Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ»
(ТУСУР)

Кафедра моделирования и системного анализа

Панов С.А.

ТЕОРИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

Курс лекций

Томск
2015

Панов С.А. Теория информационных систем / Курс лекций – Томск: Томский государственный университет систем управления и радиоэлектроники, Кафедра моделирования и системного анализа, 2015. – 87 с.

СОДЕРЖАНИЕ

Введение

Для развития современного эффективного образовательного процесса на всех ступенях и во всех его формах, а также в целях повышения эффективности научных исследований необходимо обеспечить информационно-аналитическую поддержку как образовательных технологий, так и научных исследований.

Характерно, что образование, творчество, научные исследования как средства постижения истины и всего нового тесно переплетаются, имеют схожие семантические очертания, а потому могут поддерживаться весьма схожими информационными системами (ИС). Поэтому проблемы проектирования и моделирования для них вполне уместно рассматривать комплексно в едином контексте, что и предпринимается в данной попытке формирования вспомогательного учебного пособия, составленного по открытым материалам сети Интернет в интересах информационно-методической поддержки дисциплины «Проектирование ИС» по специальностям «Информационные системы», «Информационные технологии в образовании», «Информационные системы в научных исследованиях».

При этом в научных исследованиях и в наиболее развитых в творческом отношении частях образовательной деятельности высшей школы (диссертационные работы, дипломные, курсовые проекты, НИРС и т.п.) возникает явная потребность в создании и использовании проблемно-ориентированных систем поддержки принятия решений с применением новейших информационных технологий, методов компьютерного моделирования и интеллектуального анализа данных. Методы компьютерного моделирования позволяют проводить комплексный анализ сложнейших систем и проектов в сфере образовательной деятельности, научной работы и управленческой деятельности учреждений образования и науки с учетом множества факторов и элементов неопределенности, прогнозировать будущее состояние научных и образовательных проблем, выявлять скрытые взаимосвязи, анализировать аномалии, что в конечном итоге позволяет принимать обоснованные решения как по совершенствованию и оптимизации научных и образовательных технологий, так и по реализации великого множества частных конкретных проектных задач, стоящих перед исследователями и учащимися.

Методы компьютерного моделирования позволяют создавать и эксплуатировать многофункциональные высокоэффективные информационные системы, поддерживающие научные исследования и образовательные технологии.

Такие комплексные информационные системы (ИС), в свою очередь включающие в свой состав локальные ИС и АСУ, решающие отдельные частные информационные и управленческие задачи, опираясь на возможности современных телекоммуникаций и корпоративное обустройство информационного обмена, позволяют создать в учреждениях образования и науки, а также на письменном столе каждого отдельно взятого ученого или учащегося развитую информационно-управленческую среду (информсреду), интегрированную в мировое информационное Интернет-пространство.

Средства и наполнение комплексных ИС и АСУ, поддерживающие информсреду образования и науки, включают следующие основополагающие компоненты:

- *многоуровневая телекоммуникационная среда* обмена данными через технологические центры коммутации и средства комплексирования разнородных данных на основе общесистемных стандартов и классификаторов, соглашений по протоколам представления и обмена информацией с обеспечением благодаря

телекоммуникационной среде выхода на региональные и межрегиональные базы данных (БД), включая международные информационные сети (прежде всего Интернет);

- *геоинформационные системы* сбора, представления и обработки информации об объектах науки, образования, культуры и т.п.;
- *системы информационных технологий и автоматизированных банков данных (БД)* образования и науки на государственном отраслевом\межотраслевом, региональном, ведомственном, учрежденческом уровнях во взаимодействии с информсредой великого множества конечных пользователей – подразделений и отдельных участников образовательных и научных процессов;
- *банки общественно-полезных и профессионально-значимых данных*, которые обеспечивают автоматизированную обработку документопотоков, автоматизированную многоаспектную обработку исходных данных по вопросам управления учреждениями науки и образования, планирования деятельности отдельных участников образовательных и научных технологий, оптимального проектирования, статучета, поддержания информационно-социальных технологий и функций и т.п.;
- *системы информационно-аналитического обслуживания* для принятия решений;

Некоторые вводные понятия и определения, используемые в настоящем УМК

(по публикациям академика Э.Якубайтиса, М.Броя и других известных авторов в сфере информатики и НИТ)

ИНФОРМАТИКА **~ informatics –** научное направление, изучающее модели, методы, средства, способы сбора, поиска, распознавания, получения, хранения, защиты, обработки и передачи информации.

(Прим.: Термин информатики здесь трактуется расширенно в соответствии с последними публикациями проф.В.Мордвинова и его коллег.)

Теоретической информатикой называют науку о структурах, основывающихся на математике и логике. С другой стороны, ***практическая информатика*** является инженерной дисциплиной, опирающейся на **сети и системы**.

Информатика изучает информационные процессы и связанные с ними явления в технике, природе и обществе. В круг этих вопросов входят базы данных, базы и банки знаний и информационные (автоматизированные, экспертные, информационно-поисковые и другие) системы, гиперсреда. Важное значение в информатике имеют вопросы языков, компьютерного перевода.

Информатика является новой научной областью, опирающейся на традиционные науки. К ним в первую очередь относятся:

- математика — логические структуры данных и математические модели представления информации;
- теория информации — математическое описание методов передачи данных и обработки данных, а также классификация информации;
- искусственный интеллект — способность устройства решать задачи, ассоциируемые с разумными действиями человека;
- электроника, обеспечивающая техническую базу информатики;
- семиотика — комплекс направлений, изучающих знаковые системы.

