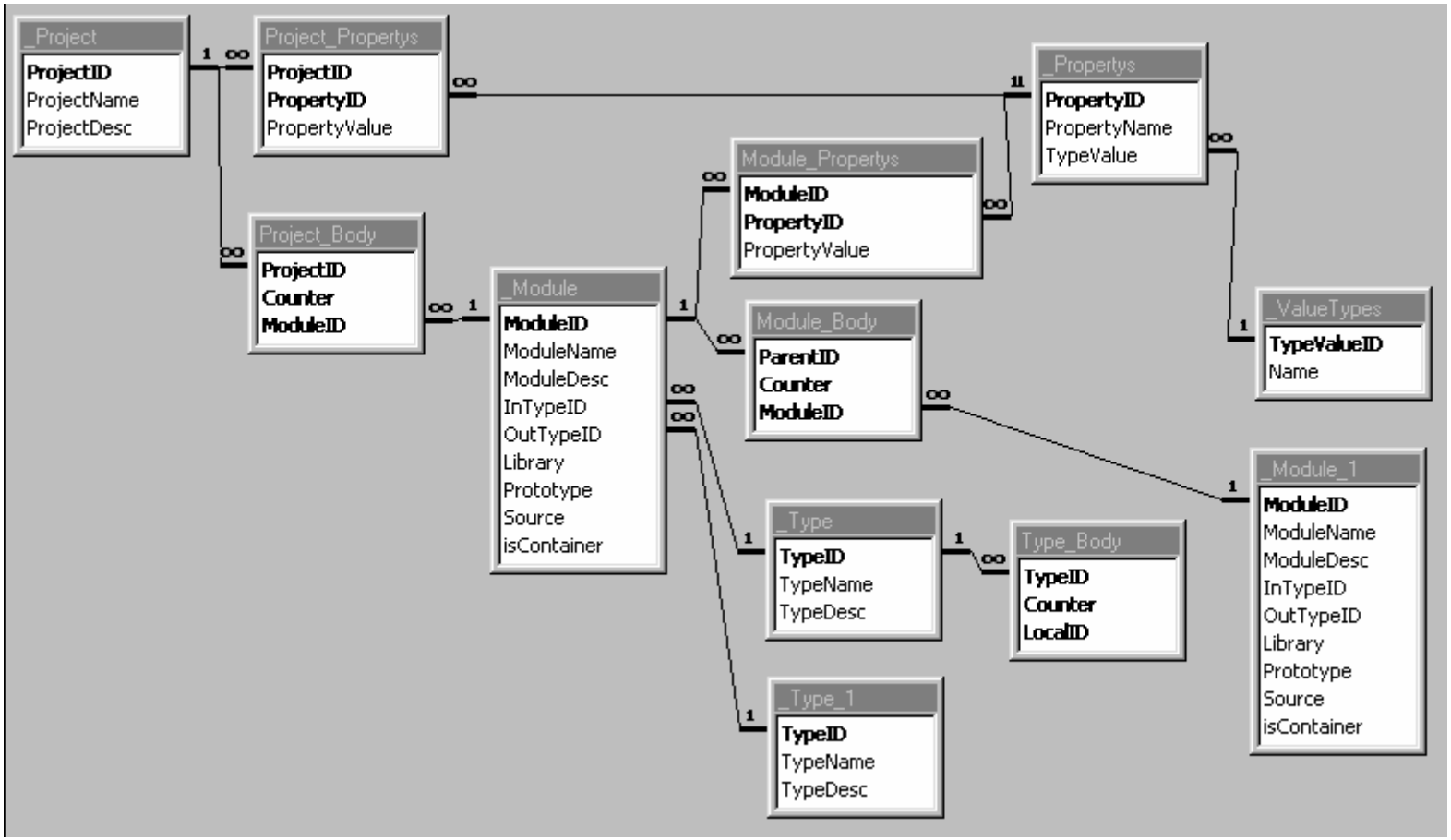


Рис. 2.2. Структура БД САПР ЛИДС

Вся информация, имеющаяся в БД, хранится в соответствующих именованных таблицах. Для каждой пары взаимодействующих таблиц определен тип связи: "один ко многим" или "многие к одному". Все проекты, создаваемые в комплексе, хранятся в таблице Project. Записи таблицы имеют по три поля. В поле ProjectID находится уникальный идентификатор проекта, в поле ProjectName – его название, а в поле ProjectDesc – описание проекта. Таблица проектов связывается с таблицей свойств _Propertys через таблицу Project_Propertys посредством ее уникального ключа ProjectID (все ключи в схеме БД выделены жирным шрифтом). Связь проекта с ПМ осуществляется через таблицу Project_Body. Ее ключ выбирает по идентификатору проекта все входящие в него модули по их порядковым номерам. Counter – это номер ПМ, а ModuleID – его идентификатор. Параметры ПМ хранятся в таблице _Module, где ModuleName – имя модуля, ModuleDesc – описание модуля, InTypeID – входные параметры, OutTypeId – выходные параметры, Library – статистика модуля, Prototype – прототипы фунций для модулей, написанных на языке С, Source – код, реализующий модуль на языке программирования, isContainer – признак модуля-контейнера. Таблица модулей связывается с таблицей свойств _Propertys через таблицу Module_Propertys посредством ее уникального ключа ModuleID. По полю InTypeID таблица модулей связывается с таблицей _Type, в которой хранятся имена типов и их описания для входных параметров модуля. Если тип параметра является сложным, то его расшифровка хранится в таблице Type_Body. По полю OutTypeID таблица модулей связывается с таблицей _Type1, которая является аналогом таблицы _Type. Если тип параметра является сложным, то его расшифровка выбирается из таблицы Type_Body. Таблица _Type1 присутствует на схеме для того, чтобы не нарушать корректность связей типа "многие к одному". Физически в БД содержится только одна таблица, имеющая структуру _Type.

