МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

СИСТЕМЫ АНАЛИТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Методические рекомендации к курсовому проектированию для студентов специальности 1-40 80 02 «Системный анализ, управление и обработка информации» очной и заочной форм обучения

Могилев 2020

УДК 004.43
ББК 32.973-018
С40

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Автоматизированные системы управления» «09» июня 2020 г., протокол № 11

Составители: д-р техн. наук, доц. А. И. Якимов;
канд. техн. наук Е. А. Якимов

Рецензент канд. техн. наук, доц. И. В. Лесковец

Даны методические рекомендации к выполнению курсовой работы по дисциплине «Системы аналитического программирования», а также изложены требования к содержанию и оформлению курсовой работы и список литературы для подготовки.

Учебно-методическое издание

СИСТЕМЫ АНАЛИТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Ответственный за выпуск	А. И. Якимов
Корректор	А. А. Подошевко
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 16 экз. Заказ № .

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, Могилев.

Содержание

Введение

Целью изучения дисциплины «Системы аналитического программирования» является приобретение студентами теоретических знаний и практических навыков в области основ теории обучения машин, современных методов восстановления зависимостей по эмпирическим данным, включая дискриминантный, кластерный и регрессионный анализ, овладение навыками практического решения задач интеллектуального анализа данных.

Студент, изучивший дисциплину, будет **знать**:

– фундаментальные понятия, современные подходы, методы и проблемы машинного обучения и интеллектуального анализа данных.

Студент, изучивший дисциплину, будет **уметь**:

– понять и формализовать поставленную задачу анализа данных;

– использовать современные методы машинного обучения для практического решения задач анализа данных;

– при необходимости, продиктованной особенностями поставленной задачи, создавать новые методы машинного обучения;

– проводить численные эксперименты на модельных и реальных данных и интерпретировать их результаты;

– представлять результаты исследований в устной и письменной форме.

Студент, изучивший дисциплину, будет **владеть**:

– навыками освоения большого объема информации и решения сложных теоретических и практических задач анализа данных;

– навыками самостоятельной работы и освоения новых дисциплин;

– культурой постановки, анализа и решения математических и прикладных задач, требующих для своего решения использования математических подходов и методов;

– предметным языком машинного обучения и интеллектуального анализа данных, навыками описания решения задач и представления полученных результатов.

1 Цель и задачи курсовой работы

Цель курсовой работы – самостоятельное изучение дополнительного материала в области основ теории обучения машин, современных методов восстановления зависимостей по эмпирическим данным, включая дискриминантный, кластерный и регрессионный анализ, овладение навыками практического решения задач интеллектуального анализа данных и приобретение навыков выполнения самостоятельной исследовательской работы.

Задачи курсовой работы:

– приобретение навыков интеллектуального анализа данных в различных производственных системах;

– приобретение навыков практического применения современных методов восстановления зависимостей по эмпирическим данным, включая дискриминантный, кластерный и регрессионный анализ, овладение навыками практического решения задач интеллектуального анализа данных;

– совершенствование навыков перехода от метода решения формализованной задачи к конкретному алгоритму, на основе которого будет сформирована и реализована система аналитического программирования.

2 Организация курсового проектирования

Выполнение курсовой работы основывается на применении методологии системного анализа и использовании современных компьютерных технологий для решения, в основном, хорошо структурированных и неструктурированных задач интеллектуального анализа данных.

Курсовая работа выполняется в соответствии с выданным заданием, подписанным руководителем и утвержденным заведующим кафедрой. В задании указываются тема курсовой работы, исходные данные и календарный график выполнения работы.

После каждого этапа в соответствии с контрольными датами все материалы курсовой работы должны быть предоставлены руководителю для контроля.

Выполненная курсовая работа подписывается студентом и сдается на проверку руководителю, по результатам которой решается вопрос о допуске к защите. При наличии замечаний курсовая работа возвращается студенту на доработку, после которой повторно представляется на проверку.

Для защиты работы организуется комиссия в составе не менее двух преподавателей, один из которых – руководитель работы. На защите студент делает доклад продолжительностью до 5…10 мин, в котором кратко излагает суть выполненной работы, а затем отвечает на вопросы членов комиссии. По результатам доклада, содержанию ответов на вопросы, демонстрации программных средств и качеству оформления работы выставляется оценка.

В докладе необходимо:

– сформулировать цель работы;

– перечислить задачи, решаемые при выполнении курсовой работы;

– показать способность ориентироваться в основных принципах и понятиях интеллектуального анализа данных;

– изложить принятые решения, касающиеся поставленной проблемы анализа данных и программной реализации задачи;

– подвести итоги выполнения курсовой работы.

3 Содержание курсовой работы

Курсовая работа содержит пояснительную записку, структура которой учитывает основные принципы интеллектуального анализа данных и приведена в таблице 1.

Таблица 1 – Структура пояснительной записки

Наименование раздела	Рекомендуемый объем, с.
Титульный лист	1
Задание на курсовую работу	1
Содержание	1
Введение	1
1 Постановка задачи	
1.1 Определение целей функционирования системы аналитического программирования	0,5…1
1.2 Определение входных и выходных параметров системы	0,5
1.3 Определение ограничений системы	0,5
2 Выбор и обоснование метода решения задачи	
2.1 Анализ существующих методов решения задачи	1…2
2.2 Выбор метода (комплекса методов) решения задачи	2…4
2.3 Формализация решения поставленной задачи выбранным методом (комплексом методов)	2…5
3 Разработка программного модуля для автоматизации решения поставленной задачи выбранным методом (комплексом методов)	
3.1 Проектирование схем алгоритмов решения задачи	4…8
3.2 Проектирование диаграммы классов	1
3.3 Реализация программного модуля	1…2
3.4 Тестирование программного модуля	5…8
4 Реализация выбора и принятие решения	1…2
Заключение	1
Список литературы	1
Приложение	

Пояснительная записка должна содержать последовательное изложение всех этапов выполнения курсовой работы с использованием существующей терминологии и стандартов. Исходные положения и принимаемые решения должны быть обоснованы и логически взаимосвязаны.