ИНФОРМАЦИЯ ~ information – в системном понимании это совокупность взаимосвязанных фактов, явлений, событий, подлежащих регистрации и обработке. По

отношению к материальному миру, мирозданию вообще, информация представляет всеобщую, бесконечную в пространств и во времени детерминированную совокупность проявлений, отображающих в той или иной сигнальной форме существование материального мира в его движении и видоизменениях и предопределяющих, задающих эти изменения и движения. Иными словами по отношению к материи информация одновременно первична и вторична. Информация соединяет незримым, но необходимым образом такие категории мироздания, как материя, пространство и время. Видимо, природа появления информации может быть когда-нибудь объяснена переменностью (непостоянством) потока времени, являющегося единственной причиной всемирной энергии, всеобщего движения, и как следствие этого, отображения этих явлений в сигнальной овеществляемой форме, присущей тому, что мы именуем информацией.

Возвращаясь к толкованию информации в системном плане, можно утверждать, что в рассматриваемом понятии информации всегда существуют два партнера: источник и потребитель (приемник) информации. Как первым, так и вторым могут быть объекты науки, техники, общества и природы, животные, люди. Во взаимодействии между ними и рождается информация. В зависимости от области знаний различают научную, патентную информацию, техническую, коммерческую и другие виды.

ИНФОРМАЦИЯ является абстрактным содержанием какого-либо высказывания, описания, указания, сообщения или известия. Внешнюю форму изображения называют **представлением** (конкретная форма сообщения).

Переход от представления к абстрактной информации, то есть к значению представления, называют **интерпретацией**. ВИДЫ ИНФОРМАЦИИ

Существует несколько форм представления информации. **Символьная** основана на использовании символов — букв, цифр, знаков, в том числе знаков пунктуации и других знаков. **Текстовая** также использует образующие тексты символы, но расположенные в определенном порядке. Самой емкой и сложной является **графическая форма**. К ней относятся различные виды изображений. Наименьшей единицей количества информации в двоичной системе счисления является **бит**. Обработка информации осуществляется компьютерами или конечными автоматами. На их основе создаются системы и сети. Большое значение в процессах обработки данных имеет **логика**.

Информация, представленная в виде, удобном для обработки, называется **данными**. Важное значение имеет многомерное представление данных. Определенная структура информационного объекта, подвергаемого обработке, именуется **форматом**.

Информация является объектом изучения информатики.

ИНФОРМАЦИОННАЯ СИСТЕМА. Информационная система (ИС) – инженерное изделие, спроектированное на системной основе, представляющее собой совокупность программных и технических средств, а также реализованного банка данных (банка знаний), позволяющих с помощью специально разработанных в рамках системы методов, методик и нормативных ограничений (стандартов) эффективно в интересах и по запросам пользователя автоматически и однозначно поддерживать сбор, поиск, распознавание, получение, хранение, защиту, обработку и передачу информации.

Информация, находящаяся в какой-либо информационной системе, воспринимается как некоторая математическая структура. Переход от представления к

элементам этой математической структуры называется интерпретацией. Установление отношений к реальному миру называется **пониманием**.

Различные системы представления информации по-разному эффективны. В информатике обычно рассматривается точное описание множества R представлений с интерпретацией I в множестве A элементов (информаций). Интерпретация I данному представлению r (сообщению) ставит в соответствие некоторое абстрактное информационное содержание I[r]. Таким образом, интерпретации соответствует отображение

I:R→A. Через (A,R,I) обозначается **информационная система**. Таким образом, информационная система соответствует понятию отображения из математики. R называют также системой представления, а A – **семантической моделью**.

Система – определенная архитектура взаимодействующих компонент, отграниченных от окружения. Если активности компонент могут происходить одновременно, – то это **параллельно работающие системы** или **параллельно протекающие (параллельные) процессы**. Если такие системы построены из отдельных, удаленных друг от друга в пространстве компонент, тогда речь идет об **распределенных системах**, или об **распределенных процессах**.

Для описания процессов применяются следующие термины:

- **распределение** означает пространственное расположение (или разделение) отдельных компонент процесса;
- **параллелизм** относится к временн***ы***м отношениям между действиями компонентов процесса, которые могут протекать одновременно (параллельно);
- **интерактивность**, **реакция**, **коммуникация**, **координация**, **синхронизация** касаются причинно-следственных отношений между пространственно разделенными и выполняемыми наряду друг с другом действиями (с помощью них осуществляется обмен сообщениями и сигналами);
- **недетерминированность** возникает при моделировании распределенных процессов из-за целенаправленного опускания из рассмотрения определенных деталей информации, например, временных взаимозависимостях в функционировании процесса, которые являются существенными при принятии решений в ходе работы процесса.

Процесс в общем случае состоит из множества **событий**, которые соответствуют выполнению **действий** и возникают в определенном причинно-следственном (временн***о***м) порядке.

ДАННЫЕ ~ data - информация, представленная в формализованном виде, пригодном для автоматической обработки при возможном участии человека.

Данные в информатике формируются в группы, образуя компоненты баз данных и баз знаний. Наименьшим компонентом является *элемент данных* - информационный объект, определяемый его наименованием и совокупностью описывающих его значений (величин). Объектом может быть процесс, явление, предмет, страна, область науки и т.д. Совокупность значений элементов данных, которая описывает рассматриваемый объект, именуется *записью*. Для передачи данных последние формируются в блоки данных, для хранения - компонуются в файлы, каталоги, массивы, таблицы, списки. При этом в системах и сетях может использоваться несколько способов кодирования данных.

БАЗА ДАННЫХ ~ database - совокупность взаимосвязанных данных, организованная по определенным правилам.

Строго говоря, базой данных являются специальным образом организованные один либо группа файлов. Для работы с ними используется система управления базой данных. При этом подразумевается, что база данных определена по схеме, не зависящей от программ, которые к ней обращаются. База данных характеризуется ее концепцией - совокупностью требований, обусловленных представлениями пользователей о необходимой им информации.

