

Документ подписан простой электронной подписью

Информация о владельце:

ФИО: Фурсов Владимир Алексеевич

Должность: И.о. директора Пятигорского института (филиал) Северо-Кавказского
Федерального государственного автономного образовательного учреждения высшего образования
федерального университета

Дата подписания: 04.06.2026 10:17:57

Уникальный программный ключ:

1c378726a41fd0143ae5bcc8ba81860b00daa62

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ СИСТЕМНЫЙ АНАЛИЗ ДАННЫХ И МОДЕЛИ ПРИНЯТИЯ РЕШЕНИЙ

Направление подготовки

09.04.02

Направленность (профиль)

**Информационные системы и технологии
Технологии работы с данными и знаниями,
анализ информации**

Квалификация выпускника

Магистр

Пятигорск, 2026

СОДЕРЖАНИЕ

1. Цель и задачи освоения дисциплины	3
2. Место дисциплины в структуре основной образовательной программы	3
3. Наименование лабораторных работ	3
4. Содержание лабораторных работ	3
Лабораторная работа № 1. Анализ данных и проектирование систем	3
Диаграммы классов. Общие сведения.....	3
Класс.....	3
Управляющие классы	5
Пакет. Механизм пакетов.....	10
5. Типовые контрольные задания и иные материалы, характеризующие этапы формирования компетенций	15
6. Учебно-методическое и информационное обеспечение дисциплины	15

1. Цель и задачи освоения дисциплины

Целью освоения дисциплины является формирование набора профессиональных компетенций будущего магистра по направлению 09.04.02 Информационные системы и технологии для решения прикладных задач в рамках направленности (профиля) «Технологии работы с данными и знаниями, анализ информации».

Задачи освоения дисциплины: изучение основных понятий системного анализа данных и теории принятия решений, освоение методов и инструментов системного анализа данных и моделирования алгоритмов принятия решений.

2. Место дисциплины в структуре основной образовательной программы

Дисциплина «Системный анализ данных и модели принятия решений» относится к дисциплинам обязательной части.

3. Наименование лабораторных работ

Тема 2. Анализ данных и проектирование систем

Модели данных. Алгоритмы анализа данных. Методы Байеса. Вероятностный подход к построению графа принятия решения. Методы классификации и кластеризации. Методы DATA MINING

4. Содержание лабораторных работ

Лабораторная работа № 1. Анализ данных и проектирование систем

Форма проведения: решение проблемных задач

Ход лабораторной работы:

1. Провести анализ данных поставленной задачи
2. Определить методы классификации и кодирования данных
3. Структурировать данные
4. Подвести итоги проведенного исследования

Вопросы для обсуждения:

Системное проектирование. Структурное проектирование. Логическое и физическое моделирование данных. Техническое и технологическое проектирование. Техническое задание. Техничко-экономическое обоснование.

Теоретический материал

Диаграммы классов. Общие сведения

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.

Диаграмма классов UML - это граф, узлами которого являются элементы статической структуры проекта (классы, интерфейсы), а дугами - отношения между узлами (ассоциации, наследование, зависимости).

На диаграмме классов изображаются следующие элементы:

- Пакет (package) - набор элементов модели, логически связанных между собой;
- Класс (class) - описание общих свойств группы сходных объектов;
- Интерфейс (interface) - абстрактный класс, задающий набор операций, которые объект произвольного класса, связанного с данным интерфейсом, предоставляет другим объектам.

Класс

Класс - это группа сущностей (объектов), обладающих сходными свойствами, а именно, данными и поведением. Отдельный представитель некоторого класса называется объектом класса или просто объектом.

Под поведением объекта в UML понимаются любые правила взаимодействия объекта с внешним миром и с данными самого объекта.

На диаграммах класс изображается в виде прямоугольника со сплошной границей, разделенного горизонтальными линиями на 3 секции:

- Верхняя секция (секция имени) содержит имя класса и другие общие свойства (в частности, стереотип).
- В средней секции содержится список атрибутов
- В нижней - список операций класса, отражающих его поведение (действия, выполняемые классом).

Любая из секций атрибутов и операций может не изображаться (а также обе сразу). Для отсутствующей секции не нужно рисовать разделительную линию и как-либо указывать на наличие или отсутствие элементов в ней.

На усмотрение конкретной реализации могут быть введены дополнительные секции, например, исключения (Exceptions).

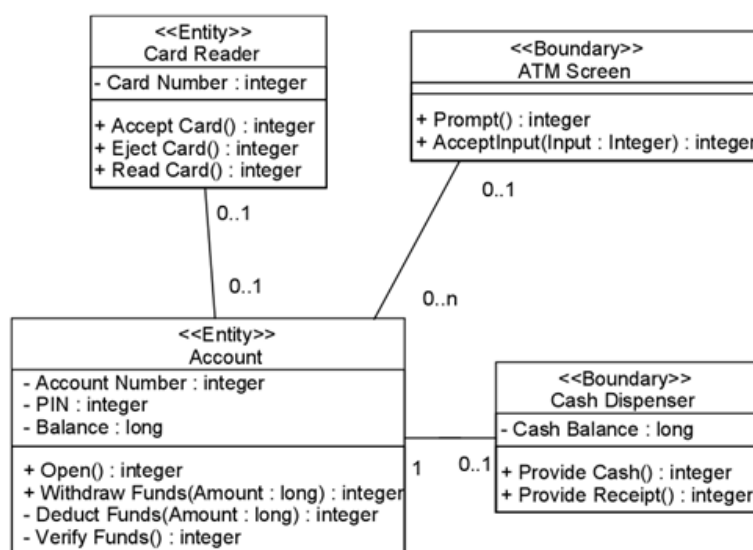


Рисунок 3.9 - Пример диаграммы классов

Стереотипы классов

Стереотипы классов – это механизм, позволяющий разделять классы на категории. В языке UML определены три основных стереотипа классов:

- Boundary (граница);
- Entity (сущность);
- Control (управление).

Граничные классы

Граничными классами (boundary classes) называются такие классы, которые расположены на границе системы и всей окружающей среды. Это экранные формы, отчеты, интерфейсы с аппаратурой (такой как принтеры или сканеры) и интерфейсы с другими системами.

Чтобы найти граничные классы, надо исследовать диаграммы вариантов использования. Каждому взаимодействию между действующим лицом и вариантом использования должен соответствовать, по крайней мере, один граничный класс. Именно такой класс позволяет действующему лицу взаимодействовать с системой.

Классы-сущности

Классы-сущности (entity classes) содержат хранимую информацию. Они имеют наибольшее значение для пользователя, и потому в их названиях часто используют термины из предметной области. Обычно для каждого класса-сущности создают таблицу в базе данных.

Управляющие классы

Управляющие классы (control classes) отвечают за координацию действий других классов. Обычно у каждого варианта использования имеется один управляющий класс, контролирующий последовательность событий этого варианта использования. Управляющий класс отвечает за координацию, но сам не несет в себе никакой функциональности, так как остальные классы не посылают ему большого количества сообщений. Вместо этого он сам посылает множество сообщений. Управляющий класс просто делегирует ответственность другим классам, по этой причине его часто называют классом-менеджером.

В системе могут быть и другие управляющие классы, общие для нескольких вариантов использования. Например, может быть класс SecurityManager (менеджер безопасности), отвечающий за контроль событий, связанных с безопасностью. Класс TransactionManager (менеджер транзакций) занимается координацией сообщений, относящихся к транзакциям с базой данных. Могут быть и другие менеджеры для работы с другими элементами функционирования системы, такими как разделение ресурсов, распределенная обработка данных или обработка ошибок.

Помимо упомянутых выше стереотипов можно создавать и свои собственные.

Атрибуты

Атрибут – это элемент информации, связанный с классом. Атрибуты хранят инкапсулированные данные класса.

Так как атрибуты содержатся внутри класса, они скрыты от других классов. В связи с этим может понадобиться указать, какие классы имеют право читать и изменять атрибуты. Это свойство называется видимостью атрибута (attribute visibility).

У атрибута можно определить четыре возможных значения этого параметра:

- Public (общий, открытый). Это значение видимости предполагает, что атрибут будет виден всеми остальными классами. Любой класс может просмотреть или изменить значение атрибута. В соответствии с нотацией UML общему атрибуту предшествует знак « + ».

- Private (закрытый, секретный). Соответствующий атрибут не виден никаким другим классом. Закрытый атрибут обозначается знаком « - » в соответствии с нотацией UML.

- Protected (защищенный). Такой атрибут доступен только самому классу и его потомкам. Нотация UML для защищенного атрибута – это знак « # ».

- Package or Implementation (пакетный). Предполагает, что данный атрибут является общим, но только в пределах его пакета. Этот тип видимости не обозначается никаким специальным значком.

В общем случае, атрибуты рекомендуется делать закрытыми или защищенными. Это позволяет лучше контролировать сам атрибут и код.

С помощью закрытости или защищенности удастся избежать ситуации, когда значение атрибута изменяется всеми классами системы. Вместо этого логика изменения атрибута будет заключена в том же классе, что и сам этот атрибут. Задаваемые параметры видимости повлияют на генерируемый код.

Операции

Операции реализуют связанное с классом поведение. Операция включает три части – имя, параметры и тип возвращаемого значения.

Параметры – это аргументы, получаемые операцией «на входе». Тип возвращаемого значения относится к результату действия операции.

На диаграмме классов можно показывать как имена операций, так и имена операций вместе с их параметрами и типом возвращаемого значения. Чтобы уменьшить загруженность диаграммы, полезно бывает на некоторых из них показывать только имена операций, а на других их полную сигнатуру.

В языке UML операции имеют следующую нотацию:

Имя Операции (аргумент: тип данных аргумента, аргумент2:тип данных аргумента2,...): тип возвращаемого значения

Следует рассмотреть четыре различных типа операций:

- Операции реализации;
- Операции управления;
- Операции доступа;
- Вспомогательные операции.

Операции реализации

Операции реализации (implementor operations) реализуют некоторые бизнес-функции. Такие операции можно найти, исследуя диаграммы взаимодействия. Диаграммы этого типа фокусируются на бизнес-функциях, и каждое сообщение диаграммы, скорее всего, можно соотнести с операцией реализации.

Каждая операция реализации должна быть легко прослеживаема до соответствующего требования. Это достигается на различных этапах моделирования. Операция выводится из сообщения на диаграмме взаимодействия, сообщения исходят из подробного описания потока событий, который создается на основе варианта использования, а последний – на основе требований. Возможность проследить всю эту цепочку позволяет гарантировать, что каждое требование будет реализовано в коде, а каждый фрагмент кода реализует какое-то требование.

Операции управления

Операции управления (manager operations) управляют созданием и уничтожением объектов. В эту категорию попадают конструкторы и деструкторы классов.

Операции доступа

Атрибуты обычно бывают закрытыми или защищенными. Тем не менее, другие классы иногда должны просматривать или изменять их значения. Для этого существуют операции доступа (access operations). Такой подход дает возможность безопасно инкапсулировать атрибуты внутри класса, защитив их от других классов, но все же позволяет осуществить к ним контролируемый доступ. Создание операций Get и Set (получения и изменения значения) для каждого атрибута класса является стандартом.

Вспомогательные операции

Вспомогательными (helper operations) называются такие операции класса, которые необходимы ему для выполнения его обязанностей, но о которых другие классы не должны ничего знать. Это закрытые и защищенные операции класса.

Чтобы идентифицировать операции, выполните следующие действия:

Изучите диаграммы последовательности и кооперативные диаграммы. Большая часть сообщений на этих диаграммах является операциями реализации. Рефлексивные сообщения будут вспомогательными операциями.

Рассмотрите управляющие операции. Может потребоваться добавить конструкторы и деструкторы.

Рассмотрите операции доступа. Для каждого атрибута класса, с которым должны будут работать другие классы, надо создать операции Get и Set.

Связи

Связь представляет собой семантическую взаимосвязь между классами. Она дает классу возможность узнавать об атрибутах, операциях и связях другого класса. Иными словами, чтобы один класс мог послать сообщение другому на диаграмме последовательности или кооперативной диаграмме, между ними должна существовать связь.