Синтез объектов средствами САПР ЛИДС выполняется на основе программных модулей, которые хранятся в БД комплекса. Для описания ПМ и их занесения в БД заполняется соответствующий шаблон, его форма представлена на рис. 2.3. Режим "Состав" предусмотрен для ПМ типа "Контейнер" и предоставляет список модулей, которые входят в состав рассматриваемого ПМ. В режиме "Модуль" можно просмотреть реализацию ПМ на языке программирования. Режим "Статистика" дает информацию о том, сколько раз и в каких проектах использовался данный ПМ. Режим "Прочие" хранит второстепенную информацию о соответствующей программной единице. В описание ПМ необходимо внести информацию об их входных и выходных параметрах. Занесение сведений о типах данных проекта осуществляется в специальных окнах (рис. 2.4 и рис. 2.5). Модуль формирования типов данных предназначен для создания и корректировки типов данных проектируемого программного средства в интерактивном режиме. Этап описания данных является ключевым, поскольку все остальные подсистемы комплекса оперируют с данными, определяемыми на этом этапе. В системе могут использоваться данные следующих типов: предусмотренных в языке программирования, описанных разработчиком, сложных "контейнерных". На рис. 2.4 приведено описание сложного типа "массив",

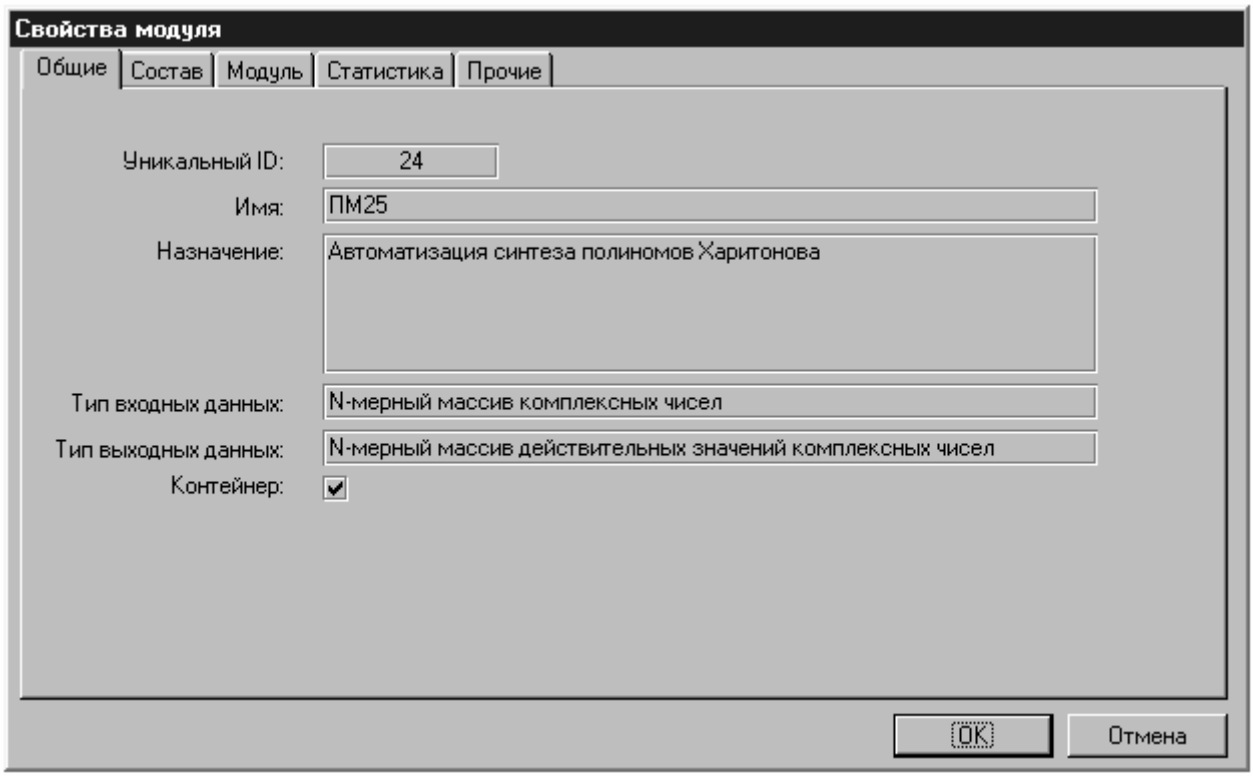


Рис.2.3. Шаблон заполнения ПМ

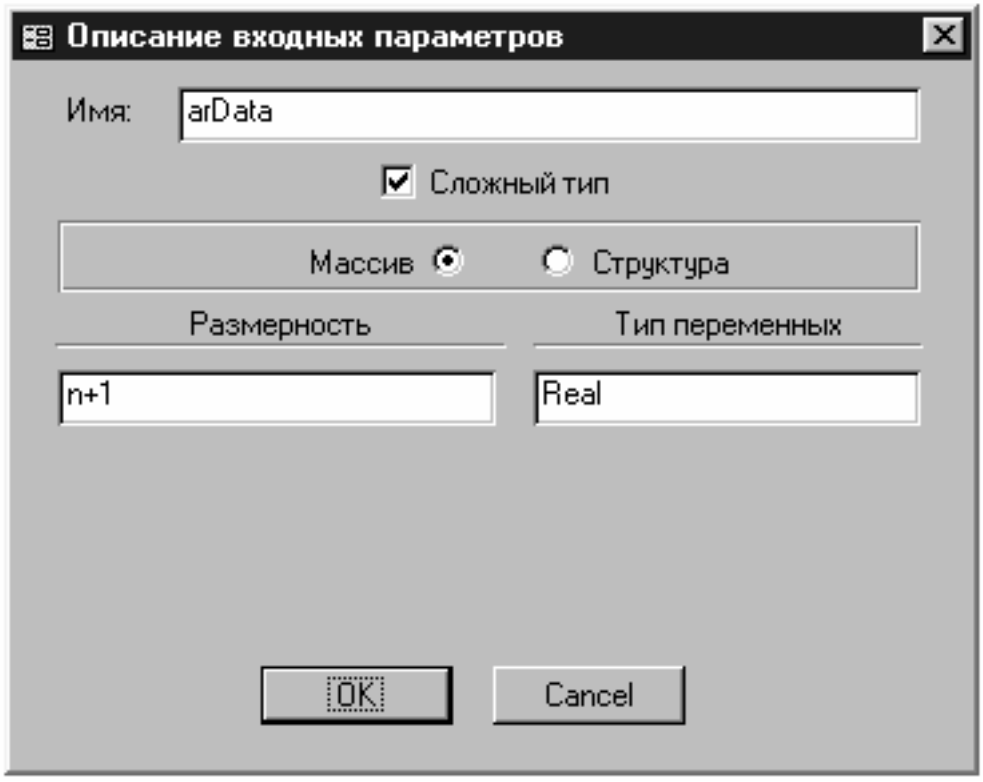


Рис. 2.4. Входные параметры ПМ

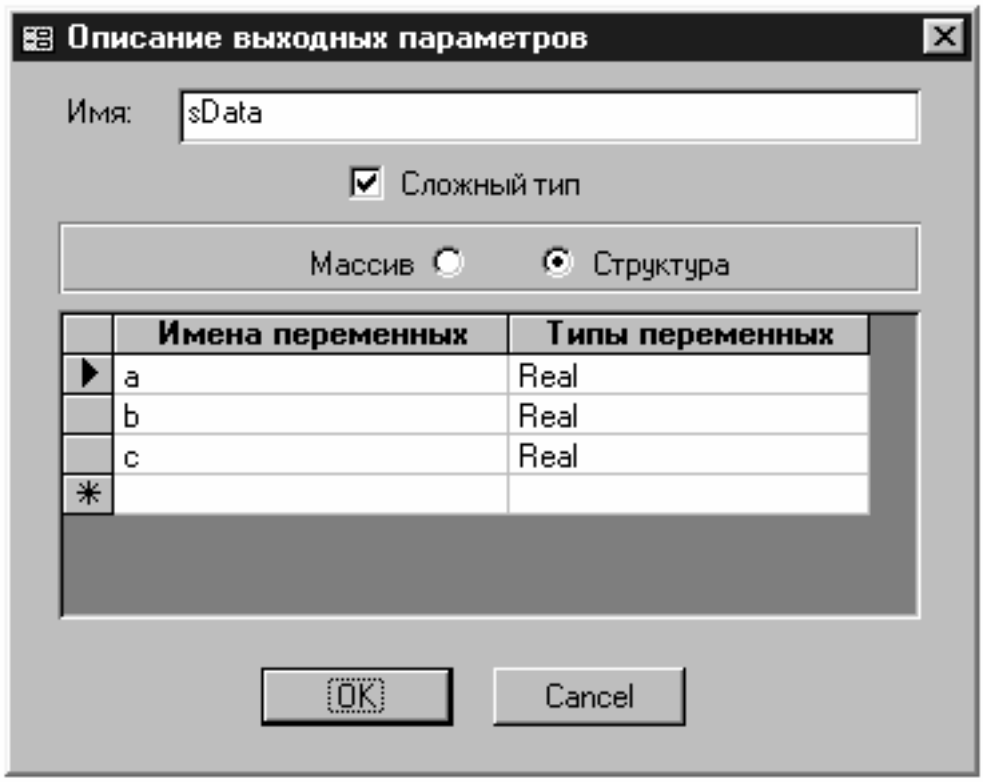


Рис. 2.5. Выходные параметры ПМ

имеющего размерность данных *(n+1)* и тип данных языка С – real. На рис. 2.5 представлено формирование сложного типа "структура", содержащего три переменные *a, b, c* типа real. Вся информация о программных модулях заносится в соответствующие таблицы БД проекта и в случае необходимости может подвергнуться корректировке без нарушения целостности данных. Это предусмотрено структурой БД и встроенными средствами БД Microsoft Jet.