4 Методические рекомендации

Методические рекомендации раскрывают содержание разделов пояснительной записки курсовой работы.

Введение.

Во введении должны быть отражены цель и назначение курсовой работы, определен круг задач, решаемых в курсовой работе. Необходимо дать общую характеристику интеллектуального анализа данных, используемого при решении поставленной задачи.

Введение должно содержать краткое обоснование актуальности выполненного исследования: где и как оно может быть применено, в чем состоит его практическая польза.

При необходимости введение может содержать дополнительные сведения, например, информацию о практическом применении полученных результатов, в том числе в учебном процессе, об опубликовании полученных результатов и т. п.

1 Постановка задачи.

1.1 Определение целей функционирования системы аналитического программирования.

Цель – это желаемый результат деятельности, достижимый в пределах некоторого интервала времени. Цель исследований следует сформулировать исходя из сущности решения, на получение которых ориентирована данная проблема.

После того как цель сформулирована, появляется возможность выбора связанных с ней критериев. Под критерием понимается правило, по которому проводится отбор тех или иных средств достижения цели или выбор наилучшего варианта среди определенного множества. Основой такого правила может быть однозначная численная характеристика объекта, представляющая собой скалярную функцию. Эта характеристика содержательно отображает цель поиска, в связи с чем ее называют целевой функцией. Она позволяет количественно выразить качество объекта и поэтому называется также функцией качества. Таким образом, в основе построения правила предпочтения лежит целевая функция.

Задача параметрической оптимизации системы заключается в поиске параметров, при которых целевая функция достигает экстремального значения. Параметры системы, доставляющие экстремум целевой функции, называются оптимальными.

Аргументами целевой функции являются управляемые параметры. В качестве управляемых параметров выступают внутренние параметры технического объекта, подлежащие оптимизации. Изменяя соответствующим образом эти параметры в процессе оптимизации, осуществляют поиск экстремума целевой функции. Следует отметить, что часть внутренних параметров объекта задается, и они не подлежат оптимизации (например, параметры стандартизированных или унифицированных элементов, параметры, регламентированные техническим заданием на проектирование, и др.).

В подразделе необходимо описать цель функционирования системы и определить критерии выбора наилучшего варианта среди определенного множества или целевую функцию системы.

1.2 Определение входных и выходных параметров системы.

Параметр – это величина, характеризующая свойство или режим работы объекта. Под объектом здесь понимается как отдельный элемент системы, так и вся система в целом.

Все учитываемые параметры задачи нужно разделить на входные, управляемые и выходные.

Входным параметром называется неизвестная переменная величина, изменение которой приводит к изменению критерия оптимальности.

Управляемые параметры – это параметры элементов объекта, подлежащие оптимизации.

Выходными параметрами технической системы являются показатели качества и эффективности: производительность, рабочая скорость, грузоподъемность, габариты, масса, показатели надежности и др. Выходными параметрами экономической системы являются прибыль, доход, рентабельность и т. п.

При описании входных и выходных данных указываются пределы, в которых они могут изменяться, и значения, которые они не могут принимать.

1.3 Определение ограничений системы.

Различают прямые и функциональные ограничения.

Прямые ограничения накладываются на управляемые параметры: $x_i > x_{\text{н}i}$; $x_i < x_{\text{в}i}$; $i \in [1:n]$, где $x_{\text{н}i}$, $x_{\text{в}i}$ – нижнее и верхнее граничные значения управляемого параметра x_i; n – размерность пространства управляемых параметров.

Область в пространстве управляемых параметров, заданную прямыми ограничениями, называют допустимой областью или областью допустимых значений управляемых параметров. В качестве примера прямого ограничения можно назвать условие неотрицательности переменных (количества перевозимого груза) в транспортной задаче линейного программирования.

Функциональные ограничения подразделяются на два вида: ограничения-неравенства и ограничения-равенства. Функциональные ограничения устанавливают некоторые зависимости между управляемыми параметрами, нарушение которых недопустимо по условиям обеспечения работоспособности или регламентируемой эффективности функционирования технического объекта. Прямые ограничения можно рассматривать как частный случай функциональных ограничений.

Наличие ограничений приводит к задаче условной оптимизации, при которой находится условный экстремум целевой функции.

Наложенные ограничения приводят к тому, что поиск оптимального решения ограничивается некоторой областью в пространстве управляемых параметров, которая составляет лишь некоторую часть области определения целевой функции.

В подразделе при описании ограничений системы необходимо определить прямые и функциональные ограничения.

2 Выбор и обоснование метода решения задачи.

2.1 Анализ существующих методов решения задачи.

Возможность применения того или иного метода решения проблемы зависит от степени структурированности проблемы.

Согласно классификации, предложенной Г. Саймоном и А. Ньюэллом, все множество проблем системного анализа в зависимости от глубины их познания подразделяется на три следующих класса.

Хорошо структурированные проблемы – многовариантные по существу и количественно сформулированные проблемы, в которых основные зависимости выяснены настолько хорошо, что они могут быть выражены в числах или символах, а в результате решения получаются количественные оценки. Для решения хорошо структурированных проблем используется методология исследования операций. Исследование операций использует математический аппарат линейного, целочисленного, нелинейного и динамического программирования, теории оптимизации, сетевого планирования и управления, управления запасами, теории массового обслуживания.

Неструктурированные, или качественно выраженные проблемы содержат только описание важнейших признаков и характеристик, при этом количественные зависимости между ними неизвестны. В неструктурированных проблемах применяются методы принятия решений в условиях неопределенности (теория игр), а также на основе экспертных методов, эвристического программирования.

Слабоструктурированные проблемы, которые содержат как количественные, так и качественные малоизвестные и неопределенные зависимости. Для решения слабоструктурированных проблем используются статистические и вероятностные методы, векторная оптимизация, методы корреляционного, регрессионного и кластерного анализа, методы, использующие нечеткие множества, методы искусственного интеллекта (нейронные сети, генетические алгоритмы).