Существуют четыре типа связей, которые могут быть установлены между классами: ассоциации, зависимости, агрегации и обобщения.

Ассоциации

Ассоциация (association) – это семантическая связь между классами. Их рисуют на диаграмме классов в виде обыкновенной линии.



Рисунок 3.10 - Связь ассоциация

Ассоциации могут быть двунаправленными, как в примере, или однонаправленными. На языке UML двунаправленные ассоциации рисуют в виде простой линии без стрелок или со стрелками с обеих ее сторон. На однонаправленной ассоциации изображают только одну стрелку, показывающую ее направление.

Направление ассоциации можно определить, изучая диаграммы последовательности и кооперативные диаграммы. Если все сообщения на них отправляются только одним классом и принимаются только другим классом, но не наоборот, между этими классами имеет место однонаправленная связь. Если хотя бы одно сообщение отправляется в обратную сторону, ассоциация должна быть двунаправленной.

Ассоциации могут быть рефлексивными. Рефлексивная ассоциация предполагает, что один экземпляр класса взаимодействует с другими экземплярами этого же класса.

Зависимости

Связи зависимости (dependency) также отражают связь между классами, но они всегда однонаправлены и показывают, что один класс зависит от определений, сделанных в другом. Например, класс А использует методы класса В. Тогда при изменении класса В необходимо произвести соответствующие изменения в классе А.

Зависимость изображается пунктирной линией, проведенной между двумя элементами диаграммы, и считается, что элемент, привязанный к концу стрелки, зависит от элемента, привязанного к началу этой стрелки.



Рисунок 3.11 - Связь зависимость

При генерации кода для этих классов к ним не будут добавляться новые атрибуты. Однако, будут созданы специфические для языка операторы, необходимые для поддержки связи.

Агрегации

Агрегации (aggregations) представляют собой более тесную форму ассоциации. Агрегация – это связь между целым и его частью. Например, у вас может быть класс Автомобиль, а также классы Двигатель, Покрышки и классы для других частей автомобиля. В результате объект класса Автомобиль будет состоять из объекта класса Двигатель, четырех объектов Покрышек и т. д. Агрегации визуализируют в виде линии с ромбом у класса, являющегося целым:



Рисунок 3.12 - Связь агрегация

В дополнение к простой агрегации UML вводит более сильную разновидность агрегации, называемую композицией. Согласно композиции, объект-часть может принадлежать только единственному целому, и, кроме того, как правило, жизненный цикл частей совпадает с циклом целого: они живут и умирают вместе с ним. Любое удаление целого распространяется на его части.

Такое каскадное удаление нередко рассматривается как часть определения агрегации, однако оно всегда подразумевается в том случае, когда множественность роли составляет 1..1; например, если необходимо удалить Клиента, то это удаление должно распространиться и на Заказы (и, в свою очередь, на Строки заказа).

Обобщения (Наследование)

Обобщение (наследование) - это отношение типа общее-частное между элементами модели. С помощью обобщений (generalization) показывают связи наследования между двумя классами.

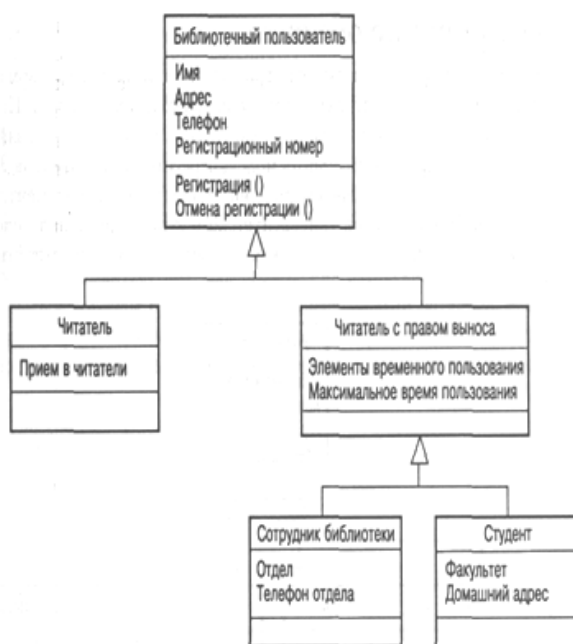


Рисунок 3.13 - Пример связи наследование

Большинство объектно-ориентированных языков непосредственно поддерживают концепцию наследования. Она позволяет одному классу наследовать все атрибуты, операции и связи другого. Наследование пакетов означает, что в пакете-наследнике все сущности пакета-предка будут видны под своими собственными именами (т.е. пространства имен объединяются). Наследование показывается сплошной линией, идущей от класса-потомка к классу-предку (в терминологии ООП - от потомка к предку, от сына к отцу, или от подкласса к суперклассу). Со стороны более общего элемента рисуется большой полый треугольник.

Помимо наследуемых, каждый подкласс имеет свои собственные уникальные атрибуты, операции и связи.

Множественность

Множественность (multiplicity) показывает, сколько экземпляров одного класса взаимодействуют с помощью этой связи с одним экземпляром другого класса в данный момент времени.

Например, при разработке системы регистрации курсов в университете можно определить классы Course (курс) и Student (студент). Между ними установлена связь: у курсов могут быть студенты, а у студентов – курсы. Вопросы, на который должен ответить параметр множественности: «Сколько курсов студент может посещать в данный момент? Сколько студентов может за раз посещать один курс?»

Так как множественность дает ответ на оба эти вопроса, её индикаторы устанавливаются на обоих концах линии связи. В примере регистрации курсов мы решили, что один студент может посещать от нуля до четырех курсов, а один курс могут слушать от 0 до 20 студентов.

В языке UML приняты определенные нотации для обозначения множественности.

Таблица 3.1 - Обозначения множественности связей в UML

Множественность	Значение
0..*	Ноль или больше
1..*	Один или больше
0..1	Ноль или один
1..1 (сокращенная запись: 1)	Ровно один

Имена связей

Связи можно уточнить с помощью имен связей или ролевых имен. Имя связи – это обычно глагол или глагольная фраза, описывающая, зачем она нужна. Например, между классом Person (человек) и классом Company (компания) может существовать ассоциация. Можно задать в связи с этим вопрос, является ли объект класса Person клиентом компании, её сотрудником или владельцем? Чтобы определить это, ассоциацию можно назвать «employs» (нанимает):

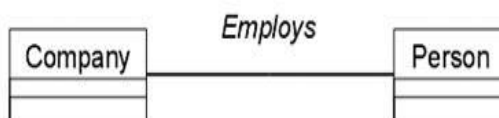


Рисунок 3.14 - Пример имен связей

Роли

Ролевые имена применяют в связях ассоциации или агрегации вместо имен для описания того, зачем эти связи нужны. Возвращаясь к примеру с классами Person и Company, можно сказать, что класс Person играет роль сотрудника класса Company. Ролевые имена – это обычно имена существительные или основанные на них фразы, их показывают на диаграмме рядом с классом, играющим соответствующую роль. Как правило, пользуются или ролевым именем, или именем связи, но не обоими сразу. Как и имена связей, ролевые имена не обязательны, их дают, только если цель связи не очевидна. Пример ролей приводится ниже:

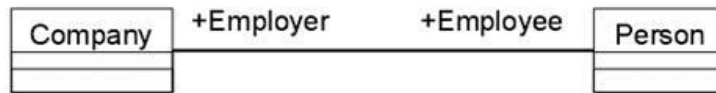


Рисунок 3.15 - Пример ролей связей

Пакет. Механизм пакетов

В контексте диаграмм классов, пакет - это вместилище для некоторого набора классов и других пакетов. Пакет является самостоятельным пространством имен.

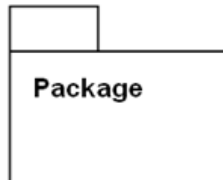


Рисунок 3.16 - Обозначение пакета в UML

В UML нет каких-либо ограничений на правила, по которым разработчики могут или должны группировать классы в пакеты. Но есть некоторые стандартные случаи, когда такая группировка уместна, например, тесно взаимодействующие классы, или более общий случай - разбиение системы на подсистемы.

Пакет физически содержит сущности, определенные в нем (говорят, что "сущности принадлежат пакету"). Это означает, что если будет уничтожен пакет, то будут уничтожены и все его содержимое.

Существует несколько наиболее распространенных подходов к группировке.

Во-первых, можно группировать их по стереотипу. В таком случае получается один пакет с классами-сущностями, один с граничными классами, один с управляющими классами и т.д. Этот подход может быть полезен с точки зрения размещения готовой системы, поскольку все находящиеся на клиентских машинах пограничные классы уже оказываются в одном пакете.

Другой подход заключается в объединении классов по их функциональности. Например, в пакете Security (безопасность) содержатся все классы, отвечающие за безопасность приложения. В таком случае другие пакеты могут называться Employee Maintenance (Работа с сотрудниками), Reporting (Подготовка отчетов) и Error Handling (Обработка ошибок). Преимущество этого подхода заключается в возможности повторного использования.

Механизм пакетов применим к любым элементам модели, а не только к классам. Если для группировки классов не использовать некоторые эвристики, то она становится произвольной. Одна из них, которая в основном используется в UML, – это зависимость. Зависимость между двумя пакетами существует в том случае, если между любыми двумя классами в пакетах существует любая зависимость.

Таким образом, диаграмма пакетов представляет собой диаграмму, содержащую пакеты классов и зависимости между ними. Строго говоря, пакеты и зависимости являются элементами диаграммы классов, то есть диаграмма пакетов – это форма диаграммы классов.

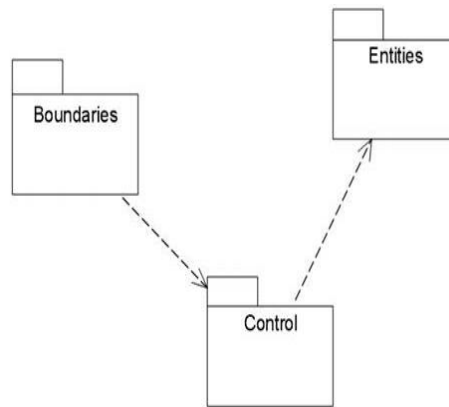


Рисунок 3.17 - Пример диаграммы пакетов

Зависимость между двумя элементами имеет место в том случае, если изменения в определении одного элемента могут повлечь за собой изменения в другом. Что касается классов, то причины для зависимостей могут быть самыми разными:

- один класс посылает сообщение другому;
- один класс включает часть данных другого класса; один класс использует другой в качестве параметра операции.

Если класс меняет свой интерфейс, то любое сообщение, которое он посылает, может утратить свою силу.

Пакеты не дают ответа на вопрос, каким образом можно уменьшить количество зависимостей в вашей системе, однако они помогают выделить эти зависимости, а после того, как они все окажутся на виду, остается только поработать над снижением их количества. Диаграммы пакетов можно считать основным средством управления общей структурой системы.

Пакеты являются жизненно необходимым средством для больших проектов. Их следует использовать в тех случаях, когда диаграмма классов, охватывающая всю систему в целом и размещенная на единственном листе бумаги формата А4, становится нечитаемой.

Диаграммы состояний

Диаграммы состояний определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате наступления некоторых событий.

Существует много форм диаграмм состояний, незначительно отличающихся друг от друга семантикой.

На диаграмме имеются два специальных состояния – начальное (start) и конечное (stop). Начальное состояние выделено черной точкой, оно соответствует состоянию объекта, когда он только что был создан. Конечное состояние обозначается черной точкой в белом кружке, оно соответствует состоянию объекта непосредственно перед его уничтожением. На диаграмме состояний может быть одно и только одно начальное состояние. В то же время, может быть столько конечных состояний, сколько вам нужно, или их может не быть вообще. Когда объект находится в каком-то конкретном состоянии, могут выполняться различные процессы. Процессы, происходящие, когда объект находится в определенном состоянии, называются действиями (actions).

С состоянием можно связывать данные пяти типов: деятельность, входное действие, выходное действие, событие и история состояния.