Результатом рассмотренного этапа является сформированная БД ПМ с детальной проработкой всех типов данных, на которых будут создаваться подсистемы САПР ЛИДС, обеспечивающие дальнейшее проектирование прикладной программной системы.

Порядок выполнения работы

1. Изучить теоретическую часть лабораторной работы.
2. Разработать структуру БД программных модулей проекта.
3. Разработать шаблон описания программных модулей.
4. Продемонстрировать БД проекта и оформить отчет по лабораторной работе.

ЛАБОРАТОРНАЯ РАБОТА №3

ПОСТРОЕНИЕ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ ПРОЕКТА

Цель работы – создание подсистемы концептуального моделирования на основе базы данных программных модулей.

Теоретическая часть

Рассмотрим метод поиска проектных решений, а следовательно, и синтеза концептуальных моделей, ориентированный на жесткие связи по входным и выходным данным программных модулей.

В качестве примера выберем задачи, решаемые САПР ЛИДС, а именно, определение граничных траекторий устойчивости полей корневых годографов. Корневой годограф представляется как отображение заданной кривой или прямой плоскости свободного параметра алгебраического уравнения на плоскость комплексного переменного. Функция отображения определяется из заданного уравнения (в данном случае – это уравнение ЛИДС). Отображение описывается основными аналитическими уравнениями корневых годографов, которые строятся с помощью исходного уравнения вида

$$\Phi(p) + K\Psi(p) = A_0 p^n + A_1 p^{n-1} + ... + A_n = 0, \quad (3.1)$$

где $\Phi(p) = a_0 p^n + a_1 p^{n-1} + ... + a_n$, (3.2)

$$\Psi(p) = b_0 p^m + b_1 p^{m-1} + ... + b_m, \quad (3.3)$$

$$p = \delta + i\omega. \quad (3.4)$$

Из уравнения (3.1) определяется функция отображения комплексной плоскости K на плоскость p:

$$-K = \Phi(p)/\Psi(p). \quad (3.5)$$

Выделим два типа корневых годографов: с постоянной комплексной частью (КГПК) и с постоянной вещественной частью или постоянного усиления (КГПУ) варьируемого параметра.