Базы данных имеют различные размеры - от небольших, портативных баз данных, находящихся в персональных компьютерах, до крупных, расположенных в суперкомпьютерах. Каждая из отдельно рассматриваемых баз одновременно может обслуживать тысячи пользователей. Все большее распространение получают распределенные базы данных.

Данные в базе располагаются так и для того, чтобы их можно было легко найти и обработать. Эти задачи выполняются системой управления базой данных. Существует много методов доступа к данным, находящимся в базах. Особой популярностью пользуется метод, определяемый *языком структурированных запросов (SQL).* Все большее распространение получают *аудиовидеобазы.* Они характерны тем, что в них размещаются, хранятся и выдаются тексты, звуки, неподвижные и движущиеся изображения.

Увеличение скорости обработки данных, создание большой памяти, построение коммуникационных сетей с высокой пропускной способностью привели к использованию полнотекстовых баз данных. Служба глобального соединения (WWW) обеспечивает универсальный доступ к большому числу баз данных, расположенных на различных континентах.

Увеличивается распространение реляционных баз данных, а также баз данных, имеющих объектно-ориентированную архитектуру и многомерное представление данных. В этих базах создаются модули объектов, в том числе прикладных программ, которые управляются внешними событиями с помощью графического интерфейса пользователя.

БАЗА ЗНАНИЙ ~ knowledge base - организованная совокупность знаний, относящихся к какой-нибудь предметной области.

Знанием является проверенный практикой результат познания действительности. Иначе говоря, знание - это накопленные человечеством истины, факты, принципы и прочие объекты познания. Поэтому в *отличие* от базы данных в базе знаний располагаются познаваемые сведения, содержащиеся в документах, книгах, статьях, отчетах.

В базе знаний в соответствии с принятой в ней методологией классификации располагаются объекты познания, образующие совокупность знаний. В любом объекте представляется набор элементов знаний. Элементы знаний благодаря концептуальным связям объединяются, образуя базу знаний.

Такие связи бывают четырех видов:

- общность,
- партитивность,
- противопоставление,
- функциональная взаимозависимость.

Общность — это связь двух элементов по содержанию их характеристик. Принцип партитивности подразумевает соотношение целого и его частей. Противопоставление встречается в элементах, которые имеют положительные и

отрицательные характеристики. Функциональная взаимозависимость отражает взаимосвязь элементов.

Базы широко используются не только для получения пользователями тех или иных знаний. Они также применяются и при решении задач искусственного интеллекта. Так, в рамках *экспертных систем* используются два важных класса баз. Статическая база знаний содержит сведения, отражающие специфику конкретной области и остающиеся неизменными в ходе решения задачи. Динамическая база знаний используется для хранения сведений, существенных для решения конкретной задачи и меняющихся в процессе этого решения (например, во время проведения лабораторных исследований).

Каждая база знаний включает в себя набор сведений, правил и механизм логического вывода.

БАНК ДАННЫХ ~ databank - комплекс информационных, технических, программных, языковых и организационных средств, обеспечивающих сбор, хранение, поиск и обработку данных.

Банк данных предназначен для хранения больших массивов информации, быстрого поиска нужных сведений и документов. Создается банк в абонентской системе любой производительности — от персонального компьютера до суперкомпьютера. Но даже самый крупный банк ограничен в своих возможностях, поэтому банки в сети специализируются, собирая информацию в определенных областях науки, технологии, продукции.

Банк поддерживается прикладными процессами, получающими в абонентской системе сервис области взаимодействия. Благодаря этой области банк взаимодействует с большим числом пользователей сети, а также с другими банками. Ядром банка являются базы данных и базы знаний.

Лица, работающие в банке данных, делятся на три группы - сотрудники банка, администратор банка и пользователи. Задача сотрудников — сбор и запись в базу всей первичной информации, определяемой тематикой этой базы. Сотрудники должны также удалять устаревшую информацию. Наряду с этим обновление информации может быть разрешено и некоторым пользователям. Сотрудники банка и некоторые пользователи составляют программы, позволяющие из первичной информации получать необходимые вторичные сведения, составлять отчеты. Администратор обеспечивает руководство банком. Он решает вопросы, связанные с бесперебойной и надежной работой, хранением информации и безопасностью данных. В нужных случаях администратор осуществляет копирование содержимого баз и организует хранение копий. Пользователи банка взаимодействуют с необходимыми им банками.

Защита от несанкционированного доступа к базам создается за счет введения паролей и кодов, обеспечивающих идентификацию пользователей. Формирование и ведение банков данных связаны с большими затратами. Они становятся рентабельными лишь при большом *трафике.*

В различных странах создано большое число банков данных. Одним из крупнейших в мире является расположенный в США банк LEXIS-NEXIS, который в 1994 г. предоставлял около 200 млн. документов из более 4000 источников.

БЕЗОПАСНОСТЬ ДАННЫХ ~ data security - концепция защиты данных от случайного, либо умышленного изменения, уничтожения, разглашения, а также несанкционированного использования.

Безопасность данных является многоплановой проблемой, охватывающей ряд важных задач. В первую очередь к ним относится конфиденциальность, которая

обеспечивается за счет средств криптографии. *Шифрование* позволяет не только закрывать данные от посторонних лиц. Оно также решает задачу *целостности данных* — их сохранения при наличии помех и искажения.

Важным аспектом безопасности данных является идентификация пользователя - процесс анализа характеристик кодов, используемых им для подтверждения прав на доступ в сеть, работу с данными и их изменение. Она обеспечивается введением паролей и анализом электронной подписи. В результате осуществляется так называемый санкционированный (разрешенный) доступ. В этом процессе определяется несколько групп пользователей. Первым разрешается только чтение файлов, вторым - еще и изменение перечня файлов, их ликвидация, создание новых файлов. Особо выделяется круг лиц, работающих с конкретными файлами (редактирование, копирование).