В подразделе необходимо определить класс, к которому принадлежит решаемая проблема, и кратко проанализировать сущность, достоинства и недостатки наиболее подходящих методов для решения поставленной задачи. Формирование списка анализируемых методов можно осуществить, например, в соответствии со следующими критериями:

– методы должны позволять решать задачи с ограничениями, как правило, легко вычислимыми;

– иметь малые вычислительные затраты;
– обеспечивать малое число обращений к целевой функции;
– позволять решать задачи больших размерностей;
– иметь приемлемую точность решения;
– иметь сравнительную легкость реализации.

2.2 Выбор метода (комплекса методов) решения задачи.

В данном подразделе приводится обоснование выбора метода (комплекса методов) решения задачи и краткое описание выбранного метода (комплекса методов).

При выполнении этого подраздела производится изучение и анализ литературных и иных источников по выбранному численному методу решения поставленной задачи. При работе над подразделом обязательно указываются используемые литературные источники в соответствии с требованиями ГОСТ 7.1–2003.

2.3 Формализация решения поставленной задачи выбранным методом (комплексом методов).

Формализация – это отображение результатов мышления на языке знаков, математических формул. Формализация предполагает построение математической модели решаемой задачи. Построение модели системы есть не что иное, как процесс уяснения общих свойств и закономерностей ее функционирования и развития, т. е. овладение методами построения моделей систем является необходимым условием успешного овладения методологией системного анализа.

По форме представления математических моделей различают инвариантную, алгоритмическую, аналитическую и графическую модели объекта проектирования.

В инвариантной форме математическая модель представляется системой уравнений (дифференциальных, алгебраических) вне связи с методом решения этих уравнений.

В алгоритмической форме соотношения модели связаны с выбранным численным методом решения и записаны в виде алгоритма – последовательности вычислений.

Аналитическая модель представляет собой явные зависимости искомых переменных от заданных величин (обычно зависимости выходных параметров объекта от внутренних и внешних параметров). Такие модели получают на основе физических законов либо в результате прямого интегрирования исходных дифференциальных уравнений, используя табличные интегралы. К ним относятся также регрессионные модели, получаемые на основе результатов эксперимента.

Графическая (схемная) модель представляется в виде графов, эквивалентных схем, динамических моделей, функциональных, кинематических и алгоритмических схем, диаграмм, циклограмм и т. п. Для использования графических моделей должно существовать правило однозначного соответствия условных изображений элементов графической и компонентов инвариантной математических моделей.

Среди алгоритмических моделей выделяют имитационные модели, предназначенные для имитации физических и информационных процессов, протекающих в объекте при функционировании его под воздействием различных факторов внешней среды.

По характеру отображаемых свойств объекта математические модели подразделяются на функциональные и структурные.

Структурные модели отображают только структуру объектов и используются при решении задач структурного синтеза. Параметрами структурных моделей являются признаки функциональных или конструктивных элементов, из которых состоит объект и по которым один вариант структуры объекта отличается от другого. Эти параметры называют морфологическими переменными. Структурные модели имеют форму таблиц, матриц и графов.

Функциональные модели описывают процессы функционирования объектов и имеют форму систем уравнений. Они учитывают структурные и функциональные свойства объекта и позволяют решать задачи как параметрического, так и структурного синтеза.

Данный подраздел должен содержать формализованное описание стратегии решения задачи выбранным методом.

Для задачи из задания на курсовую работу составляется математическая модель с указанием ее типа (см. приведенную выше классификацию моделей). Все математические зависимости должны быть подробно описаны и содержать ссылки на литературные источники.

3 Разработка программного модуля для автоматизации решения поставленной задачи выбранным методом (комплексом методов).

3.1 Проектирование схем алгоритмов решения задачи.

При проектировании алгоритма следует использовать следующие принципы:

– принцип нисходящего проектирования, в соответствии с которым производится декомпозиция исходной задачи на подзадачи;

– принцип модульности, согласно которому выделенные подзадачи реализуются как отдельные модули, что обеспечивает легкость составления алгоритмов и отладки программ, легкость сопровождения и модификации;

– принцип структурного программирования, в соответствии с которым в структуре модулей можно выделить типовые блоки.

Схема алгоритма метода должна четко и однозначно отображать всю последовательность дискретных операций на рассматриваемом этапе и быть оформлена согласно ГОСТ 19.701–90.

При разработке подраздела вначале оговариваются дополнительные переменные для промежуточных результатов расчета. Далее реализовывается вычислительный алгоритм выбранного метода (комплекса методов).

В подразделе также должна присутствовать блок-схема обобщенного алгоритма функционирования приложения.

3.2 Проектирование диаграммы классов.

Методология системного анализа служит концептуальной основой системно-ориентированной декомпозиции предметной области. Результатом системного анализа является построение некоторой модели системы. Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов отражает различные взаимосвязи между объектами и подсистемами, а также описывает их внутреннюю структуру и типы отношений. Диаграмма классов показывает отношения между классами, которые не изменяются во время выполнения, – это отношения наследования, включения, ассоциации.

Класс является абстрактным описанием или представлением свойств множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. Графически класс изображается в виде прямоугольника, который разделен горизонтальными линиями на секции. В этих секциях указывается имя класса, атрибуты (переменные) и операции (методы).

Имя класса должно быть уникальным в пределах пакета, который может содержать несколько диаграмм классов. Имя класса записывается по центру самой верхней секции полужирным шрифтом и должно начинаться с заглавной буквы. В качестве имен классов используются существительные, записанные без пробелов.

Атрибут класса служит для представления отдельного свойства или признака, который является общим для всех объектов данного класса. Атрибуты класса записываются во второй секции прямоугольника класса.

Общий формат записи отдельного атрибута класса следующий:

<квантор видимости> <имя атрибута> [кратность]:<тип атрибута> = <исходное значение> {строка-свойство}

Квантор видимости (visibility) может принимать одно из четырех возможных значений и, соответственно, отображается при помощи специальных символов.

Символ «+» обозначает атрибут с областью видимости типа общедоступный (public). Атрибут с этой областью видимости доступен или виден из любого другого класса пакета, в котором определена диаграмма.

Символ «#» обозначает атрибут с областью видимости типа защищенный (protected). Атрибут с этой областью видимости недоступен или невиден для всех классов, за исключением подклассов данного класса.