Деятельность

Деятельностью (activity) называется поведение, реализуемое объектом, пока он находится в данном состоянии. Деятельность – это прерываемое поведение. Оно может выполняться до своего завершения, пока объект находится в данном состоянии, или может быть прервано переходом объекта в другое состояние. Деятельность изображают внутри самого состояния, ей должно предшествовать слово do (делать) и двоеточие.

Входное действие

Входным действием (entry action) называется поведение, которое выполняется, когда объект переходит в данное состояние. Данное действие осуществляется не после того, как объект перешел в это состояние, а, скорее, как часть этого перехода. В отличие от деятельности, входное действие рассматривается как непрерываемое. Входное действие также показывают внутри состояния, ему предшествует слово entry (вход) и двоеточие.

Выходное действие

Выходное действие (exit action) подобно входному. Однако, оно осуществляется как составная часть процесса выхода из данного состояния. Оно является частью процесса такого перехода. Как и входное, выходное действие является непрерываемым.

Выходное действие изображают внутри состояния, ему предшествует слово exit (выход) и двоеточие.

Поведение объекта во время деятельности, при входных и выходных действиях может включать отправку события другому объекту. В этом случае описанию деятельности, входного действия или выходного действия предшествует знак « ^ ».

Соответствующая строка на диаграмме выглядит как

Do: ^Цель.Событие (Аргументы)

Здесь Цель – это объект, получающий событие, Событие – это посылаемое сообщение, а Аргументы являются параметрами посылаемого сообщения.

Деятельность может также выполняться в результате получения объектом некоторого события. При получении некоторого события выполняется определенная деятельность.

Переходом (Transition) называется перемещение из одного состояния в другое. Совокупность переходов диаграммы показывает, как объект может перемещаться между своими состояниями. На диаграмме все переходы изображают в виде стрелки, начинающейся на первоначальном состоянии и заканчивающейся последующим.

Переходы могут быть рефлексивными. Объект может перейти в то же состояние, в котором он в настоящий момент находится. Рефлексивные переходы изображают в виде стрелки, начинающейся и завершающейся на одном и том же состоянии.

У перехода существует несколько спецификаций. Они включают события, аргументы, ограждающие условия, действия и посылаемые события.

События

Событие (event) – это то, что вызывает переход из одного состояния в другое. События размещают на диаграмме вдоль линии перехода.

На диаграмме для отображения события можно использовать как имя операции, так и обычную фразу.

Большинство переходов должны иметь события, так как именно они, прежде всего, заставляют переход осуществиться. Тем не менее, бывают и автоматические переходы, не имеющие событий. При этом объект сам перемещается из одного состояния в другое со скоростью, позволяющей осуществиться входным действиям, деятельности и выходным действиям.

Ограждающие условия

Ограждающие условия (guard conditions) определяют, когда переход может, а когда не может осуществиться. В противном случае переход не осуществится.

Ограждающие условия изображают на диаграмме вдоль линии перехода после имени события, заключая их в квадратные скобки.

Ограждающие условия задавать необязательно. Однако если существует несколько автоматических переходов из состояния, необходимо определить для них взаимно исключающие ограждающие условия. Это поможет читателю диаграммы понять, какой путь перехода будет автоматически выбран.

Действие

Действием (action), как уже говорилось, является непрерываемое поведение, осуществляющееся как часть перехода. Входные и выходные действия показывают внутри состояний, поскольку они определяют, что происходит, когда объект входит или выходит из него.

Большую часть действий, однако, изображают вдоль линии перехода, так как они не должны осуществляться при входе или выходе из состояния.

Действие рисуют вдоль линии перехода после имени события, ему предшествует косая черта.

Событие или действие могут быть поведением внутри объекта, а могут представлять собой сообщение, посылаемое другому объекту. Если событие или действие посылается другому объекту, перед ним на диаграмме помещают знак « ^ ».



Рисунок 3.18 - Пример диаграммы состояний

Диаграммы состояний не надо создавать для каждого класса, они применяются только в сложных случаях. Если объект класса может существовать в нескольких состояниях и в каждом из них ведет себя по-разному, для него может потребоваться такая диаграмма.

Диаграммы размещения

Диаграмма размещения (deployment diagram) отражает физические взаимосвязи между программными и аппаратными компонентами системы. Она является хорошим средством для того, чтобы показать маршруты перемещения объектов и компонентов в распределенной системе.

Каждый узел на диаграмме размещения представляет собой некоторый тип вычислительного устройства – в большинстве случаев, часть аппаратуры. Эта аппаратура может быть простым устройством или датчиком, а может быть и мэйнфреймом.

Диаграмма размещения показывает физическое расположение сети и местонахождение в ней различных компонентов.

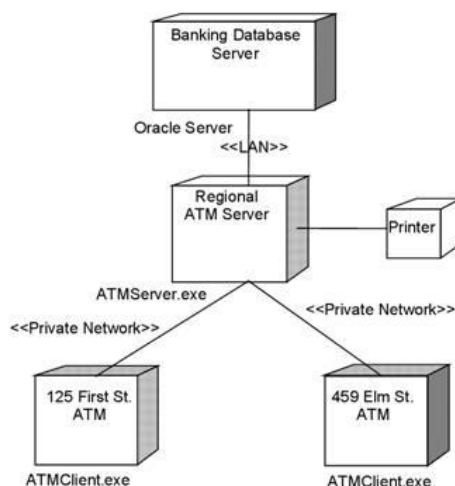


Рисунок 3.19 - Пример диаграммы размещения

Диаграмма размещения используется менеджером проекта, пользователями, архитектором системы и эксплуатационным персоналом, чтобы понять физическое размещение системы и расположение её отдельных подсистем.

Диаграммы компонентов

Диаграммы компонентов показывают, как выглядит модель на физическом уровне. На них изображены компоненты программного обеспечения и связи между ними. При этом на такой диаграмме выделяют два типа компонентов: исполняемые компоненты и библиотеки кода.

Каждый класс модели (или подсистема) преобразуется в компонент исходного кода. После создания они сразу добавляются к диаграмме компонентов. Между отдельными компонентами изображают зависимости, соответствующие зависимостям на этапе компиляции или выполнения программы.

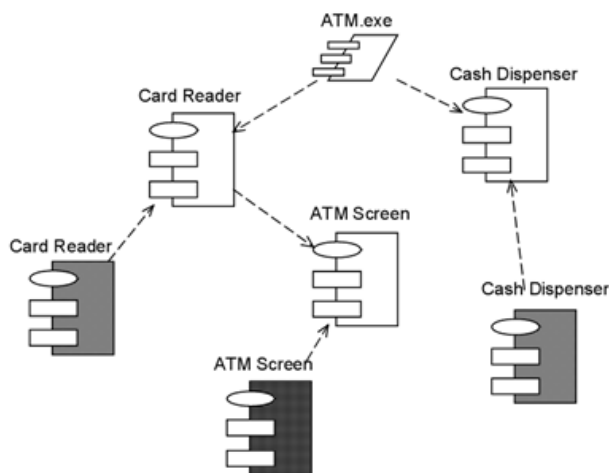


Рисунок 3.20 -. Пример диаграммы компонентов

Диаграммы компонентов применяются теми участниками проекта, кто отвечает за компиляцию системы. Из нее видно, в каком порядке надо компилировать компоненты, а также какие исполняемые компоненты будут созданы системой. На такой диаграмме показано соответствие классов реализованным компонентам. Она нужна там, где начинается генерация кода.

Объединение диаграмм компонентов и развертывания

В некоторых случаях допускается размещать диаграмму компонентов на диаграмме развертывания. Это позволяет показать какие компоненты выполняются и на каких узлах.

Работа с литературой:

Рекомендуемые источники информации (№ источника)			
Основная	Дополнительная	Методическая	Интернет-ресурсы
1-3	1-3	1-3	1-3

Оценочные средства: вопросы к собеседованию (См.: Фонд оценочных средств)

5. Типовые контрольные задания и иные материалы, характеризующие этапы формирования компетенций

5.1. Критерии оценивания компетенций

Оценка «отлично» выставляется студенту, если студент имеет глубокие знания, умения, навыки, демонстрирует полное понимание сущности рассматриваемых вопросов.

Оценка «хорошо» выставляется студенту, если студент имеет полные знания, умения, навыки, демонстрирует понимание сущности рассматриваемых вопросов в целом, но показывает недостаточно уверенное владение некоторыми теоретическими и практическими положениями дисциплины.

Оценка «удовлетворительно» выставляется студенту, если студент имеет низкий уровень знаний, умений, навыков, демонстрирует частичное понимание сущности рассматриваемых вопросов.

Оценка «неудовлетворительно» выставляется студенту, если студент имеет пробелы в знаниях, умениях, навыках, демонстрирует незнание сущности рассматриваемых вопросов.

5.2. Описание шкалы оценивания

Рейтинговая система оценки успеваемости студентов заочной формы не предусмотрена.

5.3. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Процедура проведения оценочных мероприятий включает в себя собеседование, подготовку к экзамену и выполнение индивидуальных заданий к лабораторным работам.

Предлагаемые студенту задания позволяют проверить компетенции ПК-6, ПК-13, ПК-14. Результаты экзамена и отчетов по индивидуальным заданиям оцениваются по:

- точности ответов на поставленные вопросы;
- последовательности и рациональности выполнения заданий;
- знанию технологий, использованных при выполнении задания.

6. Учебно-методическое и информационное обеспечение дисциплины

6.1 Перечень основной и дополнительной литературы, необходимой для освоения дисциплин

6.1.1. Перечень основной литературы

1. Основы компьютерного моделирование [Электронный ресурс] : учебно-методический комплекс / . — Электрон. текстовые данные. — Алматы: Нур-Принт, 2024.—175с. — 9965-756-09-0. — Режим доступа: <http://www.iprbookshop.ru/67115.html>.

2. Маглинец Ю.А. Анализ требований к автоматизированным информационным системам [Электронный ресурс]/ Маглинец Ю.А.—Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 191 с.— Режим доступа: <http://www.iprbookshop.ru/52184>.— ЭБС «IPRbooks», по паролю.

6.1.2. Перечень дополнительной литературы

1. Шорников Ю.В. Инструментальное моделирование гибридных систем [Электронный ресурс]: учебное пособие/ Шорников Ю.В., Томилов И.Н., Достовалов Д.Н.— Электрон.текстовые данные.— Новосибирск: Новосибирский государственный технический университет, 2014.— 70с.—Режим доступа: <http://www.iprbookshop.ru/44929>.— ЭБС IPRbooks..

2. Клименко И.С. Теория систем и системный анализ [Электронный ресурс]: учебное пособие / И.С. Клименко. — Электрон. текстовые данные. — М.: Российский новый университет, 2014. — 264 с. —Режим доступа: <http://www.iprbookshop.ru/21322.html>.

6.2. Перечень учебно-методического обеспечения самостоятельной работы обучающихся по дисциплине

1. Методические указания по выполнению лабораторных работ по дисциплине «Системный анализ данных и модели принятия решений».

2. Методические рекомендации для студентов по организации самостоятельной работы по дисциплине «Системный анализ данных и модели принятия решений».

6.3. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

1. Университетская библиотека online. <http://www.biblioclub.ru>.

2. ЭБС «IPRbooks». <http://www.iprbookshop.ru>.

3. Электронная библиотека СКФУ.. <http://catalog.ncstu.ru>.