Несанкционированный доступ связан не только с использованием, изменением или уничтожением данных лицами, которые не имеют на это разрешения. Для предотвращения противоправных действий создается безопасная среда, в которой обеспечивается также защита от всевозможных перехватов блоков данных во время передачи, их подделки, искажения, расшифровки паролей и идентификаторов пользователей, искусственного прерывания сеансов взаимодействия прикладных процессов.

С появлением и развитием сетей возникли компьютерные преступления. *Хакеры* (от англ. hack - рубить, кромсать, разбивать) вторгаются в программное обеспечение в целях кражи, искажения либо порчи данных. Возникла также проблема, связанная с появлением в системах и сетях компьютерных вирусов. Последние способны искажать либо уничтожать используемую информацию.

При решении проблемы безопасности данных обеспечивается так называемая прозрачность принимаемых мер. Она заключается в том, что введение механизмов безопасности не должно изменять нормальную работу сети. При этом задержки в передаче данных, вносимые программными и техническими средствами безопасности, должны быть минимальными. Не должна также уменьшаться надежность передачи. Естественно, что средства безопасности, в свою очередь, должны быть защищены от несанкционированного вторжения в них.

Для обеспечения юридической базы безопасности данных в ряде стран приняты соответствующие законы.

Агрегат данных - поименованная совокупность элементов данных внутри записи, рассматриваемая как единое целое. Например, агрегат данных «ДАТА» может состоять из элементов данных «МЕСЯЦ», «ДЕНЬ», «ГОД». В языке Кобол агрегат данных называется группой или групповым элементом. Data aggregate (агрегат данных) Группа записей (record, 1) или блоков (block, 2) данных, которая обычно содержит описания местоположения каждого блока и его взаимосвязей с другими блоками.

Существуют два типа агрегатов данных: векторы и повторяющиеся группы.

Вектор – одномерная упорядоченная совокупность элементов данных (например, приведенная выше совокупность ДАТА).

Повторяющаяся группа – совокупность данных, которые встречаются несколько раз в экземпляре записи, например прием и выдача вкладов в записи счета сберкассы. В повторяющуюся группу могут входить отдельные элементы данных, векторы, агрегаты данных или другие повторяющиеся группы.

Агрегатам данных соответствуют абстрактные типы данных, представляемые, например, структурами в языках С, С++ и обычными записями в языках Паскаль и Модула. Они состоят из объектов данных, подключаемых непосредственно или с

использованием ссылок (указателей). При обсуждении агрегатов данных, используемых при процедурном подходе, применяется также понятие «запись». Понятие «структура», для обозначения агрегата данных не используется в связи с его перегруженностью другими смысловыми значениями.

Процедурный подход предполагает независимость записей (R) от процедур (P). Запись R состоит из элементов данных: R=(D1, D2, ..., Dk), называемых полями записи. При этом поле Di может, в свою очередь, состоять из других абстракций.

Все значения данных состоят из простых типов данных. В отличие от некоторых других языков в SQL отсутствуют массивы, указатели, векторы и другие сложные типы данных.

Все данные в реляционной базе изображаются в форме двухмерных таблиц (на языке математики – «отношений»). Каждая таблица содержит некоторое число строк (в том числе 0), называемых «картежами» и один или несколько столбцов, называемых «атрибутами». Все строки в таблице имеют одну и ту же последовательность столбцов, в которых записаны различные значения, однако наборы значений в столбцах отличаются.

Одной из основных особенностей реляционной модели является требование, чтобы каждое значение в столбце являлось атомарной величиной, то есть в каждом столбце для каждой строке может содержаться только одно значение. Если поместить несколько значений, они всегда будут обрабатываться СУБД, как единая величина.

КОМПЬЮТЕРНАЯ РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ~ computer-aided software engineering - технология автоматизированной разработки программного обеспечения включает комплекс средств, предназначенных для разработки программ, создания общей базы данных, использования единого метода взаимодействия с этой базой. Причем этот подход учитывает разнообразие операционных систем, применяемых в системах, включаемых в одну информационную сеть. Кроме того, CASE определяет единую основу для сетевых технологий, используемых разными разработчиками. CASE также предоставляет методологию и средства тестирования создаваемых программ.

Благодаря использованию CASE снижается стоимость разработок, и уменьшаются сроки их проведения. До последнего времени технология CASE разрабатывалась различными организациями. Национальный институт стандартов и технологии (NIST) США предлагает свой стандартный подход в создании CASE. Все в большей степени расширяется рынок прикладных программ, обеспечивающих CASE. К ним, например, относятся быстрая разработка программ (RAD), среда программирования AppWare, технология OpenDoc.

ОБУЧАЮЩАЯ СИСТЕМА ~ training system - система, предназначенная для обучения пользователей.

Обучающая система основывается на использовании искусственного интеллекта и базы знаний. В ней широко применяются технологии гиперсреды и гипертекста. Это позволяет выделять объекты знаний и ассоциативно связывать их друг с другом. Система осуществляет «навигацию» пользователя по базе знаний, переходя от одного объекта к другому.

Основной задачей обучающей системы является эффективная передача знаний в зависимости от степени подготовленности пользователей и их способности усваивать получаемую информацию. Системы могут быть автономными и сетевыми. Автономные являются индивидуальными и функционируют на отдельных персональных компьютерах. Сетевые являются коллективными и располагаются на

серверах, с которыми могут работать клиенты пользователей. Все более широкое распространение в обучении получают виртуальные классы. Управление процессом обучения выполняет как пользователь, так и программы обучающей системы.