Символ «—» обозначает атрибут с областью видимости типа закрытый (private). Атрибут с этой областью видимости недоступен или невиден для всех классов без исключения.

И, наконец, символ «~» обозначает атрибут с областью видимости типа пакетный (package). Атрибут с этой областью видимости недоступен или невиден для всех классов за пределами пакета, в котором определен класс-владелец данного атрибута.

Вместо условных графических обозначений можно записывать соответствующее ключевое слово: public, protected, private или package.

Имя атрибута представляет собой строку текста, которая используется в качестве идентификатора соответствующего атрибута и поэтому должна быть уникальной в пределах данного класса. Имя атрибута является единственным обязательным элементом синтаксического обозначения атрибута, оно должно начинаться со строчной (малой) буквы и не должно содержать пробелов. Тип атрибута определяется типом данных, определенным разработчиком в зависимости от языка программирования, который предполагается использовать для реализации данной модели. Типу атрибута должно предшествовать двоеточие.

Исходное значение служит для задания некоторого начального значения для соответствующего атрибута в момент создания отдельного экземпляра класса.

При задании атрибутов могут быть использованы две дополнительные синтаксические конструкции – это подчеркивание строки атрибута и пояснительный текст в фигурных скобках.

Подчеркивание строки атрибута означает, что соответствующий атрибут является общим для всех объектов данного класса.

Строка-свойство служит для указания дополнительных свойств атрибута, которые могут характеризовать особенности изменения значений атрибута в ходе выполнения соответствующей программы. Фигурные скобки как раз и обозначают фиксированное значение соответствующего атрибута для класса в целом, которое должны принимать все вновь создаваемые экземпляры класса без исключения.

Операция – это некоторый сервис, который предоставляет каждый экземпляр или объект класса по требованию своих клиентов. Операции класса записываются в третьей сверху секции прямоугольника класса, поэтому эту секцию часто называют секцией операций. Совокупность операций характеризует функциональный аспект поведения всех объектов данного класса. Каждой операции класса соответствует отдельная строка, которая состоит из квантора видимости операции, имени, выражения типа возвращаемого значения.

Общий формат записи отдельной операции класса следующий:

<квантор видимости> <имя операции>(список параметров): <выражение типа возвращаемого значения>

Имя операции представляет собой строку текста, которая используется в качестве идентификатора соответствующей операции и поэтому должна быть уникальной в пределах данного класса. Имя операции является единственным обязательным элементом синтаксического обозначения операции, должно начинаться со строчной (малой) буквы и не должно содержать пробелов.

Список параметров является перечнем разделенных запятой формальных параметров, каждый из которых, в свою очередь, может быть представлен в следующем виде:

<вид параметра> <имя параметра> : <выражение типа> = <значение параметра по умолчанию>

Здесь вид параметра – одно из ключевых слов in, out или inout со значением in по умолчанию, в случае, если он не указывается. Имя параметра – это идентификатор соответствующего формального параметра, при записи которого следуют правилам задания имен атрибутов. Выражение типа – это спецификация типа данных для возвращаемого значения соответствующего формального параметра. Значение по умолчанию – это некоторое конкретное значение для этого формального параметра.

Выражение типа возвращаемого значения также указывает на тип данных значения, которое возвращается объектом после выполнения соответствующей операции. Двоеточие и выражение типа возвращаемого значения могут быть опущены, если операция не возвращает никакого значения. Для указания нескольких возвращаемых значений данный элемент спецификации операции может быть записан в виде списка отдельных выражений.

Обязательной частью строки записи операции является наличие имени операции и круглых скобок. Список формальных параметров и тип возвращаемого значения могут не указываться.

Кроме внутреннего устройства или структуры классов важную роль при разработке проектируемой системы имеют различные отношения между классами, которые также могут быть изображены на диаграмме классов. Однако совокупность допустимых типов таких отношений строго фиксирована. Базовыми отношениями на диаграммах классов являются:

– отношение ассоциации (association relationship);
– отношение обобщения (generalization relationship);
– отношение агрегации (aggregation relationship);
– отношение композиции (composition relationship);
– отношение зависимости (dependency relationship).

Отношение ассоциации соответствует наличию произвольного отношения между классами. Данное отношение обозначается сплошной линией со стрелкой или без нее. В качестве дополнительных специальных символов могут использоваться имя ассоциации, символ навигации, а также имена и кратность классов-ролей ассоциации.

Имя ассоциации является необязательным элементом ее обозначения. Однако если оно задано, то записывается с заглавной буквы рядом с линией соответствующей ассоциации. Отдельные классы ассоциации могут играть некоторую роль в соответствующем отношении, что может быть явно указано на диаграмме заданием имени концевых точек ассоциации.

Наиболее простой случай данного отношения – бинарная ассоциация. Она связывает в точности два различных класса и может быть ненаправленным (симметричным) или направленным отношением.

Ненаправленная бинарная ассоциация изображается линией без стрелки. Для нее на диаграмме может быть указан порядок чтения классов с использованием значка в форме треугольника рядом с именем данной ассоциации.

Направленная бинарная ассоциация изображается сплошной линией с простой стрелкой на одном из ее концевых точек. Направление этой стрелки указывает на то, какой из классов является первым, а какой – вторым (на него указывает стрелка ассоциации).

Следующий элемент обозначений – кратность ассоциации. Кратность относится к концам ассоциации и обозначается в виде интервала целых чисел, аналогично кратности атрибутов и операций классов, но без прямых скобок. Этот интервал записывается рядом с концом соответствующей ассоциации и означает потенциальное число отдельных экземпляров класса, которые могут иметь место.

Ассоциация является наиболее общей формой отношения. Все другие типы рассматриваемых отношений можно считать частным случаем данного отношения.

Отношение обобщения является отношением между более общим элементом (родителем) и более частным элементом (дочерним). Данное отношение может использоваться для представления иерархических взаимосвязей между пакетами, классами, вариантами использования.