Введем еще одно обозначение для K:

$$K = U + iV. \quad (3.6)$$

Определим КГПК как отображение прямой, параллельной действительной оси плоскости K, отстоящей от нее на расстояние V = const, причем U может принимать любые значения. Выполнив соответствующие подстановки из уравнений (3.4) и (3.6) в уравнение (3.5), а затем, выбрав из функции отображения комплексные слагаемые, получаем основное аналитическое уравнение КГПК:

$$F(\delta,\omega)P(\delta,\omega) - E(\delta,\omega)R(\delta,\omega) + V[P^2(\delta,\omega) + R^2(\delta,\omega)] = 0,$$

где

$$E(\delta,\omega) = \mathrm{Re}\Phi(p), \quad F(\delta,\omega) = \mathrm{Im}\Phi(p), \quad P(\delta,\omega) = \mathrm{Re}\Psi(p), \; R(\delta,\omega) = \mathrm{Im}\Psi(p).$$

Определим КГПУ как отображение прямой, параллельной мнимой оси iV плоскости K, отстоящей от нее на расстояние U = const, на комплексную плоскость p, причем V может принимать любые значения. Выполнив соответствующие подстановки из уравнений (3.4) и (3.6) в уравнение (3.5), а затем, выбрав из функции отображения действительные слагаемые, получаем основное аналитическое уравнение КГПУ

$$E(\delta,\omega)P(\delta,\omega) + F(\delta,\omega)R(\delta,\omega) + U[P^2(\delta,\omega) + R^2(\delta,\omega)] = 0.$$

Одним из наиболее важных вопросов, связанных с изучением особенностей САУ, является вопрос об их устойчивости. Методом корневых траекторий достаточно удобно исследовать характер и условия устойчивости систем автоматического управления. Для перехода от произвольной САУ к интервальной системе рассмотрим систему, заданную уравнением вида

$$f(p) = a_0 p^n + a_1 p^{n-1} + \ldots + a_n, \; p = \delta + i\omega \qquad (3.7)$$

с коэффициентами a_i, варьируемыми в интервалах $\underline{a_i} \le a_i \le \overline{a_i}, \; i = \overline{0,n}, \; \underline{a_0} \ne 0.$

Необходимые и достаточные условия устойчивости интервальных полиномов с действительными коэффициентами были получены В.Л.Харитоновым. Они заключаются в том, что у четырех полиномов вида (3.7) с определенными наборами коэффициентов все корни характеристического уравнения должны иметь отрицательные вещественные части.

Использование единичного годографа не позволяет оценить зависимость скорости изменения характеристик качества САУ от варьируемых элементов, хотя задачи такого типа нередко встречаются на практике. Эти задачи можно решать, используя обобщенный корневой годограф, которым называют совокупность траекторий характеристического уравнения системы при заданной области изменения свободного параметра. Обобщенный годограф с непрерывной областью изменения свободного параметра принято называть полем корневых траекторий. Будем рассматривать КГПК и КГПУ при условии, что $\Psi(p)$=1. Тогда уравнения полей годографов указанных типов принимают следующий вид

КГПК: $\mathrm{Im}\Phi(p) + V_i = 0 \Leftrightarrow F(\delta,\omega) + V_i = 0, \; i \in [1,\infty[,$

КГПУ: $\mathrm{Re}\Phi(p) + U_j = 0 \Leftrightarrow E(\delta,\omega) + U_j = 0, \; j \in [1,\infty[.$

Объединив понятия полей и их граничных траекторий, можно решать проблемы скорости изменения характеристик качества САУ, не выходя за границы ее

устойчивости. Для построения граничных траекторий полей введем определения мажорирующей (мажоранта) и минорирующей (миноранта) функций, представляющих граничные траектории полей ЛИДС.

Определение 5. Функцией, мажорирующей (минорирующей) в точке δ, назовем корневой годограф определенного типа, построенный на основе полиномов Харитонова, который при заданном значении δ принимает максимальное (минимальное) значение ω среди множества граничных корневых годографов.

На рис. 2.6. изображена графовая модель, соответствующая алгоритму определения граничных траекторий устойчивости полей в комплексе САПР ЛИДС. Данная сеть отражает концептуальную модель и имеет восемь уровней, которые обозначены цифрами рядом с вершинами сети. Внутри вершин помечены номера реализующих их ПМ. Синтез КМ выполняется от первого до восьмого уровней. В сети развиваются два процесса: (1,2) → 3 → А1 → (Ш1, 5) → 4 → Ш3 → 8 → 10 и (1,2) → 3 → А1 → (Ш2, 6) → 7 → Ш4 → 9 → 10. Цифры, заключенные в скобки, определяют ПМ, выполняемые на одном уровне маршрута. В маршруты входят ПМ трех типов: функциональные, обозначенные круглыми вершинами, анализаторы (А1) и шлюзы (Ш1). Функциональные вершины – это основные элементы управляющей сети, они решают поставленную задачу и при изображении имеют по одному входу и выходу. Анализаторы выполняют роль интерфейсных модулей с одним входом и множеством выходов, они группируют полученные данные по принадлежности к маршрутам и передают выделенные структуры соответствующим маршрутам. Шлюзы имеют множество входов и один выход. Они выполняют действия, обратные анализаторам: из полученных данных создают одну структуру и передают ее в качестве входного параметра нужному маршруту сети. Связи по входным/выходным параметрам использованных примитивов следующие:

1) входные данные – корни или коэффициенты ЛИДС, выходные данные – $E(\sigma,\omega)$, $F(\sigma,\omega)$, $P(\sigma,\omega)$, $R(\sigma,\omega)$;

2) входные данные – корни или коэффициенты ЛИДС, выходные данные – четыре полинома Харитонова;

3) входные данные – выходные данные ПМ1, выходные данные – корневые годографы заданных типов;

4) входные данные – результаты ПМ2 и ПМ3, выходные данные – граничные траектории полей типа КГПК;

5) входные данные – выходные данные ПМ3, выходные данные – поля корневых годографов типа КГПК;

6) входные данные – выходные данные ПМ3, выходные данные – поля корневых годографов типа КГПУ;

7) входные данные – результаты ПМ2 и ПМ3, выходные данные – граничные траектории полей типа КГПУ;