Система осуществляет проверку знаний учащегося, ведет регистрацию этапов процесса. Обучение сопровождается неподвижными изображениями, видеосюжетами и фрагментами звука, в том числе речи. Процесс обучения нередко называется *программируемым обучением*.

Системный подход в задачах анализа, моделирования и структурирования ИС и АСУ образования и науки

Предопределяющим фактором в достижении высокой эффективности проектируемых комплексных ИС и АСУ дидактического, научного и управленческого назначения является опора на системный подход при решении задач анализа, моделирования и структурирования указанных систем. Системный подход имеет первостепенное значение в комплексном информационном обслуживании научных исследований, творчества, обучения и управленческой деятельности, так как органически соединяет анализ и синтез. Системный подход предполагает рассмотрение сложного объекта как системы, то есть в виде совокупности конечного множества элементов, связанных в единое целое структурно как части другой более широкой системы и в то же время как целостной системы, состоящей из подсистем, которые могут изучаться в качестве самостоятельных объектов. Так, например, информационные системы поддержки отдельных учебных дисциплин могут одновременно представлять подсистемы информационного обеспечения той или иной специальности высшей школы на весь период обучения, а ИС специальностей, в свою очередь, входить в комплексную ИС всего многопрофильного процесса обучения в вузе и\или в единую межвузовскую отраслевую ИС по данной специальности. Точно также отдельные частные направления творчества и научных исследований поддерживаются на системном и подсистемном уровнях по отношению к многоплановым исследованиям и проектам учреждения, ведомства, отрасли, региона и так далее.

Характерные признаки сложной системы – многомерность, многообразие и переменность структуры, тесная взаимосвязь элементов, изменение их связей и состояния. Именно эти характеристики как нельзя более относятся к образованию, творчеству, науке, информационно-социальным технологиям и явлениям.

Одним из важнейших принципов системного подхода является исследование элементов сложной системы. Только на основе детального качественного изучения элементов системы возможен достоверный синтез и познание интегральных закономерностей системы как целого. Такой анализ позволяет выявить противоречия в сложных системах, их иерархичность, структуру главных противоречий, определить рациональные способы их разрешения на каждом этапе развития.

Системный анализ представляет собой совокупность научных методов решения проблем на основе системного подхода и рассмотрения объекта исследования в виде системы. Системный анализ предусматривает использование наряду с формализованными качественных методов анализа и имеет практическую ориентацию – вместе с выявлением и всесторонним анализом проблем разрабатываются конкретные меры по их решению.

Логика системного анализа, с одной стороны предусматривает структуризацию каждой проблемы, разбивку ее на подпроблемы, которые требуют особого подхода (вследствие специфичности условий и факторов их разрешения) и имеют частные

оптимальные решения; с другой стороны, предопределяет целостность решения проблемы, что предполагает подчинение частных решений общей цели ее разрешения. Такой подход подразумевает: определение целей функционирования и развития данной системы, анализ уровней ее иерархии и выделение отдельных подсистем, выявление и постановку проблем, определение их сущности и структуры, оценку разрешимости, формирование альтернативных разрешений проблем на основе анализа и использования факторов и усилий.

Применительно к образовательным технологиям в ходе системного анализа изучаются основные количественные и качественные изменения, происходящие в накоплении знаний, умений и навыков учащимися в процессе реализации комплекса работ, мер, усилий по овладению учебным материалом и его творческой компонентой в результате пошагового обучения и творчества (от темы к теме, от задачи к задаче) в целостной подсистеме единой информсреде данной учебной дисциплины в ее взаимосвязях с другими учебными дисциплинами и учебно-творческими задачами различного уровня вплоть до выпускной итоговой творческой отчетной работы (дипломного проекта для студентов вузов и техникумов, выпускной творческой работы в учреждениях дополнительного образования, диссертации в аспирантуре и докторантуре и т.п.). При этом информсреда образования и науки равно как и средства ее поддержки (ИС, АСУ) не являются застывшим явлением, а наоборот, непрерывно расширяются, развиваются и совершенствуются. Это обусловлено прежде всего тем, что в практике работ и на основании специально проводимого системного анализа выявляются основные проблемы функционирования и дальнейшего развития, определяются и оцениваются альтернативные пути их разрешения. При этом нужно руководствоваться следующими принципами:

- рассматривать проблемы во взаимосвязи;
- учитывать целевую ориентацию их разрешения;
- исходить из возможности решения каждой проблемы различными путями;
- использовать однотипный подход к исследованию групп проблем, объединенных по определенным признакам;
- учитывать ближайшие и более отдаленные профессионально-значимые, экономические и социальные последствия разрешения важнейших проблем;
- исходить из понимания четкой и постоянно укрепляющейся связи образования, науки, культуры и социологии, а также возрастании роли личности каждого участника образовательных и научных процессов в их реализации;
- видеть в информатизации всех процессов и явлений в образовании и науке безальтернативную линию их успешного развития, полагая при этом, что высокая эффективность информатизации опирается исключительно на эффективное создание и применение современных высокопродуктивных ИС и АСУ, функционирующих в мировом информационном Интернет-пространстве с широким использованием современных Интернет\Экстранет/Интранет-технологий.

Общая архитектура информационных и информационно-аналитических систем

Современный уровень развития аппаратных и программных средств сделал возможным повсеместное ведение баз данных (БД) оперативной и накопленной информации на разных уровнях управления и информационного обеспечения. В процессе своей деятельности промышленные предприятия, корпорации, ведомственные структуры, органы государственной власти и управления, социальные структуры, учреждения образования, науки и культуры, отдельные граждане накопили большие объемы данных. Они хранят в себе большие потенциальные возможности по

извлечению полезной аналитической информации, на основе которой можно выявлять скрытые тенденции, строить стратегию развития, находить новые решения, осуществлять различного рода проектирование, вести обучение, поддерживать творчество и научный поиск.