4. Государственная публичная научно- техническая библиотека России. (ГПНТБ России). www.gpntb.ru.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал)

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ СТУДЕНТОВ К ВЫПОЛНЕНИЮ
САМОСТОЯТЕЛЬНОЙ РАБОТЫ
ПО ДИСЦИПЛИНЕ
СИСТЕМНЫЙ АНАЛИЗ ДАННЫХ И МОДЕЛИ ПРИНЯТИЯ РЕШЕНИЙ**

Направление подготовки	09.04.02
Направленность (профиль)	Информационные системы и технологии Технологии работы с данными и знаниями, анализ информации
Квалификация выпускника	Магистр

Пятигорск, 2026

СОДЕРЖАНИЕ

1. Цель и задачи освоения дисциплины	19
2. Место дисциплины в структуре основной образовательной программы	19
3 Технологическая карта самостоятельной работы обучающегося	19
4. Содержание самостоятельной работы.....	19
Тема самостоятельного изучения № 1. Методы и средства системного анализа данных	20
Тема самостоятельного изучения № 2. Анализ данных и проектирование систем	20
5. Теоретический материал	20
5.1 Математическое моделирование	20
5.2 Основы компьютерного моделирования.....	21
5.3 Жизненный цикл моделируемой системы	22
5.4 Моделирование как метод системного анализа	22
5.5 Компьютерная реализация математических вычислений	24
5.6 Построение диаграмм с применением вычислительной техники	27
5.7 Построение графика поверхности в системе Mathcad.	30
5.8 Применение диаграмм Байеса. Задачи выбора и классификации	33
5.9 Методы классификации и прогнозирования. Деревья решений.....	34
5.10 Построение классификационной модели.....	35
5.11 Преимущества деревьев решений.....	36
6. Типовые контрольные задания и иные материалы, характеризующие этапы формирования компетенций	38
7. Учебно-методическое и информационное обеспечение дисциплины	38

1. Цель и задачи освоения дисциплины

Целью освоения дисциплины является формирование набора профессиональных компетенций будущего магистра по направлению 09.04.02 Информационные системы и технологии для решения прикладных задач в рамках направленности (профиля) «Технологии работы с данными и знаниями, анализ информации».

Задачи освоения дисциплины: изучение основных понятий системного анализа данных и теории принятия решений, освоение методов и инструментов системного анализа данных и моделирования алгоритмов принятия решений.

2. Место дисциплины в структуре основной образовательной программы

Дисциплина «Системный анализ данных и модели принятия решений» относится к дисциплинам обязательной части. Ее освоение происходит в 4 семестре.

3 Технологическая карта самостоятельной работы обучающегося

Для студентов очной формы обучения:

Коды реализуемых компетенций, индикатора(ов)	Вид деятельности студентов	Средства и технологии оценки	Объем часов, в том числе		
			СРС	Контактная работа с преподавателем	Всего
2 семестр					
ПК-6, ПК-13, ПК-14	Подготовка к лекциям	Собеседование	1,08	0,12	1,2
ПК-6, ПК-13, ПК-14	Самостоятельное изучение литературы по темам 1, 2	Собеседование	71,28	7,92	79,2
ПК-6, ПК-13, ПК-14	Подготовка к лабораторным работам	Отчет письменный	3,24	0,36	3,6
			75,6	8,4	84

Для студентов заочной формы обучения:

Коды реализуемых компетенций, индикатора(ов)	Вид деятельности студентов	Средства и технологии оценки	Объем часов, в том числе		
			СРС	Контактная работа с преподавателем	Всего
2 семестр					
ПК-6, ПК-13, ПК-14	Подготовка к лекциям	Собеседование	0,36	0,04	0,4
ПК-6, ПК-13, ПК-14	Самостоятельное изучение литературы по темам 1, 2	Собеседование	88,56	9,84	98,4
ПК-6, ПК-13, ПК-14	Подготовка к лабораторным работам	Отчет письменный	1,08	0,12	1,2
			90	10	100

4. Содержание самостоятельной работы

Тема самостоятельного изучения № 1. Методы и средства системного анализа данных

Вид деятельности студентов: самостоятельное изучение литературы

Итоговый продукт самостоятельной работы: конспект

Средства и технологии оценки: собеседование

План конспекта:

Применение методов и средств анализа данных. Правила классификации. Деревья решений. Корреляционный и регрессионный анализ. Ассоциативные правила. Кластеризация. Типы алгоритмов.

Работа с литературой:

Рекомендуемые источники информации (№ источника)			
Основная	Дополнительная	Методическая	Интернет-ресурсы
1-3	1-3	1-3	1-3

Тема самостоятельного изучения № 2. Анализ данных и проектирование систем

Вид деятельности студентов: самостоятельное изучение литературы

Итоговый продукт самостоятельной работы: конспект

Средства и технологии оценки: собеседование

План конспекта:

Методы классификации. Методы Байеса. Наивный Байес. Вероятностный подход к наступлению состояния системы. Обучающая выборка. Оценочная функция Лапласа. Экспертная система. Система правил. Логический вывод. Семантическая модель. Продукционные, фреймовые и семантические сети. Интерпретатор. Применение баз знаний. Система авиалиний. ГИС-системы.

Работа с литературой:

Рекомендуемые источники информации (№ источника)			
Основная	Дополнительная	Методическая	Интернет-ресурсы
1-3	1-3	1-3	1-3

5. Теоретический материал

5.1 Математическое моделирование

Построение модели

Модель и моделирование - универсальные понятия, атрибуты одного из наиболее мощных методов познания в любой профессиональной области, познания системы, процесса, явления.

Вид модели и методы ее исследования больше зависят от информационно - логических связей элементов и подсистем моделируемой системы, ресурсов, связей с окружением, а не от конкретного наполнения системы.

Модельный стиль мышления позволяет вникать в структуру и внутреннюю логику моделируемой системы.

Построение модели - системная задача, требующая анализа и синтеза исходных данных, гипотез, теорий, знаний специалистов. Системный подход позволяет не только построить модель реальной системы, но и использовать эту модель для оценки (например, эффективности управления или функционирования) системы.

Модель - это объект или описание объекта, системы для замещения одной системы (оригинала) другой системой для лучшего изучения оригинала или воспроизведения каких-либо его свойств.

Например, отображая физическую систему на математическую систему, получим математическую модель физической системы. Любая модель строится и исследуется при определенных допущениях, гипотезах.

Пример. Рассмотрим физическую систему: тело массой m скатывается по наклонной плоскости с ускорением a , на которое воздействует сила F .

Исследуя такие системы, Ньютон получил математическое соотношение: $F = m \cdot a$. Это физико-математическая модель системы или математическая модель физической системы скатывающегося тела.

При описании этой системы приняты следующие гипотезы:

- поверхность идеальна (коэффициент трения равен нулю);
- тело находится в вакууме (сопротивление воздуха равно нулю);
- масса тела неизменна;
- тело движется с одинаковым постоянным ускорением в любой точке.

Пример. Физиологическая система (система кровообращения человека) - подчиняется некоторым законам термодинамики. Описывая эту систему на физическом (термодинамическом) языке балансовых законов, получим физическую, термодинамическую модель физиологической системы. Если записать эти законы на математическом языке, т.е. соответствующие термодинамические уравнения, то уже получаем математическую модель системы кровообращения.

Пример. Совокупность предприятий функционирует на рынке, обмениваясь товарами, сырьем, услугами, информацией. Если описать экономические законы, правила их взаимодействия на рынке с помощью математических соотношений, например, системы алгебраических уравнений, где неизвестными будут величины прибыли, получаемые от взаимодействия предприятий, а коэффициентами уравнения будут значения интенсивностей таких взаимодействий, то получим экономико-математическую модель системы предприятий на рынке.

Слово "модель" (лат. *modelium*) означает "мера", "способ", "сходство с какой-то вещью".

Моделирование базируется на математической теории подобия, согласно которой абсолютное подобие может иметь место лишь при замене одного объекта другим точно таким же.

При моделировании большинства систем абсолютное подобие невозможно, и основная цель моделирования заключается в том, что модель достаточно хорошо должна отображать функционирование моделируемой системы.

Пример фрактальной модели - множество Кантора. Рассмотрим отрезок $[0;1]$. Разделим его на 3 части и выбросим средний отрезок. Оставшиеся 2 промежутка опять разделим на три части и выкинем средние промежутки и т.д. Получим множество, называемое множеством Кантора. В пределе получаем несчетное множество изолированных точек.



Рисунок 1.1 - Множество Кантора для 3-х делений

5.2 Основы компьютерного моделирования

Границы между моделями различного вида весьма условны. Можно говорить о различных режимах использования моделей - имитационном, стохастическом, динамическом, детерминированном и др.

Как правило, модель включает в себя: объект О, субъект А (не обязательно) , задачу Z, ресурсы В, среду моделирования С.

Модель можно представить формально в виде: $M = \langle O, A, Z, B, C \rangle$.

Основные свойства любой модели:

1. Целенаправленность - модель всегда отображает некоторую систему, т.е. имеет цель такого отображения;
2. Конечность - модель отображает оригинал лишь в конечном числе его отношений и ресурсы моделирования конечны;
3. Упрощенность - модель отображает только существенные стороны объекта и она должна быть проста для исследования или воспроизведения;
4. Наглядность, обозримость основных ее свойств и отношений;
5. Доступность и технологичность для исследования или воспроизведения;
6. Информативность - модель должна содержать достаточную информацию о системе (в рамках гипотез, принятых при построении модели) и должна давать возможность получать новую информацию;
7. Полнота - в модели должны быть учтены все основные связи и отношения, необходимые для обеспечения цели моделирования;
8. Управляемость - модель должна иметь хотя бы один параметр, изменениями которого можно имитировать поведение моделируемой системы в различных условиях.

5.3 Жизненный цикл моделируемой системы

Жизненный цикл моделируемой системы состоит из следующих этапов:

1. Сбор информации об объекте, выдвижение гипотез, предварительный модельный анализ;
2. Проектирование структуры и состава моделей (подмоделей);
3. Построение спецификаций модели, разработка и отладка отдельных подмоделей, сборка модели в целом, идентификация (если это нужно) параметров моделей;
4. Исследование модели - выбор метода исследования и разработка алгоритма (программы) моделирования;
5. Исследование адекватности, устойчивости, чувствительности модели;
6. Оценка средств моделирования (затраченных ресурсов);
7. Интерпретация, анализ результатов моделирования и установление некоторых причинно-следственных связей в исследуемой системе;
8. Генерация отчетов и проектных (народно-хозяйственных) решений;
9. Уточнение, модификация модели, если это необходимо, и возврат к исследуемой системе с новыми знаниями, полученными с помощью модели и моделирования.

5.4 Моделирование как метод системного анализа

Часто в системном анализе при модельном подходе исследования может совершаться одна методическая ошибка, а именно, - построение корректных и адекватных моделей (подмоделей) подсистем системы и их логически корректная увязка не дает гарантий корректности построенной таким способом модели всей системы.

Модель, построенная без учета связей системы со средой, может служить подтверждением теоремы Геделя, а точнее, ее следствия, утверждающего, что в сложной изолированной системе могут существовать истины и выводы, корректные в этой системе и некорректные вне ее.

Наука моделирования состоит в разделении процесса моделирования (системы, модели) на этапы (подсистемы, подмодели), детальном изучении каждого этапа, взаимоотношений, связей, отношений между ними и затем эффективного описания их с максимально возможной степенью формализации и адекватности.

В случае нарушения этих правил получаем не модель системы, а модель "собственных и неполных знаний".

Моделирование рассматривается, как особая форма эксперимента, эксперимента не над самим оригиналом, т.е. простым или обычным экспериментом, а над копией оригинала. Здесь важен изоморфизм систем оригинальной и модельной.

Изоморфизм - равенство, одинаковость, подобие.

Модели и моделирование применяются по основным направлениям:

- в обучении, в познании и разработке теории исследуемых систем;
- в прогнозировании (выходных данных, ситуаций, состояний системы);
- в управлении (системой в целом, отдельными ее подсистемами);
- в автоматизации (системы или ее отдельных подсистем).