Применительно к диаграмме классов данное отношение описывает наследование их свойств и поведения. Согласно принципу наследования, класс-потомок обладает всеми свойствами и поведением класса-предка, а также имеет свои собственные свойства и поведение, которые могут отсутствовать у класса-предка. На диаграммах отношение обобщения обозначается сплошной линией с треугольной стрелкой на одном из концов. Стрелка указывает на более общий класс (родитель).

В дополнение к простой стрелке-обобщению может быть присоединена строка текста, указывающая на некоторые специальные свойства этого отношения. Этот текст будет относиться ко всем линиям обобщения, которые идут к классам-потомкам.

В качестве ограничений могут быть использованы следующие ключевые слова:

– (complete) означает, что в данном отношении обобщений специфицированы все классы-потомки, и других классов-потомков у данного класса-предка быть не может;

– (incomplete) означает случай, противоположный первому, т. е. предполагается, что на диаграмме указаны не все классы-потомки. В последующем, возможно, разработчик восполнит их перечень, не изменяя уже построенную диаграмму.

Отношение агрегации имеет место между несколькими классами в том случае, если один из классов включает в себя в качестве составных частей другие классы.

Данное отношение применяется для представления системных взаимосвязей типа «часть – целое». Раскрывая внутреннюю структуру системы, отношение агрегации показывает, из каких элементов состоит система и как они связаны между собой. Графически отношение агрегации изображается сплошной линией, один из концов которой представляет собой не закрашенный внутри

ромб. Этот ромб указывает на тот из классов, который представляет собой класс-контейнер. Остальные классы являются его «частями».

Отношение композиции является частным случаем отношения агрегации. Специфика этой взаимосвязи заключается в том, что части не могут выступать в отрыве от целого, т. е. с уничтожением целого уничтожаются и все его составные части. Графически отношение композиции изображается сплошной линией, один из концов которой представляет собой закрашенный внутри ромб. Этот ромб указывает на тот из классов, который представляет собой класс-композит. Остальные классы являются его «частями».

Отношение зависимости используется в такой ситуации, когда некоторое изменение одного элемента модели может потребовать изменения другого, зависящего от него элемента модели. Отношение зависимости графически изображается пунктирной линией между соответствующими элементами со стрелкой на одном из ее концов. На диаграмме классов данное отношение связывает отдельные классы между собой, при этом стрелка направлена от класса-клиента зависимости или зависимого класса к классу-источнику или независимому классу.

Наиболее часто в качестве классов-клиентов и классов-источников зависимости могут выступать целые множества классов, организованные в пакеты. В этом случае стрелка должна быть направлена от пакета-клиента зависимости или зависимого пакета к пакету-источнику или независимому пакету.

При моделировании предметной области различают следующие виды классов.

Управляющий класс (control class), отвечающий за координацию действий других классов. У каждого варианта использования и диаграммы классов должен быть хотя бы один управляющий класс, контролирующий последовательность выполнения действий этого варианта использования. Этот класс является активным и инициирует рассылку множества сообщений другим классам модели. Кроме специального обозначения управляющий класс может быть изображен в форме обычного прямоугольника класса со стереотипом «control».

Класс-сущность (entity) содержит информацию, которая должна храниться постоянно и не уничтожаться с уничтожением объектов данного класса или прекращением работы моделируемой системы (выключением системы или завершением программы). Этот класс может соответствовать отдельной таблице базы данных, в этом случае его атрибуты могут быть полями таблицы, а операции – присоединенными процедурами. Как правило, этот класс является пассивным и лишь принимает сообщения от других классов модели. Класс-сущность может быть изображен также стандартным образом в форме обычного прямоугольника класса со стереотипом «entity».

Граничный класс располагается на границе системы с внешней средой, но является составной частью системы. Граничный класс может быть изображен также стандартным образом в форме обычного прямоугольника класса со стереотипом «boundary».

Интерфейс является специальным случаем класса, у которого имеются только операции и отсутствуют атрибуты. Для обозначения интерфейса исполь-

зуется специальный графический символ: окружность или стандартный способ – прямоугольник класса со стереотипом «interface». На диаграмме вариантов использования интерфейс изображается в виде маленького круга, рядом с которым записывается его имя. Интерфейсы на диаграмме служат для спецификации таких элементов модели, которые видимы извне, но их внутренняя структура остается скрытой от клиентов. Применительно к диаграммам классов, интерфейсы определяют совокупность операций, которые обеспечивают необходимый набор возможных действий. С системно-аналитической точки зрения интерфейс определяет общие границы проектируемой системы. Графическое изображение интерфейсов в форме окружности может использоваться и на других типах канонических диаграмм, например, диаграммах развертывания и диаграммах вариантов использования.

3.3 Реализация программного модуля.

При реализации интерфейса в пояснительной записке приводится описание основных компонентов для каждой формы, которые будут использоваться в программе. Типовой интерфейс приложения должен содержать:

– главную форму программы, которая раскрывается при запуске программы и включает следующий минимальный набор элементов: заголовок приложения и строку главного меню в верхней части формы. Режимы работы программы можно задавать с помощью переключателей;

– форму ввода исходных данных с клавиатуры;

– форму вывода результатов расчета, где расчетные данные отображаются в виде таблиц и графиков. Следует реализовать также возможность сохранения результатов в отдельном файле;

– форму вывода справочной информации по программе. В составе справки в программном модуле должны быть следующие разделы: краткая характеристика использованного численного метода; требования к аппаратной части компьютера; сведения об авторе программы.

В записке следует описать структуру интерфейса каждой формы, указав использованные стандартные классы и пояснив настройку их отдельных свойств.

В подразделе следует привести копию экрана с каждой из форм, реализованной в проекте, которые оформляются в виде рисунков согласно требованиям ГОСТ 2.105–95.

В данном пункте перечисляются возможные ошибки пользователя при работе с программным комплексом, указываются способы диагностики и защиты от этих ошибок.

При реализации программного модуля необходимо предусмотреть обработку следующих ошибок:

- неправильный ввод исходных данных;
- расхождение вычислительного процесса.

В последнем случае следует предусмотреть возможность прекращения вычислений и выдачи соответствующего сообщения.