Полная структура информационно-аналитической системы (как и некоторые другие материалы настоящих глав приведены в Интернете в творческой работе С.А.Митичкина, выполненной под руководством доц. Н.Н.Лычкиной), построенной на основе хранилища данных, показана на рисунке ниже В конкретных реализациях отдельные компоненты этой схемы часто отсутствуют. Так, в ИС, предназначенных только для информационной поддержки преподавания отдельных дисциплин в учебной деятельности, могут иметь ослабленные те или иные входящие системы (подсистемы) или не иметь их вовсе. Дело в том, что комплексную информационно-аналитическую систему корпорации, предприятия и даже отдельного подразделения можно рассматривать как совокупность функционирующих воедино набора различных частных систем (подсистем). Поэтому для большей последовательности изложения условимся рассматривать почти всякую ИС образовательных и научных технологий как частный случай или составную часть развернутой комплексной универсальной информационно-аналитической системы, изображение архитектуры которой приводится ниже.

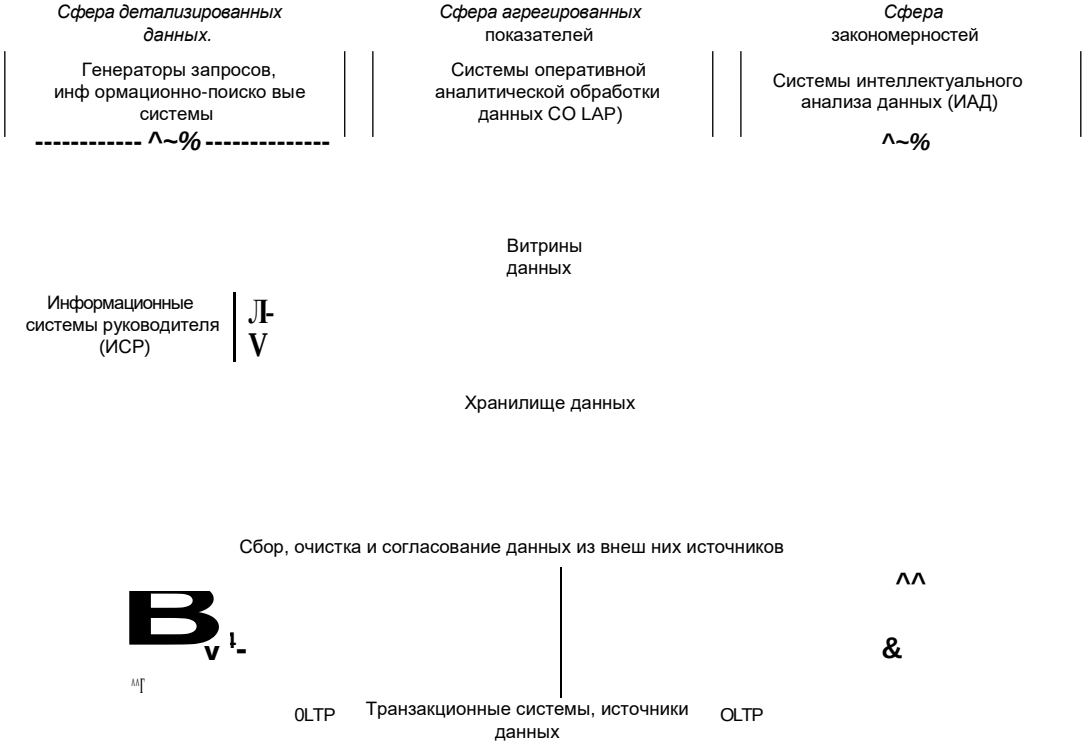


Рис. Общая архитектура информационно-аналитической системы (ИАС)

Данная архитектура является достаточно общепринятым пониманием места и роли различных технологий обработки данных, в том числе используемых в образовательных технологиях и научных исследованиях.

Очень часто информационно-аналитические системы, создаваемые в расчете на непосредственное использование лицами, принимающими решения, оказываются чрезвычайно просты в применении, но жестко ограничены в функциональности. Такие статические системы называются в литературе **Информационными системами руководителя (ИСР)**, или Executive Information Systems (EIS). Они содержат в себе предопределенные множества запросов и, будучи достаточными для повседневного

обзора, не способны ответить на все вопросы к имеющимся данным, которые могут

возникнуть при принятии решений. Результатом работы такой системы, как правило, являются многостраничные отчеты, после тщательного изучения которых, у аналитика появляется новая серия вопросов. Однако каждый новый запрос, непредусмотренный при проектировании такой системы, должен быть сначала формально описан, закодирован программистом и только затем выполнен. Время ожидания в таком случае может составлять часы и дни, что не всегда приемлемо. Таким образом, внешняя простота статических информационно-аналитических систем (статических СППР), за которую активно борется большинство заказчиков информационно-аналитических систем, оборачивается катастрофической потерей гибкости.

Динамические информационнно-аналитические системы (или динамические СППР, DSS), напротив, ориентированы на обработку нерегламентированных (ad hoc) запросов аналитиков к данным. Работа аналитиков с этими системами заключается в интерактивной последовательности формирования запросов и изучения их результатов.

Но динамические СППР могут действовать не только в области оперативной аналитической обработки (OLAP); поддержка принятия управленческих решений на основе накопленных данных может выполняться в трех базовых сферах.