Классификация видов моделирования

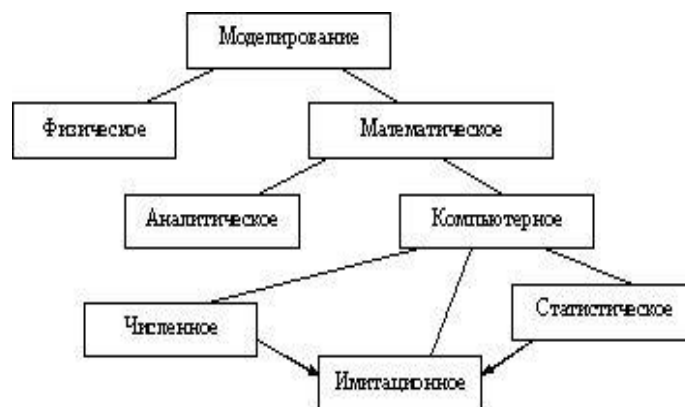


Рисунок 3.1 - Классификация видов моделирования

При физическом моделировании используется сама система, либо подобная ей в виде макета, например, летательный аппарат в аэродинамической трубе.

Математическое моделирование есть процесс установления соответствия реальной системе S математической модели M и исследование этой модели, позволяющее получить характеристики реальной системы.

При аналитическом моделировании процессы функционирования элементов записываются в виде математических соотношений (алгебраических, интегральных, дифференциальных, логических и др.).

Аналитическая модель может быть исследована методами:

- аналитическими (устанавливаются явные зависимости, получаются, в основном, аналитические решения);
- численными (получаются приближенные решения);

Компьютерное математическое моделирование формулируется в виде алгоритма (программы для ЭВМ), что позволяет проводить над моделью вычислительные эксперименты.

Численное моделирование использует методы вычислительной математики.

Статистическое моделирование использует обработку данных о системе с целью получения статистических характеристик системы.

Имитационное моделирование воспроизводит на ЭВМ (имитирует) процесс функционирования исследуемой системы, соблюдая логическую и временную последовательность протекания процессов, что позволяет узнать данные о состоянии системы или отдельных ее элементов в определенные моменты времени.

Применение математического моделирования позволяет исследовать объекты, реальные эксперименты над которыми затруднены или невозможны.

Экономический эффект при математическом моделировании состоит в том, что затраты на проектирование систем в среднем сокращаются в 50 раз.

Mathcad — система компьютерной алгебры

Mathcad — система компьютерной алгебры из класса систем автоматизированного проектирования, ориентированная на подготовку интерактивных документов с вычислениями и визуальным сопровождением, отличается легкостью использования и применения для коллективной работы.

Широкий интерес к системе Mathcad привел к тому, что в России наконец-то появились книги по отдельным версиям. Система Mathcad была создана в 80-х годах в университете Станфорда (США).

Современные версии для ПК готовит фирма MathSoft Application. Достоинство — программирование на языке математики. Это универсальный математический интерфейс:

- Есть мощная поддержка графики.
- Возможен импорт графики из других программ.
- Большое количество встроенных математических функций (сотни).
- Встроенные справочники по предметным областям.
- Возможна анимация.
- Символьная математика.

Недостатки:

- Это интерпретатор.
- Возможности программирования ограничены.

Системы Mathcad пользуются огромной популярностью во всем мире благодаря удобным средствам подготовки документов, имеющих вид обычных статей или книг. В то же время оказывается, что эти средства вполне достаточны для решения подавляющего большинства задач по математике, физике и других направлений науки и техники.

Mathcad является математически ориентированными универсальными системами. Помимо собственно вычислений они позволяют решать оформительские задачи. Они позволяют готовить статьи, книги, диссертации, научные отчеты, дипломные и курсовые проекты с доступным набором самых сложных математических формул и изысканным графическим представлением результатов.

С самого первого своего появления системы класса Mathcad имели удобный пользовательский интерфейс — совокупность средств общения с пользователем в виде масштабируемых и перемещаемых окон, кнопок и иных элементов. У этой системы есть эффективные средства типовой научной графики, они просты в применении и интуитивно понятны. Словом, системы Mathcad ориентированы на массового пользователя — от ученика начальных классов до академика.

Несмотря на то, что эта программа в основном ориентирована на пользователей-непрограммистов, Mathcad также используется в сложных проектах, чтобы визуализировать результаты математического моделирования, путем использования распределённых вычислений и традиционных языков программирования. Также Mathcad часто используется в крупных инженерных проектах, где большое значение имеет трассируемость и соответствие стандартам.

Mathcad достаточно удобно использовать для обучения, вычислений и инженерных расчетов. Открытая архитектура приложения в сочетании с поддержкой технологий NET и XML позволяют легко интегрировать приложение.

Рассмотрим наиболее часто применяемые функции для статистических расчетов, которые встроены в программу.

5.5 Компьютерная реализация математических вычислений

Математическое моделирование сложных систем

Будем считать, что элемент s есть некоторый объект, обладающий определенными свойствами, внутреннее строение которого для целей исследования не играет роли, например, самолет для моделирования полета — не элемент, а для моделирования работы аэропорта — элемент.

Связь l между элементами есть процесс их взаимодействия, важный для целей исследования.

Система S – совокупность элементов со связями l и целью функционирования системы F .

Сложная система – это система, состоящая из разнотипных элементов с разнотипными связями.

Большая система – это система, состоящая из большого числа однотипных элементов с однотипными связями.

В общем виде систему математически можно представить в виде:

$$S = \{\{s\}, \{l\}, F\}$$

Автоматизированная система S_A есть сложная система с определяющей ролью элементов двух типов: технических средств s_T и действий человека s_H :

$$S_A = \{\{s_T\}, \{s_H\}, \{s_O\}, \{l\}, F\}$$

Здесь s_O – остальные элементы системы.

Декомпозиция системы. Характеристики системы

Декомпозиция системы есть разбиение системы на элементы или группы элементов с указанием связей между ними, неизменными во время функционирования системы.

Практически все системы рассматриваются функционирующими во времени, поэтому определим их динамические характеристики.

Состояние – это множество характеристик элементов системы, изменяющихся во времени и важных для целей ее функционирования.

Процесс (динамика) – это множество значений состояний системы, изменяющихся во времени.

Цель функционирования есть задача получения желаемого состояния системы. Достижение цели обычно влечет целенаправленное вмешательство в процесс функционирования системы, которое называется управлением.

1. Расчет – определение значений параметров системы.
2. Анализ – изучение свойств функционирования системы.
3. Синтез – выбор структуры и параметров по заданным свойствам системы.

Параметры системы. Фазовая траектория. Фазовое пространство

Пусть $T = [t_0, t_1]$ есть временной интервал моделирования системы S (интервал модельного времени).

Построение модели начинается с определения параметров и переменных, определяющих процесс функционирования системы.

Параметры системы $\theta_1, \theta_2, \dots, \theta_m$ – это характеристики системы, остающиеся постоянными на всем интервале T .

Переменные бывают зависимые и независимые.

Независимые переменные есть, как правило, входные воздействия (в том числе управляющие)

$$u = u(t) = (u_1(t), \dots, u_n(t)) \in U \subseteq R^n$$

ими могут быть также воздействия внешней среды.

Последовательность изменения $x(t)$ при

$$t_1 < t_2 < \dots < t_N$$

называется фазовой траекторией системы,

$x \in X$, где X – пространство состояний или фазовое пространство.

Последовательность изменения $y(t)$ называется выходной траекторией системы.

Зависимые переменные есть выходные характеристики (сигналы)

$$y = y(t) = (y_1(t), \dots, y_n(t)) \in Y \subseteq R^n$$

Математическая модель системы

Общая схема математической модели (ММ) функционирования системы может быть представлена в виде, показанном на рисунке 4.1.

Множество переменных $\{u, v, \theta, x, y\}$ вместе с законами функционирования

$$x(t) = \dots,$$

$$y(t) = \dots$$

называется математической моделью системы.

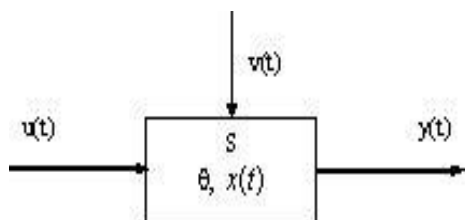


Рисунок 4.1 – Математическая модель системы

Если t непрерывно, то модель называется непрерывной, иначе – дискретной:

$$(t = i \cdot \Delta, i = 1, 2, \dots)$$

Если модель не содержит случайных элементов, то она называется детерминированной, в противном случае – вероятностной, стохастической.

Если математическое описание модели слишком сложное и частично или полностью неопределенно, то в этом случае используются агрегативные модели.

Сущность агрегативной модели заключается в разбиении системы на конечное число взаимосвязанных частей (подсистем), каждая из которых допускает стандартное математическое описание. Эти подсистемы называются агрегатами.

Моделирование случайных элементов в системах является одной из самых базовых задач математического моделирования.

Типы базовых датчиков:

- физические (любой физический шум), в последнее время практически не используются, т.к. характеристики нестабильны и реализацию повторить нельзя;
- псевдослучайные датчики.

Требования к базовым датчикам:

Отрезок аperiodичности.

Равномерность.

Некоррелированность.

Функции регрессии.

Широко распространенной задачей обработки данных является представление их совокупности некоторой функцией $y(x)$. Такое представление называется регрессией. Задача регрессии в том чтобы получить параметры функции такими, при которых функция приближает "облако" исходных точек с наименьшей квадратичной погрешностью.

Известны:

- Линейная регрессия (прямая линия);
- Полиномиальная регрессия (полином);
- Линейная регрессия общего вида (линейная сумма произвольных функций);
- Нелинейная регрессия общего вида (произвольная функция);
- Линейная регрессия.

При линейной регрессии функция $y(x)$ имеет вид $y(x) = ax + b$ и описывает отрезок прямой.

Функция предсказания

На практике нередко приходится сталкиваться с задачей расчёта последующих точек по ряду известных точек – задачей предсказания. Для предсказания поведения функциональной зависимости в Mathcad имеется функция: $\text{predict}(\text{data}, k, N)$ – предсказание, где data – вектор данных, k – число точек предшествующих предсказанию, N – число предсказываемых точек данных.

Функция предсказания обеспечивает высокую точность при монотонных исходных функциях или функциях, представляемых полиномом невысокой степени.

Функция predict применима к предсказуемым событиям, поведение которых описывается реальной математической зависимостью. В странах со стабильной экономикой эта функция вполне применима для описания сезонных и стабильных колебаний курса валют, прибылей фирм, продажа товаров и т.д.

Пример построения линейной регрессии:

$$VX := \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix} \quad VY := \begin{pmatrix} 4 \\ 11 \\ 15 \\ 21 \\ 28 \end{pmatrix} \quad \text{Исходные данные } ORIGIN := 1$$

Вычислим коэффициенты a и b линейной регрессии:

$$a := \text{intercept}(VX, VY) \quad b := \text{slope}(VX, VY) \quad a := 1.6 \quad f(x) := a + b \cdot x = -1.6 \quad b = 5.8$$

$$\text{corr}(VX, VY) = 0.996 \quad f(3) = 15.8 \quad \text{linterp}(VX, VY, 3) = 15$$

По этим данным строим график линейной регрессии:

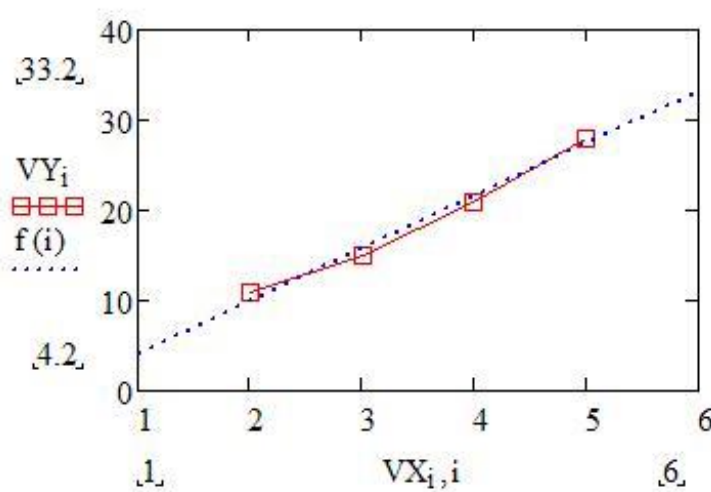


Рисунок 4.2 -Линейная регрессия

5.6 Построение диаграмм с применением вычислительной техники

Понятие о статических графиках. Основные элементы графика

Полученный в результате статистического исследования материал нередко изображается с помощью точек, геометрических линий и фигур или географических картосхем, т.е. графиков.