Текст программы расчета разрабатывается на основании составленной блок-схемы алгоритма с использованием описанных в предыдущих разделах

данных. В тексте следует использовать комментарии для пояснений соответствия текста программы схеме алгоритма и локальных идентификаторов переменных и констант. Допускается при кодировании операторной части метода детализировать разработанный алгоритм, но запрещается пропускать указанные в схеме алгоритма операции.

Текст программы оформляется в виде распечатки на принтере в приложении А пояснительной записки. Допускается на листах распечатки не отображать основную надпись. Тексты программ рекомендуется распечатывать с использованием моноширинного шрифта (например, Courier New). Для текста приложений допустимой является высота шрифта не ниже 8 пт, а для основного текста записки – 14 пт.

Классы представляют собой модели объектов реального или абстрактного мира. При этом моделирование объектов реального мира – не единственная причина создания класса. Обычно класс создается для каждого объекта реального мира, моделируемого программой, после чего в класс следует поместить нужные объекту данные и создать сервисные методы, моделирующие поведение объекта.

Другой причиной создания класса является моделирование абстрактного объекта – объекта, который не существует в реальном мире, но является абстракцией других конкретных объектов. Пример – классический объект Shape (фигура). Объекты Circle (окружность) и Square (прямоугольник) существуют на самом деле, тогда как класс Shape – это абстракция конкретных фигур.

Самая важная причина создания класса – снижение сложности. Классы следует создавать для сокрытия информации, чтобы о ней можно было не думать. О ней придется думать при написании класса, но после этого можно использовать класс, не зная о его внутренней работе. Другие причины создания классов – минимизация объема кода, облегчение сопровождения программы и снижение числа ошибок – тоже хороши, но без абстрагирующей силы классов сложные программы было бы невозможно охватить умом. Как бы ни проявлялась сложность – в форме запутанных алгоритмов, крупных наборов данных, сложных протоколов коммуникации – она является источником ошибок. При возникновении ошибки ее будет проще найти, если она будет локализована в классе, а не распределена по всему коду. Изменения, обусловленные исправлением ошибки, не повлияют на остальной код, потому что придется исправить только один класс. Если найдется более эффективный, простой или надежный алгоритм, им будет легче заменить старый алгоритм, изолированный в классе. Во время разработки будет проще попробовать несколько вариантов проектирования и выбрать наилучший.

Следует изолировать области вероятных изменений, чтобы влияние изменений ограничивалось пределами одного или нескольких классов. Приложение следует проектировать так, чтобы области вероятных изменений можно было изменить с максимальной легкостью. В число областей вероятных изменений входят зависимости от оборудования, подсистема ввода/вывода, сложные типы данных и бизнес-правила.

Используя глобальные данные, можно скрыть детали реализации за интерфейсом класса. Обращение к глобальным данным через методы доступа имеет ряд преимуществ по сравнению с их непосредственным использованием: можно менять структуру данных, не изменяя программу; можно следить за доступом к данным. Кроме того, использование методов доступа подталкивает к размышлениям о том, на самом ли деле данные глобальны; часто оказывается, что «глобальные данные» на самом деле являются просто данными какого-то объекта.

Если один параметр передается в несколько методов, это может указывать на необходимость объединения этих методов в класс, чтобы они могли использовать параметр как данные объекта. Упрощение передачи параметров в методы само по себе не цель, но передача крупных объемов данных наводит на мысль, что другая организация классов могла бы быть более эффективной.

Код, разбитый на грамотно организованные классы, легче повторно использовать в других программах, чем тот же код, помещенный в один более крупный класс. Даже если фрагмент вызывается только из одного места программы и вполне понятен в составе более крупного класса, необходимо проанализировать, может ли он понадобиться в другой программе. Если да, стоит поместить его в отдельный класс.

Если предполагается, что программу придется изменять, разумно изолировать области предполагаемых изменений в отдельных классах. После этого можно изменять классы, не влияя на остальную часть программы, или вообще заменить их на абсолютно новые классы. Размышление о том, как может выглядеть целое семейство программ, а не просто одна программа, – эффективный эвристический принцип предвосхищения целых категорий изменений.

3.4 Тестирование программного модуля.

Методика тестирования программной системы может быть представлена в виде разворачивающейся спирали. Вначале осуществляется тестирование элементов, то есть модулей, проверяющее результаты этапа кодирования программной системы.

На втором шаге выполняется тестирование интеграции, ориентированное на выявление ошибок этапа проектирования программной системы.

На третьем обороте спирали производится тестирование правильности, проверяющее корректность этапа анализа требований к программной системе.

На заключительном этапе проводится системное тестирование, выявляющее недостатки этапа системного анализа программной системы.

Целью тестирования элементов является индивидуальная проверка каждого модуля, используя способы тестирования белого ящика. Цель тестирования интеграции заключается в оценке сборки модуля в программную систему. При этом применяются в основном способы тестирования черного ящика. Цель тестирования правильности – проверить реализацию в программной системе всех функциональных и поведенческих требований, а так же требований эффективности. На этом этапе используются исключительно способы тестирования черного ящика.

Цель системного тестирования – проверка правильности объединения и взаимодействия всех элементов компьютерной системы и реализация всех функций системы.

Организация процесса тестирования в виде эволюционной развивающейся спирали обеспечивает максимальную эффективность поиска ошибок.

В данном пункте должны быть представлены:

1) условия проверки корректности введенных данных;

2) точность получения результата;

3) отображение хода процесса выполнения расчета;

4) возможность прерывания пользователем выполнения расчета и (или) автоматическое определение режима «зависания» или некорректных результатов;

5) вид результатов расчета.

Результаты, полученные при обработке тестовых исходных данных, сверяются с тестовыми результатами, которые должны быть рассчитаны другими способами, методами либо программными продуктами (Excel, MathCAD, MATLAB и т. п.). При их сопоставлении определяется идентичность результатов или их отклонения и делается заключение о правильности работы созданного приложения. Необходимо привести копию рабочей области документа, выполненного с использованием выбранного иного программного продукта, и сделать вывод о соответствии результатов расчетов, полученных в разработанном программном модуле и иным способом (средствами Excel, MathCAD, MATLAB) либо другим математическим методом.