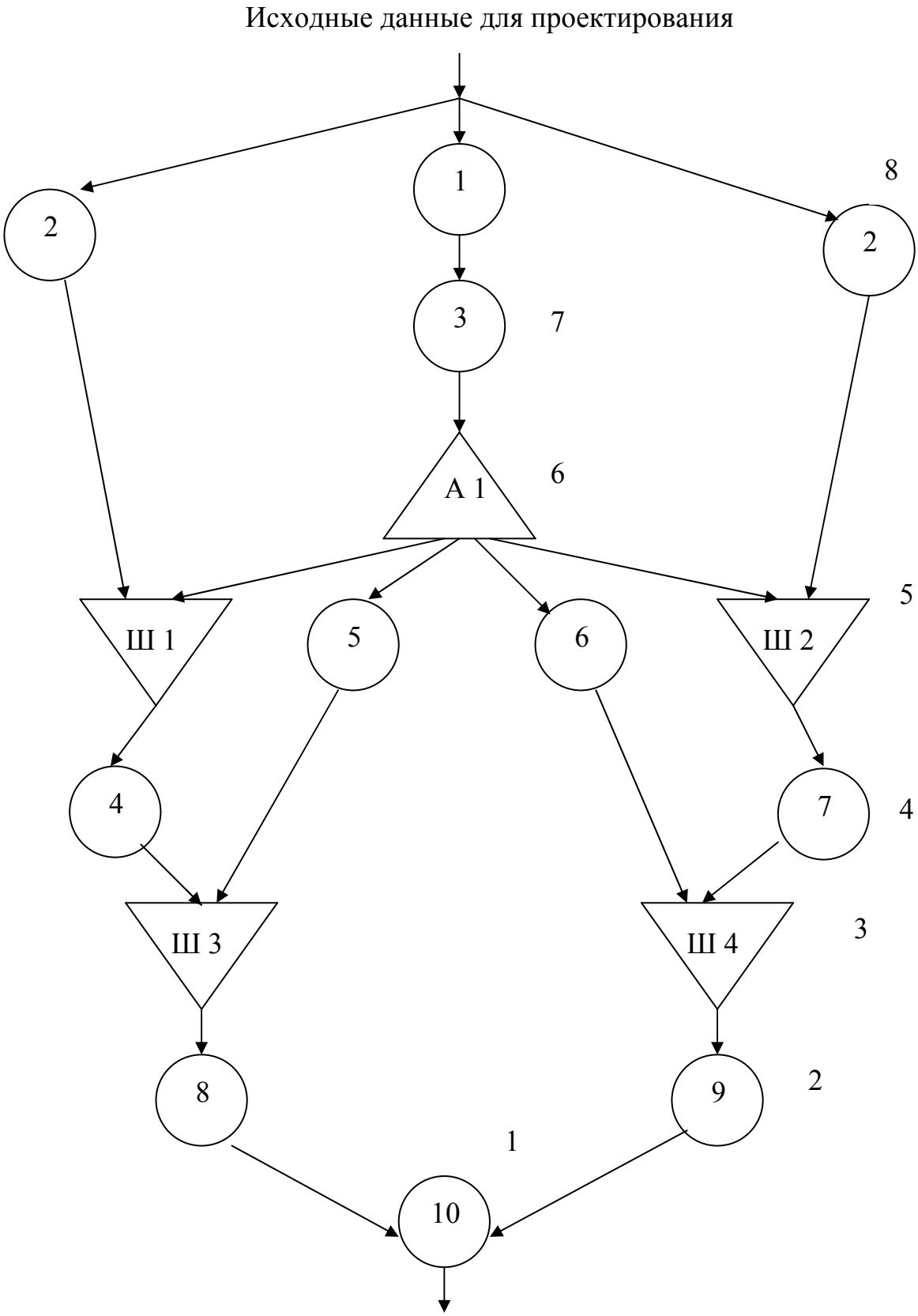


Рис. 2.6. Управляющая сеть концептуального моделирования

8) входные данные – результаты ПМ4 и ПМ5, выходные данные – изображения полей типа КГПК с соответствующими им граничными траекториями;

9) входные данные – результаты ПМ6 и ПМ7, выходные данные – изображения полей типа КГПУ с соответствующими им граничными траекториями;

10) на вход данного модуля результаты работы ПМ8 и ПМ9 могут подаваться вместе или по отдельности, в зависимости от этого он строит требуемые мажорирующие и минорирующие функции.

Жесткая связь по данным всех ПМ, использованных в примере, отражает полную формализацию построения концептуальной модели, а в результате обеспечивает автоматический синтез вычислительной модели.

Концептуальное моделирование в САПР ЛИДС является этапом, на котором выполняется построение управляющей графовой сети разрабатываемой системы. Момент начала синтеза КМ отражен на рис. 2.7.