В статистике графиком называют наглядное изображение статистических величин и их соотношений при помощи геометрических точек, линий, фигур или географических картосхем.

Представление данных в графическом виде позволяет решать самые разнообразные задачи. Основное достоинство такого представления — наглядность. На графиках легко

просматривается тенденция к изменению. Можно даже определять скорость изменения тенденции. Различные соотношения, прирост, взаимосвязь различных процессов — все это легко можно увидеть на графиках.

Графики придают изложению статистических данных большую наглядность, чем таблицы, выразительность, облегчают их восприятие и анализ. Статистический график позволяет зрительно оценить характер изучаемого явления, присущие ему закономерности, тенденции развития, взаимосвязи с другими показателями, географическое разрешение изучаемых явлений. Еще в древности китайцы говорили, что одно изображение заменяет тысячу слов. Графики делают статистический материал более понятным, доступным и неспециалистам, привлекают внимание широкой аудитории к статистическим данным, популяризируют статистику и статистическую информацию.

При любой возможности анализ статистических данных рекомендуется всегда начинать с их графического изображения. График позволяет сразу получить общее представление обо всей совокупности статистических показателей. Графический метод анализа выступает как логическое продолжение табличного метода и служит целям получения обобщающих статистических характеристик процессов, свойственных массовым явлениям.

При помощи графического изображения статистических данных решаются многие задачи статистического исследования:

- наглядное представление величины показателей (явлений) в сравнении друг с другом;
- характеристика структуры какого-либо явления;
- изменение явления во времени;
- ход выполнения плана;
- зависимость изменения одного явления от изменения другого;
- распространенность или размещение каких-либо величин по территории.

Другими словами, в статистических исследованиях применяются самые разнообразные графики. В каждом графике выделяют (различают) следующие основные элементы:

- пространственные ориентиры (систему координат);
- графический образ;
- поле графика;
- масштабные ориентиры;
- экспликация графика;
- наименование графика

Иногда п.5 и п.6 объединяют в один элемент.

А) Пространственные ориентиры задаются в виде системы координатных сеток. В статистических графиках чаще всего применяется система прямоугольных координат. Иногда используется принцип полярных (угловых) координат (круговые графики). В картограммах средствами пространственной ориентации являются границы государств, границы административных его частей, географические ориентиры (контуры рек, береговых линий морей и океанов). На осях системы координат или на карте в определенном порядке располагаются характеристики статистических признаков изображаемых явлений или процессов. Признаки, располагаемые на осях координат, могут быть качественными или количественными.

Б) Графический образ статистических данных представляет собой совокупность линий, фигур, точек, образующих геометрические фигуры разной формы (окружность, квадраты, прямоугольники и т.п.) с различной штриховкой, окраской, густотой нанесения точек.

Любое явление, изучаемое статистикой, можно представить в графической форме. Для этого требуется найти правильное графическое решение, определить тот графический образ, который лучше всего соответствует данному явлению, нагляднее изображает

статистические данные. Графический образ должен соответствовать цели графика. Поэтому перед построением графика необходимо уяснить сущность явления и цель, которая ставится перед графическим изображением. Выбранная форма графика должна соответствовать внутреннему содержанию и характеру статистического показателя. Например, сравнение на графике производится по таким измерениям, как площадь, длина одной из сторон фигур, местонахождением точек, их густотой и т.д.

Так, для изображения изменений явления во времени наиболее естественным типом графика является линия. Для рядов распределения – полигон или гистограмма.

Каждый из основных видов графических изображений в статистической практике строится с учетом определенных правил. В статистических исследованиях для выяснения характерных черт и особенностей массовых явлений, познания типичного в этих явлениях и решения других задач широко используется сравнение одних абсолютных, средних и относительных статистических величин с другими. Анализ – это, прежде всего сравнение и сопоставление статистических данных. Нередко возникает необходимость сопоставления результатов статистического исследования конкретного явления с величинами типичного (идеального) явления аналогичной природы. Поэтому наглядное представление (графическое изображение) сравнения статистических показателей относится к наиболее распространенным графикам в статистике. Для этих целей применяются диаграммы.

Диаграмма – это графическое изображение, наглядно показывающее соотношение между сравниваемыми величинами. Диаграмма представляет собой чертеж, на котором статистические данные условно изображаются геометрическими линиями, фигурами и телами различных размеров.

Различают следующие основные виды графиков (диаграмм) сравнения:

- столбиковые;
- полосовые;
- квадратные;
- круговые;
- фигурные.

Для построения диаграмм используются следующие компьютерные программы Excel, Mathcad, MATLAB и другие.

Для создания графиков в системе Mathcad имеется программный графический процессор. Основное внимание при его разработке было уделено обеспечению простоты задания графиков и их модификации с помощью соответствующих параметров. Процессор позволяет строить самые разные графики, например в декартовой и полярной системе координат, трехмерные поверхности, графики уровней.

Для построения графиков используются шаблоны. Их перечень содержит подменю Graph меню Insert. Большинство параметров графического процессора, необходимых для построения графиков, по умолчанию задается автоматически, поэтому для начального построения графика того или иного вида достаточно задать тип графика. В подменю Graph содержится список из семи основных типов графиков. Они позволяют выполнить следующие действия:

X – Y Plot (декартов график) – создать шаблон двухмерного графика в декартовой системе координат;

Polar Plot (полярный график) – создать шаблон графика в полярных координатах;

3D Plot Wizard (мастер трехмерных графиков) – вызов Мастера для построения трехмерных графиков с заданными свойствами;

Surface Plot (график поверхности) - создать шаблон для построения трехмерного графика;

Contour Plot (карта линий уровня) – создать шаблон для контурного графика трехмерной поверхности;

3D Scatter Plot (точечный график) – создать шаблон для графика в виде точек (фигур) в трехмерном пространстве;

3D Bar Plot (трехмерная столбиковая диаграмма) – создать шаблон для изображения в виде совокупности столбиков в трехмерном пространстве;

Vector Field Plot (векторное поле) – создать шаблон для графика векторного поля на плоскости.

С понятием графики связано представление о графических объектах, имеющих определенные свойства. В большинстве случаев об объектах можно забыть, если вы не занимаетесь объектно-ориентированным программированием. Большинство команд высокоуровневой графики ориентированы на конечного пользователя. Они автоматически устанавливают свойство графических объектов и обеспечивают воспроизведение графики в нужной системе координат, палитре цветов, масштабе и т.д.

5.7 Построение графика поверхности в системе Mathcad.

Построение поверхностей по матрице аппликат их точек. Поскольку элементы матрицы M – переменные с целочисленными индексами, то перед созданием матрицы требуется задать индексы в виде ранжированных переменных с целочисленными значениями, а затем уже из них формировать сетку значений x и y – координат для аппликат $z(x,y)$. Значения x и y при этом обычно должны быть вещественными числами, нередко как положительными, так и отрицательными. После выполнения указанных выше определений вводится шаблон графика (команда Surface Plot).

Построение Графика Поверхности

$z(x,y) := \cos(x \cdot y)$ Функция двух переменных x, y
 $i := 0..20$ $j := 0..20$ Целочисленные индексы

$$M_{i,j} := z\left[\frac{(i-10)}{5}, \frac{(j-10)}{5}\right]$$

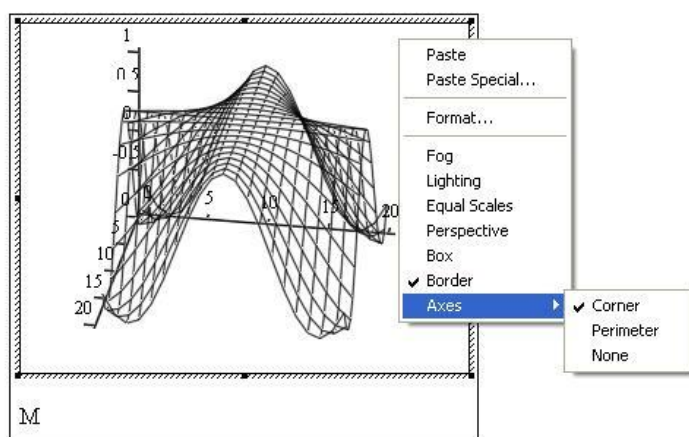


Рисунок 5.1 - Построение поверхности без удаления невидимых линий

Необходимо построить график поверхности в системе Mathcad, отформатировав его, применив алгоритм функциональной окраски и удаление невидимых линий. Показано, как отформатировать график, применение алгоритма функциональной окраски поверхности и удаление невидимых линий.

Построение Графика Поверхности

$z(x,y) := \cos(xy)$ Функция двух переменных x, y
 $i := 0..20$ $j := 0..20$ Целочисленные индексы

$$M_{i,j} := \left[\frac{(i-10)}{5}, \frac{(j-10)}{5} \right]$$

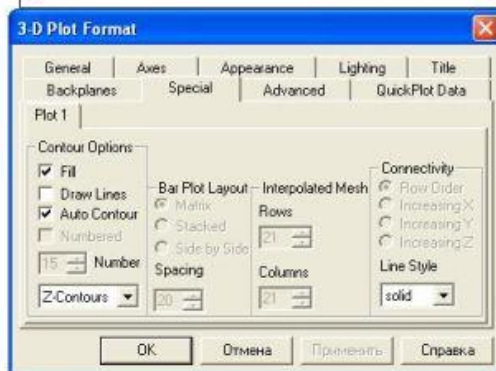
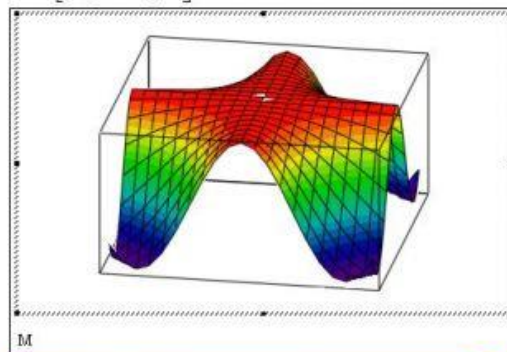


Рисунок 5.2 - Построение графика поверхности в системе Mathcad

Построение трехмерного графика в системе Mathcad

В программе Mathcad есть возможность построения трехмерных графиков – без задания матриц аппликат поверхностей. Единственным недостатком такого упрощенного метода построения поверхностей является неопределенность в масштабировании, поэтому графики требуют форматирования.