1. Сфера детализированных данных. Это область действия большинства систем, нацеленных на поиск информации. В большинстве случаев реляционные СУБД отлично справляются с возникающими здесь задачами. Общепризнанным стандартом языка манипулирования реляционными данными является SQL. Информационно-поисковые системы (ИПС), обеспечивающие интерфейс конечного пользователя в задачах поиска детализированной информации, могут использоваться в качестве надстроек как над отдельными базами данных транзакционных систем, так и над общим хранилищем данных. Эти ИПС имеют и будут долгое время иметь чрезвычайно большое распространение в научных исследованиях и учебном процессе, особенно в учебно-творческой деятельности, связанной с проектированием (дипломным, курсовым и т.д.).
2. Сфера агрегированных показателей и данных. Комплексный взгляд на собранную в хранилище данных информацию, ее обобщение и агрегация, гиперкубическое представление и многомерный анализ являются задачами систем оперативной аналитической обработки данных (OLAP) Здесь можно или ориентироваться на специальные многомерные СУБД, или оставаться в рамках реляционных технологий. Во втором случае заранее агрегированные данные могут собираться в БД звездообразного вида, либо агрегация информации может производиться на лету в процессе сканирования детализированных таблиц реляционной БД. Комплексность оценок и анализа накопленной и быстро обновляемой информации делает описанный выше подход привлекательным для информационного обслуживания профессионального обучения ВШ по отдельным специальностям, в объеме функционирования отдельной кафедры или в рамках отдельно взятой большой многогранной комплексной научной проблемы.
3. Сфера закономерностей. Интеллектуальная обработка производится методами интеллектуального анализа данных (ИАД, Data Mining), главными задачами которых являются поиск функциональных и логических закономерностей в накопленной информации, построение моделей и правил, которые объясняют найденные аномалии и/или прогнозируют развитие некоторых процессов. Иными словами, речь идет о развитии информационно-аналитического обслуживания образования и науки на уровне интеллектуальных систем.

Следует отметить, что в последние годы в мире оформился ряд новых концепций хранения и анализа данных:

- хранилища данных, или Склады данных (Data Warehouse);
- оперативная аналитическая обработка (On-Line Analytical Processing, OLAP);
- интеллектуальный анализ данных - ИАД (Data Mining).

Технологии OLAP тесно связаны с технологиями построения Data Warehouse и методами интеллектуальной обработки - Data Mining. Поэтому наилучшим вариантом является комплексный подход к их выбору и внедрению. В этой связи уместно рассмотреть несколько концептуальных подходов, полезных разработчику архитектуры ИС.

Концепция хранилищ данных (Data Warehouse)

Для того чтобы существующие хранилища данных наилучшим образом способствовали принятию проектных и управленческих решений, информация должна быть представлена аналитику, преподавателю или учащемуся в нужной форме, то есть он должен иметь развитые инструменты доступа к данным хранилища и их обработки.

Автором концепции Хранилищ Данных (Data Warehouse) является Б.Инмон, который определил Хранилища Данных, как: “предметно ориентированные, интегрированные, неизменчивые, поддерживающие хронологию наборы данных, организованные для целей поддержки управления”, призванные выступать в роли “единого и единственного источника истины” обеспечивающего менеджеров и аналитиков достоверной информацией необходимой для оперативного анализа и принятия решений.

В основе концепции Хранилищ Данных лежат две основополагающие идеи:

- интеграция ранее разъединенных детализированных данных (исторические архивы, данные из традиционных СОД, данные из внешних источников) в едином Хранилище Данных, их согласование и возможно агрегация;
- разделение наборов данных используемых для операционной обработки и наборов данных используемых для решения задач анализа.

Кроме единого справочника метаданных, средств выгрузки, агрегации и согласования данных, концепция Хранилищ Данных подразумевает: ***интегрированность, неизменчивость, поддержку хронологии и согласованность данных***. И если, два первых свойства (*интегрированность и неизменчивость)* влияют на режимы анализа данных, то последние два (*поддержка хронологии и согласованность)*, существенно сужают список решаемых аналитических задач.

Без поддержки хронологии (наличия исторических данных) нельзя говорить о решении задач прогнозирования и анализа тенденций. Но наиболее критичными и болезненными, оказываются вопросы, связанные с согласованием данных.

Основным требованием аналитика, является даже не столько оперативность, сколько достоверность ответа. Но достоверность, в конечном счете, и определяется согласованностью. Пока не проведена работа по взаимному согласованию значений данных из различных источников, сложно говорить об их достоверности. На практике различные ИС дают различные, иногда противоречивые ответы на один и тот же запрос. Противодействие этому негативному для образования и науки явлению – одна из основных задач создателя информационной системы для научных и образовательных технологий.

Концепция Витрин Данных (Data Mart)

Концепция Витрин Данных (Data Mart) была предложена Forrester Research ещё в 1991. По замыслу авторов, Витрины Данных: - *множество тематических БД,*

содержащих информацию, относящуюся к отдельным аспектам деятельности организации, пользователей.

Концепция Витрин Данных имеет ряд несомненных достоинств:

- аналитики видят и работают только с теми данными, которые им реально нужны;
- целевая БД Витрины Данных, максимально приближена к конечному пользователю;
- витрины Данных обычно содержат тематические подмножества заранее агрегированных данных, их проще проектировать и настраивать;
- для реализации Витрин Данных не требуются особо мощная вычислительная техника.

Однако, концепция Витрин Данных имеет и очень серьёзные пробелы. По существу, здесь предполагается реализация территориально распределённой информационной системы с мало контролируемой избыточностью, но не предлагается способов, как обеспечить целостность и непротиворечивость хранимых в ней данных.