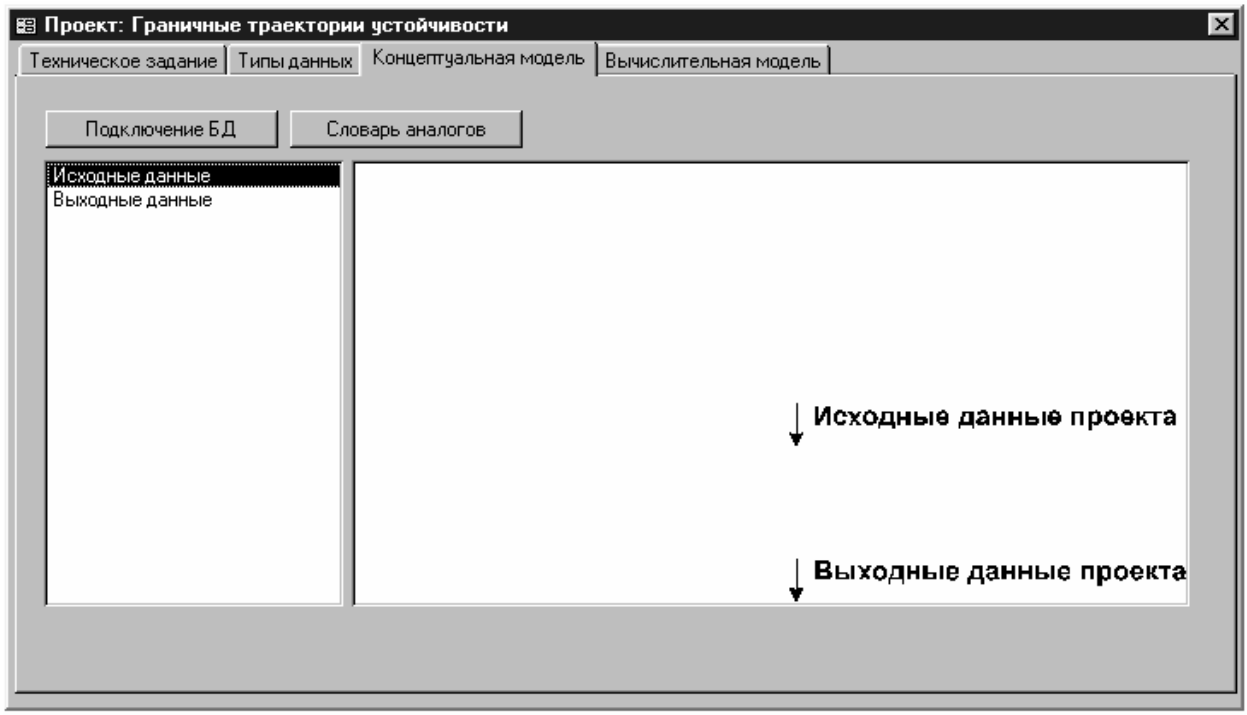


Рис. 2.7. Шаблон синтеза КМ

Первоначально вся графовая модель рассматривается как один ПМ, входными данными которого считаются исходные данные проекта, а выходными – выходные данные проекта. Структура связей по данным проекта отражается в процессе синтеза КМ в левом окне формы. В правом окне показано представление модели на текущем уровне абстракции, дальнейшее построение КМ выполняется в этом окне. Синтез У-сети развивается снизу вверх, постепенно соединяя выходные данные проекта с его исходными данными. Входные и выходные данные модулей, добавленных в процессе построения в модель, отра-

жаются в левом окне формы. Если на промежуточном этапе синтеза не будет найден ни один подходящий ПМ, следует перейти к этапу декомпозиции. Особенность декомпозиции заключается в том, что она выполняется в рамках КМ на одном из недостроенных маршрутов У-сети. В результате удается быстрее достигнуть того уровня абстракции, который можно реализовать при помощи ПМ. Подключение БД дает возможность получить полную информацию о запрашиваемом входном или выходном параметре и его принадлежности ПМ. Словарь аналогов предусмотрен на случай различного описания разработчиками программных модулей, выполняющих одинаковые функции. Использование словаря аналогов позволяет задействовать в проекте ПМ, описанные при помощи терминологии, незначительно отличающейся от принятой в данной разработке. На рис. 2.8 показано развитие КМ к моменту построения двух уровней синтеза. По рисунку видно, что в качестве ПМ первого уровня выбран модуль с условным номером 10. Следовательно, его выходные данные совпадают с тем, что должно быть на выходе проекта. В результате пространство для построения КМ сузилось, и теперь необходимо искать модули, чьи выходные данные могут являться входной информацией для ПМ10. Заданному условию удовлетворяют модули с номерами 8 и 9, они становятся объектами синтеза второго уровня. Описанный процесс продолжается до тех пор, пока не будут реализованы ПМ, принимающие на вход исходные данные проекта. Левое окно отражает передачу управления по данным с указанием номеров модулей и их принадлежность к соответствующему уровню синтеза.

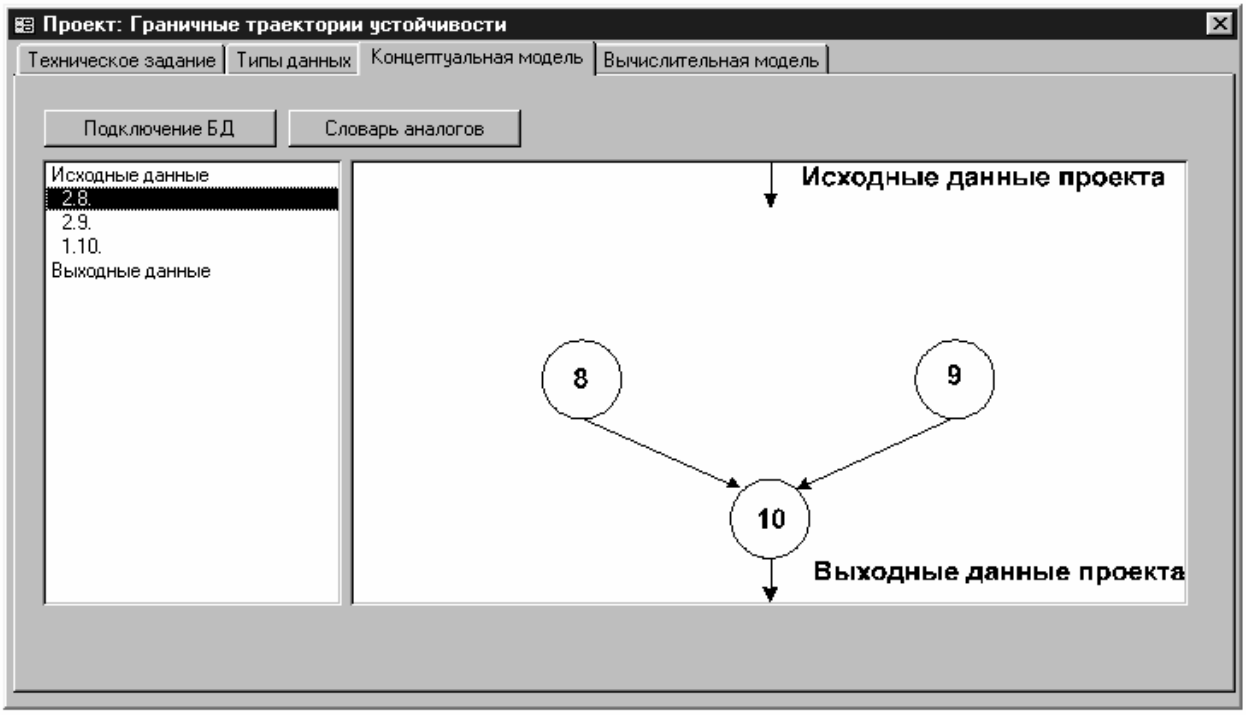


Рис. 2.8. Синтез КМ

Порядок выполнения работы

1. Изучить теоретическую часть лабораторной работы.
2. Разработать подсистему синтеза концептуальной модели.
3. Построить концептуальную модель решаемой задачи и оформить отчет по лабораторной работе.