4 Реализация выбора и принятие решения.

В данном разделе следует описать процедуры выбора оптимальных или рациональных значений параметров системы и процесс принятия решений по результатам работы программного модуля.

Для решения задач выбора существуют различные подходы, наиболее распространенный из которых – критериальный подход. Основным предположением критериального подхода является следующее: каждую отдельно взятую альтернативу можно оценить конкретным числом значением критерия. Сравнение альтернатив сводится к сравнению результатов расчетов соответствующих критериев качества, целевых функций. Если далее предположить, что выбор любой альтернативы приводит к однозначно определяемым последствиям и заданный критерий численно выражает оценку этих последствий, то наилучшей альтернативой является та, которая обладает наибольшим значением критерия. Если в результате сравнения по нескольким критериям получилось, что одна альтернатива обладает наилучшими значениями по всем критериям, то выбор не представляет затруднений, именно эта альтернатива и будет наилучшей.

В случае необходимости решения многокритериальных задач необходимо использовать известные подходы к решению таких задач: метод сведения многокритериальной задачи к однокритериальной, поиск альтернативы с заданными свойствами, нахождение паретовского множества альтернатив.

Для задач, у которых целевая функция представляет собой дискретное множество значений, вместо оптимизации используется рациональный выбор параметров из некоторого ограниченного множества значений.

Организация последовательного поиска рационального решения может осуществляться путём перебора множества комбинаций значений входных параметров. В итоге по результатам исследований формируются матрицы откликов для принятия решений по выбору рационального набора параметров системы на основе метода секущих плоскостей. Каждый из входных параметров изменяется на пяти уровнях: минимальное значение, максимальное значение, среднее значение, и два промежуточных значения между минимальным и средним, а также между максимальным и средним значениями. Все входные параметры, кроме одной, фиксируются в области средних значений, а одна характеристика меняется на пяти указанных выше уровнях значений. Рациональным считается значение выбранного параметра, обеспечивающего лучшее значение критериев. Полученное значение выбранного параметра фиксируется, выбирается второй изменяемый параметр и определяется его рациональное значение. Процесс продолжается до тех пор, пока не будут выбраны рациональные значения всех параметров системы.

Формирование обобщённого показателя качества может быть реализовано по двум методикам. Во-первых, можно использовать метод главного критерия, если выбрать в качестве главного любой выходной параметр. Во-вторых, можно использовать «свёртку» компонентов вектора откликов к скаляру, применив метод весовых коэффициентов важности откликов для исследователя: для каждого из откликов задаётся коэффициент важности $\Sigma\delta nc_h$ (причём сумма коэффициентов важности $\Sigma\delta nc_h = 1$), а затем вычисляется значение обобщённого отклика как сумма произведений коэффициентов важности на значения откликов $Wnc_i = \sum_h Ync_{ih} \cdot \delta nc_h$.

Для выбора рационального набора параметров подсистемы на основе классических критериев принятия решений осуществляется приведение откликов модели к одному типу и масштабу. В итоге формируется множество векторов компонентов, требующих максимизации и изменяющихся на отрезке [0,1]. Исследователь может принять решение по одной из следующих стратегий: нейтралитета, оптимиста, пессимиста, усреднённой, на основе критерия Сэвиджа. Выбор комбинации параметров осуществляется путём сопоставления вариантов. Выбранная комбинация обеспечивает максимум значения обобщённого критерия.

Заключение.

В заключении необходимо проанализировать полученные в ходе выполнения курсовой работы результаты. Отмечается перечень выполненных разработок. Дается оценка достигнутых результатов интеллектуального анализа данных, полноты решения поставленной задачи и принятого решения. Отражаются сильные и слабые стороны разработанного приложения, даются рекомендации по его дальнейшему применению и развитию.

Литература.

Оформление списка литературных источников, используемых при выполнении курсовой работы, производится в соответствии с требованиями ГОСТ 7.1–2003 *Библиографическое описание документа. Общие требования и правила составления*. Необходимо обратить внимание на то, что список литературы в данных методических указаниях оформлен в соответствии с требованиями указанного стандарта. Ссылка на информационные ресурсы Интернет оформляется в виде полного URL-адреса ресурса, названия источника и даты доступа.

5 Оформление курсовой работы

Пояснительную записку (ПЗ) следует оформлять в соответствии с требованиями ГОСТ 2.105–95 *Общие требования к текстовым документам* на листах формата А4.

Текст должен быть набран на ЭВМ (шрифтом Times New Roman высотой 14 pt, первая строка – отступ 1,25 см, межстрочный интервал – множитель 1,15). Допускается оформление пояснительной записки вручную, но при условии, что текст и формулы будут написаны четко и аккуратно чертежным шрифтом по ГОСТ 2.304.

Все листы ПЗ, включая графики, схемы, таблицы и приложения (кроме титульного листа и технического задания на проектирование), должны содержать стандартную рамку и иметь сквозную нумерацию страниц. Титульный лист и задание на курсовую работу не нумеруются, но при подсчете числа страниц считаются соответственно первым и вторым листами. Основную надпись третьей страницы «Содержание» в соответствии с ГОСТ 2.104 оформляют по форме 2 высотой 40 мм, а остальных страниц – по форме 2а высотой 15 мм.

Расстояние от рамки формы до границ текста в начале и в конце строк должно быть не менее 3 мм. Расстояние от верхней или нижней строки текста до верхней или нижней рамки – не менее 10 мм. Абзацы в тексте начинают отступом, равным 1,27 см.

Заголовки разделов и подразделов следует печатать с абзацного отступа с прописной буквы без точки в конце. Переносы слов в названиях заголовков не допускаются. Наименования разделов «Содержание», «Введение» и «Список литературы» располагают симметрично тексту (в центре). Каждый новый раздел следует начинать с новой страницы. Расстояние между заголовком и текстом при наборе ПЗ на компьютере должно составлять два интервала. Расстояние между заголовками раздела и подраздела – один интервал, а при выполнении рукописным способом – 8 мм.