Построение графика поверхности без
задания матрицы

$z(x,y) := \cos(xy)$ Функция двух переменных x, y

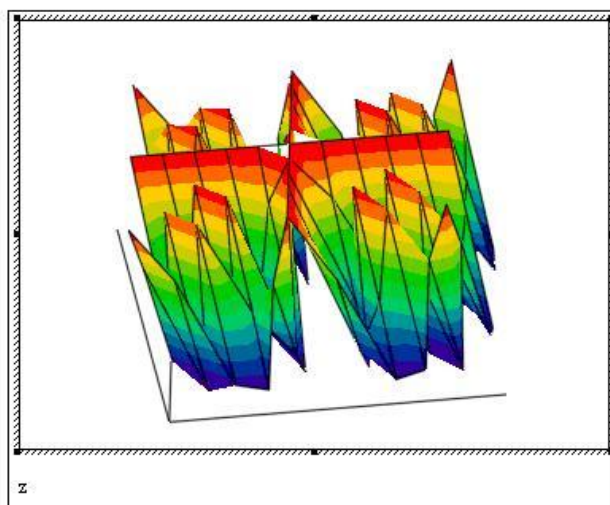


Рисунок 6.1 - Построение графика поверхности без задания матрицы

Применение графической функции CreateMesh

1. Зададим функцию двух переменных
2. Зададим пределы изменения переменных
3. Используя функцию CreateMesh создадим матрицу поверхности

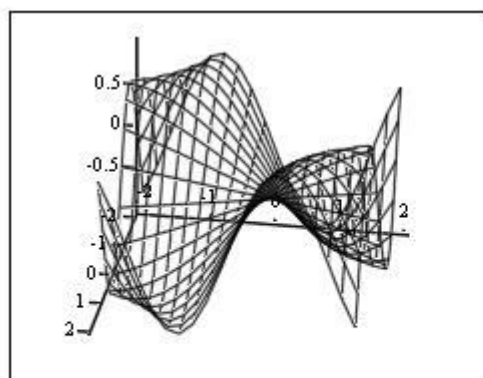
$$H(u, v) := \sin(u \cdot v)$$

$$u0 := -2 \quad u1 := 2$$

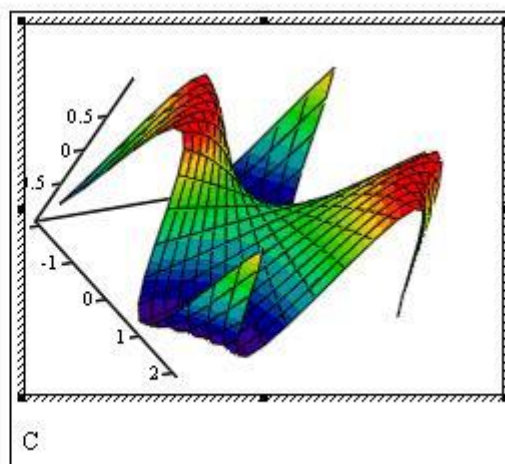
$$v0 := -2 \quad v1 := 2$$

$$C := \text{CreateMesh}(H, u0, u1, v0, v1)$$

4. Задав построение графика типа Surface Plot, получим



C



C

Рисунок 6.2 - Построение графика поверхности с применением функции CreateMesh

Еще один пример применения функции CreateMesh – построение объемной фигуры, которая получается вращением кривой, заданной функцией $f(x)$, вокруг оси X или Y. На рисунке 6.3 показан пример решения данной задачи.

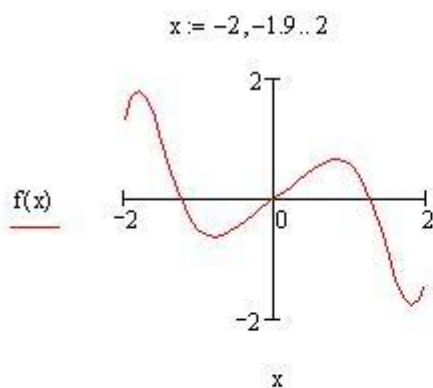
Построение фигуры, полученной вращением кривой вокруг оси X

$$f(x) := x \cdot \cos(x^2) \quad a := -2 \quad b := 2 \quad \text{mesh} := 30$$

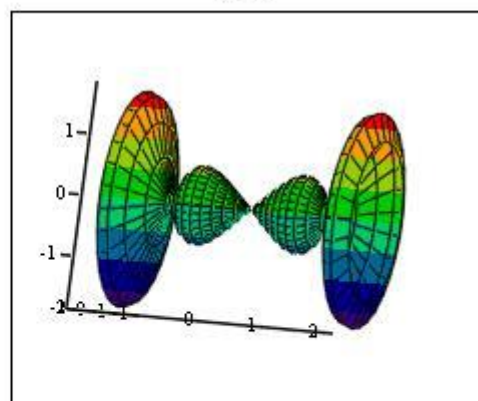
$$F(u, v) := u \quad G(u, v) := f(u) \cdot \cos(v) \quad H(u, v) := f(u) \cdot \sin(v)$$

$$S := \text{CreateMesh}(F, G, H, a, b, 0, 2\pi, \text{mesh})$$

График функции $f(x)$



Поверхность вращения графика
 $f(x)$



S

Рисунок 6.3 – Фигура, полученная вращением кривой.

5.8 Применение диаграмм Байеса. Задачи выбора и классификации

Байесовская классификация

Альтернативные названия: байесовское моделирование, байесовская статистика, метод байесовских сетей. Изначально байесовская классификация использовалась для формализации знаний экспертов в экспертных системах, сейчас байесовская классификация также применяется в качестве одного из методов Data Mining.

Так называемая наивная классификация или наивно-байесовский подход (naive-bayes approach) является наиболее простым вариантом метода, использующего байесовские сети. При этом подходе решаются задачи классификации, результатом работы метода являются так называемые "прозрачные" модели.

"Наивная" классификация - достаточно прозрачный и понятный метод классификации. "Наивной" она называется потому, что исходит из предположения о взаимной независимости признаков.

Свойства наивной классификации:

- Использование всех переменных и определение всех зависимостей между ними.
- Наличие двух предположений относительно переменных:
- все переменные являются одинаково важными;
- все переменные являются статистически независимыми, т.е. значение одной переменной ничего не говорит о значении другой.

Большинство других методов классификации предполагают, что перед началом классификации вероятность того, что объект принадлежит тому или иному классу, одинакова; но это не всегда верно.

Допустим, известно, что определенный процент данных принадлежит конкретному классу. Возникает вопрос, можем ли мы использовать эту информацию при построении модели классификации? Существует множество реальных примеров использования этих априорных знаний, помогающих классифицировать объекты. Типичный пример из медицинской практики. Если доктор отправляет результаты анализов пациента на дополнительное исследование, он относит пациента к какому-то определенному классу. Каким образом можно применить эту информацию? Мы можем использовать ее в качестве дополнительных данных при построении классификационной модели.

Отмечают такие достоинства байесовских сетей как метода Data Mining:

- в модели определяются зависимости между всеми переменными, это позволяет легко обрабатывать ситуации, в которых значения некоторых переменных неизвестны;
- байесовские сети достаточно просто интерпретируются и позволяют на этапе прогностического моделирования легко проводить анализ по сценарию "что, если";
- байесовский метод позволяет естественным образом совмещать закономерности, выведенные из данных, и, например, экспертные знания, полученные в явном виде;
- использование байесовских сетей позволяет избежать проблемы переучивания (overfitting), то есть избыточного усложнения модели, что является слабой стороной многих методов (например, деревьев решений и нейронных сетей).

Наивно-байесовский подход имеет следующие недостатки:

- перемножать условные вероятности корректно только тогда, когда все входные переменные действительно статистически независимы; хотя часто данный метод показывает достаточно хорошие результаты при несоблюдении условия статистической независимости, но теоретически такая ситуация должна обрабатываться более сложными методами, основанными на обучении байесовских сетей;
- невозможна непосредственная обработка непрерывных переменных - требуется их преобразование к интервальной шкале, чтобы атрибуты были дискретными; однако такие преобразования иногда могут приводить к потере значимых закономерностей;
- на результат классификации в наивно-байесовском подходе влияют только индивидуальные значения входных переменных, комбинированное влияние пар или троек значений разных атрибутов здесь не учитывается. Это могло бы улучшить качество

классификационной модели с точки зрения ее прогнозирующей точности, однако, увеличило бы количество проверяемых вариантов.

Байесовская классификация нашла широкое применение на практике.

Байесовская фильтрация по словам

Не так давно байесовская классификация была предложена для персональной фильтрации спама. Первый фильтр был разработан Полем Грахемом (Paul Graham). Для работы алгоритма требуется выполнение двух требований.

Первое требование - необходимо, чтобы у классифицируемого объекта присутствовало достаточное количество признаков. Этому идеально удовлетворяют все слова писем пользователя, за исключением совсем коротких и очень редко встречающихся.

Второе требование - постоянное переобучение и пополнение набора "спам - не спам". Такие условия очень хорошо работают в локальных почтовых клиентах, так как поток "не спама" у конечного клиента достаточно постоянен, а если изменяется, то не быстро.

Однако для всех клиентов сервера точно определить поток "не спама" довольно сложно, поскольку одно и то же письмо, являющееся для одного клиента спамом, для другого спамом не является. Словарь получается слишком большим, не существует четкого разделения на спам и "не спам", в результате качество классификации, в данном случае решение задачи фильтрации писем, значительно снижается.

5.9 Методы классификации и прогнозирования. Деревья решений

Метод деревьев решений (decision trees) является одним из наиболее популярных методов решения задач классификации и прогнозирования. Иногда этот метод Data Mining также называют деревьями решающих правил, деревьями классификации и регрессии. Как видно из последнего названия, при помощи данного метода решаются задачи классификации и прогнозирования.

Если зависимая, т.е. целевая переменная принимает дискретные значения, при помощи метода дерева решений решается задача классификации.

Если же зависимая переменная принимает непрерывные значения, то дерево решений устанавливает зависимость этой переменной от независимых переменных, т.е. решает задачу численного прогнозирования.

Впервые деревья решений были предложены Ховилендом и Хантом (Hoveland, Hunt) в конце 50-х годов прошлого века. Самая ранняя и известная работа Ханта и др., в которой излагается суть деревьев решений - "Эксперименты в индукции" ("Experiments in Induction") - была опубликована в 1966 году.

В наиболее простом виде дерево решений - это способ представления правил в иерархической, последовательной структуре. Основа такой структуры - ответы "Да" или "Нет" на ряд вопросов.

На рисунке 8.1 приведен пример дерева решений, задача которого - ответить на вопрос: "Играть ли в гольф?" Чтобы решить задачу, т.е. принять решение, играть ли в гольф, следует отнести текущую ситуацию к одному из известных классов (в данном случае - "играть" или "не играть"). Для этого требуется ответить на ряд вопросов, которые находятся в узлах этого дерева, начиная с его корня.

Первый узел нашего дерева "Солнечно?" является узлом проверки, т.е. условием. При положительном ответе на вопрос осуществляется переход к левой части дерева, называемой левой ветвью, при отрицательном - к правой части дерева. Таким образом, внутренний узел дерева является узлом проверки определенного условия. Далее идет следующий вопрос и т.д., пока не будет достигнут конечный узел дерева, являющийся узлом решения. Для нашего дерева существует два типа конечного узла: "играть" и "не играть" в гольф.

В результате прохождения от корня дерева (иногда называемого корневой вершиной) до его вершины решается задача классификации, т.е. выбирается один из классов - "играть" и "не играть" в гольф.

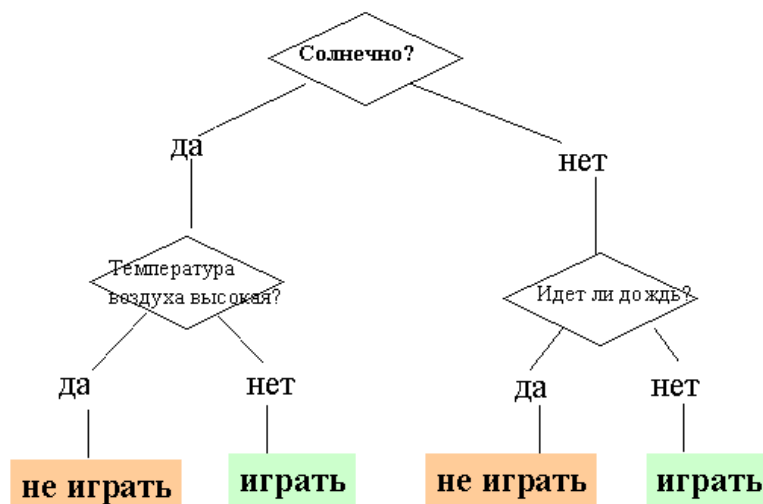


Рисунок 8.1 - Дерево решений "Играть ли в гольф?"

Целью построения дерева решения в нашем случае является определение значения категориальной зависимой переменной.

Итак, для нашей задачи основными элементами дерева решений являются:

- Корень дерева: "Солнечно?"
- Внутренний узел дерева или узел проверки: "Температура воздуха высокая?", "Идет ли дождь?"
- Лист, конечный узел дерева, узел решения или вершина: "Играть", "Не играть"
- Ветвь дерева (случаи ответа): "Да", "Нет".

В рассмотренном примере решается задача бинарной классификации, т.е. создается дихотомическая классификационная модель. Пример демонстрирует работу так называемых бинарных деревьев.

В узлах бинарных деревьев ветвление может вестись только в двух направлениях, т.е. существует возможность только двух ответов на поставленный вопрос ("да" и "нет").