ЛАБОРАТОРНАЯ РАБОТА №4

ПОСТРОЕНИЕ ВЫЧИСЛИТЕЛЬНОЙ МОДЕЛИ ПРОЕКТА

Цель работы – создание подсистемы вычислительного моделирования на основе базы данных программных модулей и концептуальной модели.

Теоретическая часть

Синтез вычислительной модели будем выполнять на основе концептуальной модели, построенной в лабораторной работе №3. У-сеть, представленная на рис. 2.6, содержит концептуальные модели решения двух задач, отличающихся типами корневых годографов. В другом случае КМ может содержать альтернативные решения одной задачи. Выбрав на сети цепочку, включающую только модули, необходимые для конкретного решения определенной задачи, получим соответствующую вычислительную модель. На рис. 2.9 показана ВМ, относящаяся непосредственно к КГПК.

На рис. 2.10 отражен этап решения задачи в среде автоматизированного проектирования, когда вся У-сеть построена, и разработчик может переходить к синтезу вычислительной модели. Режим синтеза ВМ позволяет просмотреть всю КМ и убедиться в её корректности. Синтез ВМ выполняется в порядке, обратном построению КМ: от исходных данных проекта к его результату. После нажатия кнопки "Синтез" система в диалоге с пользователем приступает к анализу ПМ, входящих в КМ, для построения ВМ. Фактически для этого необходимо “вырезать” из КМ нужное подмножество модулей. Затем на основе вычислительной модели автоматически синтезируется результирующий файл на заданном языке программирования.

Таким образом, предложенная система поддержки технологии автоматизированного проектирования отличается использованием адаптивного подхода разработки прикладных систем и визуализацией всех этапов проектирования. Адаптация работы САПР ЛИДС достигается за счет применения универсальной структуры БД, использования независимых ПМ, а поддержка основных этапов технологического процесса при создании сложных программных систем значительно повышает эффективность работы проектировщика.