Перед каждой позицией перечисления требований, положений и т. п. ставят дефис или, при необходимости ссылки в тексте ПЗ, строчную букву, после которой ставится скобка. При этом перечисления записывают с абзаца, но со строчной буквы и разделяют между собой точкой с запятой.

При использовании формул, научно-технических положений, стандартов и других данных необходимо делать ссылку на литературный источник, указывая его номер из списка литературы в квадратных скобках. Список литературы составляется либо по алфавиту, либо по мере появления ссылок в тексте ПЗ и оформляется в соответствии с ГОСТ 7.1–2003.

Формулы записывают с новой строки по центру. Формулы, следующие одна за другой и не разделенные текстом, разделяют точкой с запятой. При этом за последней формулой ставят либо запятую (если необходима расшифровка символов формул), либо точку. Все формулы должны нумероваться сквозной нумерацией арабскими цифрами, которые записывают на уровне формулы справа в круглых скобках. Допускается нумерация формул в пределах раздела. В этом случае номер формулы состоит из номера раздела и порядкового номера формулы, разделенных точкой, например (2.7). Пояснения символов, входящих в формулы, если они не пояснены ранее в тексте, должны быть приведены непосредственно под формулой. Пояснения каждого символа следует давать с новой строки в той последовательности, в которой символы приведены в формуле. Первая строка пояснения должна начинаться словом «где» (без двоеточия после него), затем ставят пробел и приводят обозначение символа, и далее через дефис дают описание физического смысла символа с указанием его размерности.

Пример – Математическая модель задачи имеет вид:

$$\begin{cases} F = 25x_1 + 12x_2 + 10x_3 + 11x_4 \to \max; \\ \begin{cases} 4x_1 + 1{,}5x_2 + x_3 + 2x_4 \le 50; \\ 4x_1 + 0{,}5x_2 + 0{,}5x_3 + 2x_4 \le 65; \end{cases} \\ x_j \ge 0, \end{cases} \tag{1.1}$$

где x_1, x_2, x_3, x_4 – неизвестные задачи, д. е.

Иллюстрации (рисунки и графики) и таблицы располагают в записке непосредственно после текста, в котором они упоминаются впервые, или на следующей странице, если в указанном месте они не помещаются. Нумерацию иллюстраций и таблиц выполняют в пределах раздела или всей записки арабскими цифрами. Иллюстрация должна иметь название, которое помещают под ней с абзацного отступа, например, первый рисунок третьего раздела подписывается так: Рисунок 3.1 – Блок-схема обобщенного алгоритма функционирования приложения. При необходимости перед названием иллюстрации помещают подрисуночный текст, поясняющий содержание иллюстрации.

Названия таблиц располагают сверху, начиная с их левого верхнего угла, с абзаца, например, третья таблица второго раздела подписывается так: Таблица 2.3 – Платежная матрица.

На все иллюстрации и таблицы в тексте должны быть даны ссылки, при этом слова «Рисунок» и «Таблица» пишутся полностью, например: «из рисунка 2.1 следует …», «в таблице 2.6 приведены …».

Если строки или графы таблицы выходят за формат страницы, ее делят на части, помещая одну часть под другой, при этом в каждой части таблицы повторяют заголовки ее граф и боковик. При делении таблицы на части допускается заголовки ее граф или боковик заменять номерами ее граф и (или) строк. При этом нумеруют (арабскими цифрами) также и графы и (или) строки первой части таблицы. Название «Таблица 2.4 – Результаты исследований» указывают один раз над первой частью таблицы, над другими частями пишут слова «Продолжение таблицы 2.1».

Приложения оформляют как продолжение ПЗ. Каждое приложение должно начинаться с новой страницы с указанием наверху посередине страницы слова «ПРИЛОЖЕНИЕ» и его обозначения (прописными буквами русского алфавита, начиная с буквы А), а под ними в круглых скобках указывают вид приложения (обязательное, рекомендуемое или справочное). Далее с новой строки симметрично тексту записывают с прописной буквы заголовок приложения. При ссылках на приложение в тексте следует писать, например, «в соответствии с приложением А …».

Список литературы

1 **Дадян, Э. Г.** Методы, модели, средства хранения и обработки данных: учебник [Электронный ресурс] / Э. Г Дадян, Ю. А. Зеленков. – Москва: Вузовский учебник; ИНФРА-М, 2017. – 168 с. – Режим доступа: http://znanium.com/. – Дата доступа 01.06.2020.

2 **Дивеев, А. И.** Применение метода вариационного аналитического программирования для синтеза управления летающим роботом / А. И. Дивеев, Н. Б. Конырбаев // Фундаментальные исследования. – 2015. – № 3. – С. 51–57.

3 **Дюк, В. А.** Применение технологий интеллектуального анализа данных в естественнонаучных, технических и гуманитарных областях / В. А. Дюк, А. В. Флегонтов, И. К. Фомина // Изв. Рос. гос. пед. ун-та им. А. И. Герцена. – 2011. – № 138. – С. 77–84.

4 **Замятин, А. В.** Интеллектуальный анализ данных: учебное пособие / А. В. Замятин. – Томск: Томский гос. ун-т, 2016. – 120 с.

5 **Митина, О. В.** Методы анализа текста: методологические основания и программная реализация / О. В. Митина, А. С. Евдокименко // Вестн. ЮУрГУ. – 2010. – № 40. – С. 29–38.

6 **Oplatkova, Z.** Investigation on Artificial Ant using Analytic Programming Commands / Z. Oplatkova, I. Zelinka // GECCO’06, July 8–12, 2006, Seattle, Washington, USA. – P. 949–950.

7 **Oplatkova, Z.** Investigation on Evolutionary Synthesis of Movement Commands / Z. Oplatkova, I. Zelinka // Modelling and Simulation in Engineering. – 2009. – Vol. 20. – 12 p.

8 **Zelinka, I.** Analytic Programming – Symbolic Regression by Means of Arbitrary Evolutionary Algorithms / I. Zelinka, Z. Oplatkova, L. Nolle // I. J. of Simulation. – 2005. – Vol. 6, № 9. – P. 44–56.