5.10 Построение классификационной модели

Бинарные деревья являются самым простым, частным случаем деревьев решений. В остальных случаях, ответов и, соответственно, ветвей дерева, выходящих из его внутреннего узла, может быть больше двух.

Рассмотрим более сложный пример. База данных, на основе которой должно осуществляться прогнозирование, содержит следующие ретроспективные данные о клиентах банка, являющиеся ее атрибутами: возраст, наличие недвижимости, образование, среднемесячный доход, вернул ли клиент вовремя кредит. Задача состоит в том, чтобы на основании перечисленных выше данных (кроме последнего атрибута) определить, стоит ли выдавать кредит новому клиенту.

Такая задача решается в два этапа: построение классификационной модели и ее использование.

На этапе построения модели, собственно, и строится дерево классификации или создается набор неких правил. На этапе использования модели построенное дерево, или путь от его корня к одной из вершин, являющийся набором правил для конкретного клиента, используется для ответа на поставленный вопрос "Выдавать ли кредит?"

Правилom является логическая конструкция, представленная в виде "если.... то....".

На рисунке 9.2 приведен пример дерева классификации, с помощью которого решается задача "Выдавать ли кредит клиенту?". Она является типичной задачей классификации, и при помощи деревьев решений получают достаточно хорошие варианты ее решения.

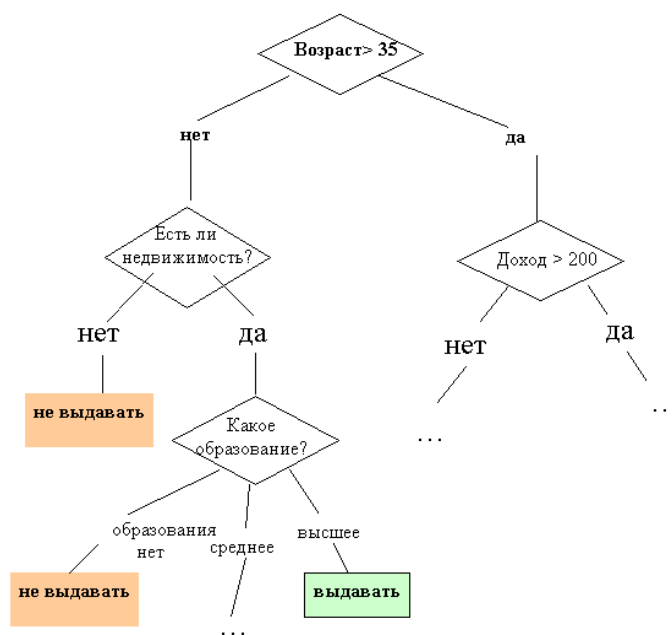


Рис. 9.2. Дерево решений "Выдавать ли кредит?"

Как мы видим, внутренние узлы дерева (возраст, наличие недвижимости, доход и образование) являются атрибутами описанной выше базы данных. Эти атрибуты называют прогнозирующими, или атрибутами расщепления (splitting attribute). Конечные узлы дерева, или листы, именуются метками класса, являющимися значениями зависимой категориальной переменной "выдавать" или "не выдавать" кредит.

Каждая ветвь дерева, идущая от внутреннего узла, отмечена предикатом расщепления. Последний может относиться лишь к одному атрибуту расщепления данного узла. Характерная особенность предикатов расщепления: каждая запись использует уникальный путь от корня дерева только к одному узлу-решению. Объединенная информация об атрибутах расщепления и предикатах расщепления в узле называется критерием расщепления (splitting criterion).

На рисунке 9.1 изображено одно из возможных деревьев решений для рассматриваемой базы данных. Например, критерий расщепления "Какое образование?", мог бы иметь два предиката расщепления и выглядеть иначе: образование "высшее" и "не высшее". Тогда дерево решений имело бы другой вид.

Таким образом, для данной задачи (как и для любой другой) может быть построено множество деревьев решений различного качества, с различной прогнозирующей точностью.

Качество построенного дерева решения весьма зависит от правильного выбора критерия расщепления. Над разработкой и усовершенствованием критериев работают многие исследователи.

Метод деревьев решений часто называют "наивным" подходом [34]. Но благодаря целому ряду преимуществ, данный метод является одним из наиболее популярных для решения задач классификации.

5.11 Преимущества деревьев решений

Интуитивность деревьев решений. Классификационная модель, представленная в виде дерева решений, является интуитивной и упрощает понимание решаемой задачи. Результат работы алгоритмов конструирования деревьев решений, в отличие, например, от нейронных сетей, представляющих собой "черные ящики", легко интерпретируется пользователем. Это свойство деревьев решений не только важно при отнесении к определенному классу нового объекта, но и полезно при интерпретации модели

классификации в целом. Дерево решений позволяет понять и объяснить, почему конкретный объект относится к тому или иному классу.

Деревья решений дают возможность извлекать правила из базы данных на естественном языке. Пример правила: Если Возраст > 35 и Доход > 200 , то выдать кредит.

Деревья решений позволяют создавать классификационные модели в тех областях, где аналитику достаточно сложно формализовать знания.

Алгоритм конструирования дерева решений не требует от пользователя выбора входных атрибутов (независимых переменных). На вход алгоритма можно подавать все существующие атрибуты, алгоритм сам выберет наиболее значимые среди них, и только они будут использованы для построения дерева. В сравнении, например, с нейронными сетями, это значительно облегчает пользователю работу, поскольку в нейронных сетях выбор количества входных атрибутов существенно влияет на время обучения.

Точность моделей, созданных при помощи деревьев решений, сопоставима с другими методами построения классификационных моделей (статистические методы, нейронные сети).

Разработан ряд масштабируемых алгоритмов, которые могут быть использованы для построения деревьев решения на сверхбольших базах данных; масштабируемость здесь означает, что с ростом числа примеров или записей базы данных время, затрачиваемое на обучение, т.е. построение деревьев решений, растет линейно. Примеры таких алгоритмов: SLIQ, SPRINT.

Быстрый процесс обучения. На построение классификационных моделей при помощи алгоритмов конструирования деревьев решений требуется значительно меньше времени, чем, например, на обучение нейронных сетей.

Большинство алгоритмов конструирования деревьев решений имеют возможность специальной обработки пропущенных значений.

Многие классические статистические методы, при помощи которых решаются задачи классификации, могут работать только с числовыми данными, в то время как деревья решений работают и с числовыми, и с категориальными типами данных.

Многие статистические методы являются параметрическими, и пользователь должен заранее владеть определенной информацией, например, знать вид модели, иметь гипотезу о виде зависимости между переменными, предполагать, какой вид распределения имеют данные. Деревья решений, в отличие от таких методов, строят непараметрические модели. Таким образом, деревья решений способны решать такие задачи Data Mining, в которых отсутствует априорная информация о виде зависимости между исследуемыми данными.

Процесс конструирования дерева решений

Напомним, что рассматриваемая нами задача классификации относится к стратегии обучения с учителем, иногда называемого индуктивным обучением. В этих случаях все объекты тренировочного набора данных заранее отнесены к одному из предопределенных классов.

Алгоритмы конструирования деревьев решений состоят из этапов "построение" или "создание" дерева (tree building) и "сокращение" дерева (tree pruning). В ходе создания дерева решаются вопросы выбора критерия расщепления и остановки обучения (если это предусмотрено алгоритмом). В ходе этапа сокращения дерева решается вопрос отсечения некоторых его ветвей.

Критерий расщепления

Процесс создания дерева происходит сверху вниз, т.е. является нисходящим. В ходе процесса алгоритм должен найти такой критерий расщепления, иногда также называемый критерием разбиения, чтобы разбить множество на подмножества, которые бы ассоциировались с данным узлом проверки. Каждый узел проверки должен быть помечен определенным атрибутом. Существует правило выбора атрибута: он должен разбивать исходное множество данных таким образом, чтобы объекты подмножеств, получаемых в результате этого разбиения, являлись представителями одного класса или же были

максимально приближены к такому разбиению. Последняя фраза означает, что количество объектов из других классов, так называемых "примесей", в каждом классе должно стремиться к минимуму.

Существуют различные критерии расщепления. Наиболее известные - мера энтропии и индекс Gini.

В некоторых методах для выбора атрибута расщепления используется так называемая мера информативности подпространств атрибутов, которая основывается на энтропийном подходе и известна под названием "мера информационного выигрыша" (information gain measure) или мера энтропии.

6. Типовые контрольные задания и иные материалы, характеризующие этапы формирования компетенций

6.1. Критерии оценивания компетенций

Оценка «отлично» выставляется студенту, если студент имеет глубокие знания, умения, навыки, демонстрирует полное понимание сущности рассматриваемых вопросов.

Оценка «хорошо» выставляется студенту, если студент имеет полные знания, умения, навыки, демонстрирует понимание сущности рассматриваемых вопросов в целом, но показывает недостаточно уверенное владение некоторыми теоретическими и практическими положениями дисциплины.

Оценка «удовлетворительно» выставляется студенту, если студент имеет низкий уровень знаний, умений, навыков, демонстрирует частичное понимание сущности рассматриваемых вопросов.

Оценка «неудовлетворительно» выставляется студенту, если студент имеет пробелы в знаниях, умениях, навыках, демонстрирует незнание сущности рассматриваемых вопросов.

6.2. Описание шкалы оценивания

Рейтинговая система оценки успеваемости студентов заочной формы не предусмотрена.

6.3. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Процедура проведения оценочных мероприятий включает в себя собеседование, подготовку к экзамену и выполнение индивидуальных заданий к лабораторным работам.

Предлагаемые студенту задания позволяют проверить компетенции ПК-6, ПК-13, ПК-14. Результаты экзамена и отчетов по индивидуальным заданиям оцениваются по:

- точности ответов на поставленные вопросы;
- последовательности и рациональности выполнения заданий;
- знанию технологий, использованных при выполнении задания.

7. Учебно-методическое и информационное обеспечение дисциплины

7.1 Перечень основной и дополнительной литературы, необходимой для освоения дисциплин

7.1.1. Перечень основной литературы

1. Основы компьютерного моделирование [Электронный ресурс] : учебно-методический комплекс / . — Электрон. текстовые данные. — Алматы: Нур-Принт, 2024.—175с. — 9965-756-09-0. — Режим доступа: <http://www.iprbookshop.ru/67115.html>.

2. Маглинец Ю.А. Анализ требований к автоматизированным информационным системам [Электронный ресурс]/ Маглинец Ю.А.—Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 191 с.— Режим доступа: <http://www.iprbookshop.ru/52184>.— ЭБС «IPRbooks», по паролю.

7.1.2. Перечень дополнительной литературы

1. Шорников Ю.В. Инструментальное моделирование гибридных систем [Электронный ресурс]: учебное пособие/ Шорников Ю.В., Томилов И.Н., Достовалов Д.Н.— Электрон.текстовые данные.— Новосибирск: Новосибирский государственный

технический университет, 2014.— 70с.—Режим доступа: <http://www.iprbookshop.ru/44929>.— ЭБС IPRbooks..

2. Клименко И.С. Теория систем и системный анализ [Электронный ресурс]: учебное пособие / И.С. Клименко. — Электрон. текстовые данные. — М.: Российский новый университет, 2014. — 264 с. —Режим доступа: <http://www.iprbookshop.ru/21322.html>.

7.2. Перечень учебно-методического обеспечения самостоятельной работы обучающихся по дисциплине

1. Методические указания по выполнению лабораторных работ по дисциплине «Системный анализ данных и модели принятия решений».

2. Методические рекомендации для студентов по организации самостоятельной работы по дисциплине «Системный анализ данных и модели принятия решений».

7.3. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

1. Университетская библиотека online. <http://www.biblioclub.ru>.

2. ЭБС «IPRbooks». <http://www.iprbookshop.ru>.

3. Электронная библиотека СКФУ.. <http://catalog.ncstu.ru>.

4. Государственная публичная научно-техническая библиотека России. (ГПНТБ России). www.gpntb.ru.