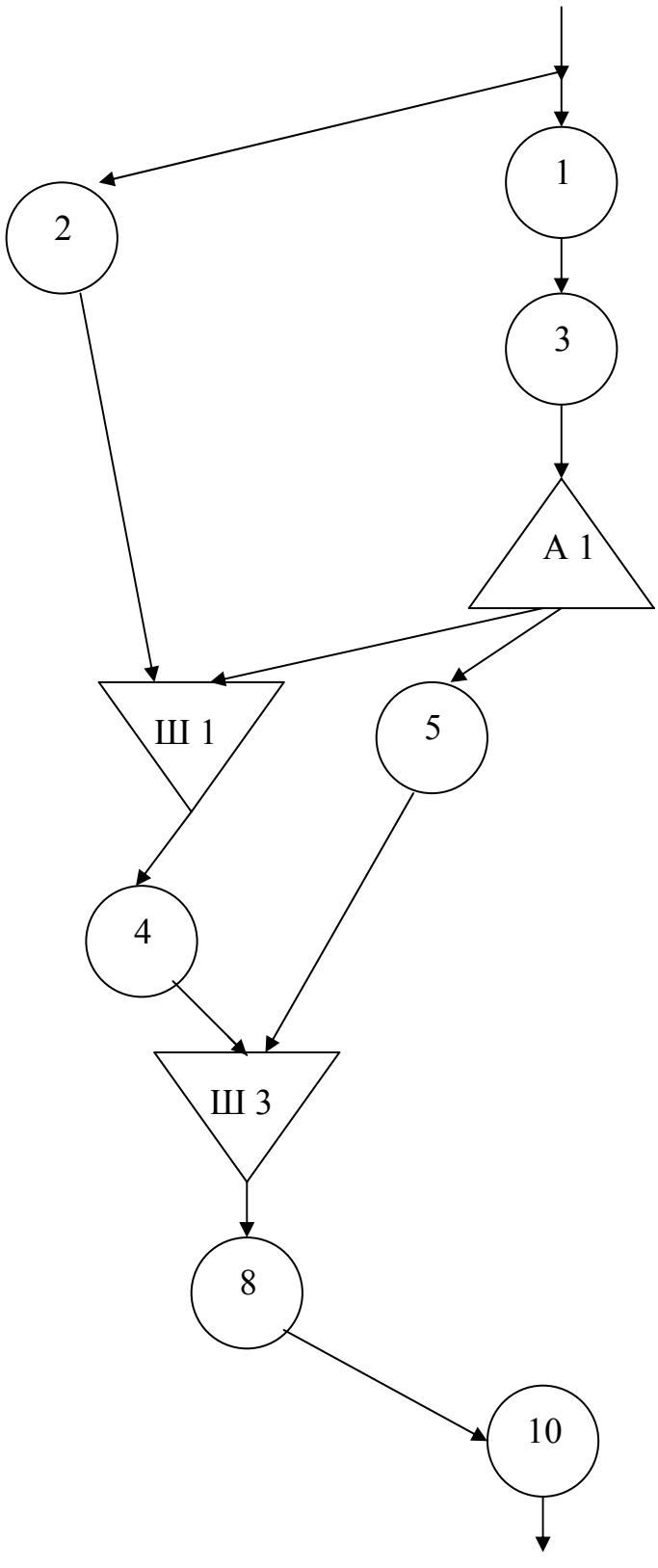


Рис. 2.9. Вычислительная модель для построения граничных траекторий полей КГПК

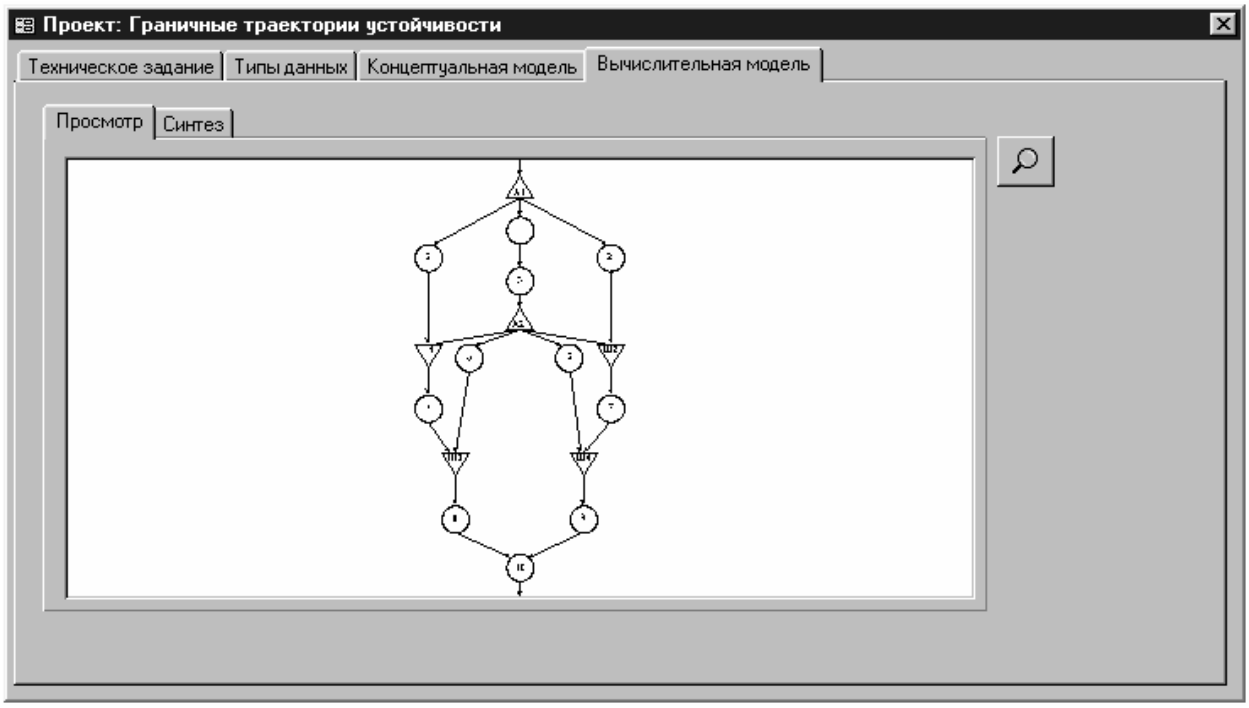


Рис. 2.10. Шаблон синтеза ВМ

Порядок выполнения работы

1. Изучить теоретическую часть лабораторной работы.
2. Разработать подсистему синтеза вычислительной модели.
3. Построить вычислительную модель решаемой задачи и оформить отчет по лабораторной работе.

ЛИТЕРАТУРА

1. Зиглер К. Методы проектирования программных систем. – М.: Мир, 1985.

2. Новоженов Ю.В. Объектно-ориентированные технологии разработки сложных программных систем. – М.: Финансы и статистика, 1996.

3. Римский Г.В. Структура и функционирование системы автоматизации модульного программирования // Программирование. – 1987. – №5. – С. 36–44.

4. Шатохин И.В. Синтез программ модульной структуры на алгоритмических языках высокого уровня. – Мн., 1989. – 22 с. (Препринт / Ин-т техн. кибернетики НАН Беларуси; №26).

5. ФИХАР – модульная система программ для реакторных расчетов / А.И. Башмачников, Б.А. Загацкий, М.Н. Зизин и др.: Тр. Ш семинара по комплексам программ математической физики. – Новосибирск, 1973.

6. Тыугу Э.Х. Решение задач на вычислительных моделях // Вычислительная математика и математическая физика. – 1970. – Т.10. – №3. – С. 716–733.

7. Римский Г.В. Теория систем автоматизированного проектирования. – Мн.: Навука і тэхніка, 1994.

8. Римский Г.В. Организация и функционирование инструментальных средств комплекса автоматизированного проектирования систем управления // Матер. по матем. обеспеч. ЭВМ. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 1991. – 80 с.

9. Римский Г.В., Бочкарёва Л.В., Мазуренко Е.Г. Инфраструктура синтеза вычислительных моделей комплекса автоматизированного проектирования // Матер. по матем. обеспеч. ЭВМ. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 1996. – 178 с.

10. Римский Г.В., Репеко Н.П. Инструментальная система поддержки процессов разработки технических заданий на проектирование в комплексе автоматизированного проектирования систем управления // Матер. по матем. обеспеч. ЭВМ. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 1991.–204 с.

11. Римский Г.В., Репеко Н.П. Инфраструктура декомпозиции проектов комплекса автоматизированного проектирования систем управления // Матер. по матем. обеспеч. ЭВМ. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 1992. – 108 с.

12. Римский Г.В., Бочкарёва Л.В., Таборовец О.В. Автоматно-временное описание процессов на семантическом полигоне. – Мн., 1997. – 28 с. (Препринт / Ин-т техн. кибернетики НАН Беларуси; №3).

13. Бочкарёва Л.В. Автоматизация исследований процессов на семантическом полигоне: Сб. науч. тр. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 1998. – Вып. 1. – С.179–186.

14. Автоматно-временной анализ процессов на семантическом полигоне / Г.В. Римский, Л.В. Бочкарёва, О.В. Таборовец, В.А. Мазуренко, А.Г. Римский // Весці НАН Беларусі. Сер фіз.-тэхн. навук. – 1999. – № 1. – С.76–83.

15. Римский Г.В. Основы общей теории корневых траекторий систем автоматического управления. – Мн: Наука и техника, 1972.

Св. план 2002, поз. 51

Учебное издание

Бочкарёва Лия Валентиновна

Учебно-методическое пособие
по курсу
«Технология разработки и системы автоматизированного проектирования программного обеспечения»
для студентов специальности
«Программное обеспечение информационных технологий»

Редактор Н.А. Бебель
Корректор Е.Н. Батурчик

Подписано в печать		Формат 60 x 84 1/16
Бумага	Печать	Усл. печ. л.
Уч.-изд. л.	Тираж 100 экз.	Заказ

Издатель и полиграфическое исполнение:
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Лицензия ЛП №156 от 05.02.2001.
Лицензия ЛВ №509 от 03.08.2001.
220013, Минск, П.Бровки, 6