Объединенная концепция Хранилищ и Витрин Данных

Идея соединить две концепции - Хранилищ Данных и Витрин Данных, по видимому, принадлежит М.Демаресту (M.Demarest), который, в 1994 году предложил объединить две концепции и использовать Хранилище Данных в качестве единого интегрированного источника данных для Витрин Данных. Такое решение имеет три уровня:

- первый уровень — общекорпоративная БД на основе РСУБД с нормализованной или слабо де нормализованной схемой (детализированные данные);
- второй уровень — БД уровня подразделения (или конечного пользователя), реализуемые на основе МСУБД (агрегированные данные);
- третий уровень — рабочие места конечных пользователей, на которых непосредственно установлен аналитический инструментарий;

Объединенная система постепенно становится стандартом де-факто, позволяя наиболее полно реализовать и использовать достоинства каждого из подходов:

- компактное хранение детализированных данных и поддержка очень больших БД, обеспечиваемые реляционными СУБД;
- простота настройки и хорошие времена отклика, при работе с агрегированными данными, обеспечиваемые многомерными СУБД.

Реляционная форма представления данных, используемая в центральной общекорпоративной БД, обеспечивает наиболее компактный способ хранения данных.

К тому же современные реляционные СУБД уже умеют работать даже с терабайтными базами. И хотя, такая центральная система, обычно не сможет обеспечить оперативного режима обработки аналитических запросов, при использовании новых способов индексации и хранения данных, а так же частичной денормализации таблиц, время обработки заранее регламентированных запросов (а в качестве таких, можно рассматривать и регламентированные процедуры выгрузки данных в многомерные БД) оказывается вполне приемлемым.

В свою очередь, использование многомерных СУБД в узлах нижнего уровня обеспечивает минимальные времена обработки и ответа на нерегламентированные запросы пользователя. Кроме того, в некоторых многомерных СУБД имеется возможность хранить данные как на постоянной основе (непосредственно в многомерной БД), так и динамически (на время сеанса) загрузить данные из реляционных БД (на основе регламентированных запросов).

Таким образом, имеется возможность хранить на постоянной основе, только те данные, которые наиболее часто запрашиваются в данном узле. Для всех остальных, хранятся только описания их структуры и программы их выгрузки из центральной БД. Отмечая то обстоятельство, что при первичном обращении к таким виртуальным данным время отклика может оказаться достаточно продолжительным, тем не менее можно полагать, что такое решение обеспечивает высокую гибкость и требует более дешевых аппаратных средств.

Оперативная аналитическая обработка данных (OLAP)

В основе концепции OLAP лежит принцип многомерного представления данных.

В 1993 году E. F. Codd рассмотрел недостатки реляционной модели, в первую очередь указав на невозможность "объединять, просматривать и анализировать данные с точки зрения множественности измерений, то есть самым понятным для корпоративных аналитиков способом", и определил общие требования к системам OLAP, расширяющим функциональность реляционных СУБД и включающим многомерный анализ как одну из своих характеристик.

По Кодду, многомерное концептуальное представление (multi-dimensional conceptual view) представляет собой множественную перспективу, состоящую из нескольких независимых измерений, вдоль которых могут быть проанализированы определенные совокупности данных. Одновременный анализ по нескольким измерениям определяется как многомерный анализ. Каждое измерение включает направления консолидации данных, состоящие из серии последовательных уровней обобщения, где каждый вышестоящий уровень соответствует большей степени агрегации данных по соответствующему измерению. Так, измерение «Исполнитель» может определяться направлением консолидации, состоящим из уровней обобщения "предприятие - подразделение - отдел - служащий". Измерение «Время» может даже включать два направления консолидации - "год - квартал - месяц - день" и "неделя - день", поскольку счет времени по месяцам и по неделям несовместим. В этом случае становится возможным произвольный выбор желаемого уровня детализации информации по каждому из измерений. Операция спуска (drilling down) соответствует движению от высших ступеней консолидации к низшим; напротив, операция подъема (rolling up) означает движение от низших уровней к высшим.

Кодд определил 12 правил, которым должен удовлетворять программный продукт класса OLAP:

Таблица Правила оценки программных продуктов класса OLAP по Кодду

№	Правило	Описание
1.	Многомерное концептуальное представление данных (Multi-Dimensional Conceptual View)	Концептуальное представление модели данных в продукте OLAP должно быть многомерным по своей природе, то есть позволять аналитикам выполнять интуитивные операции "анализа вдоль и поперек" ("slice and dice"), вращения (rotate) и размещения (pivot) направлений консолидации. Пользователь не должен знать о том, какие конкретные
2.	Прозрачность (Transparency)	средства используются для хранения и обработки данных, как данные
3.		организованы и откуда берутся. Доступность (Accessibility) Аналитик должен иметь возможность выполнять анализ в рамках общей концептуальной схемы, но при этом данные

		могут оставаться под управлением оставшихся от старого наследства СУБД, будучи при этом привязанными к общей аналитической модели. То есть инструментарий OLAP должен накладывать свою логическую схему на физические массивы данных, выполняя все преобразования, требующиеся для обеспечения единого, согласованного и целостного взгляда пользователя на информацию.
4.	Устойчивая производительность (Consistent Reporting Performance)	С увеличением числа измерений и размеров базы данных аналитики не должны столкнуться с каким бы то ни было уменьшением производительности. Устойчивая производительность необходима для поддержания простоты использования и свободы от усложнений, которые требуются для доведения OLAP до конечного пользователя.
5.	Клиент - серверная архитектура (Client-Server Architecture)	Большая часть данных, требующих оперативной аналитической обработки, хранится в мэйнфреймовых системах, а извлекается с персональных компьютеров. Поэтому одним из требований является способность продуктов OLAP работать в среде клиент-сервер. Главной идеей здесь является то, что серверный компонент инструмента OLAP должен быть достаточно интеллектуальным и обладать способностью строить общую концептуальную схему на основе обобщения и консолидации различных логических и физических схем корпоративных баз данных для обеспечения эффекта прозрачности.
6.	Равноправие измерений (Generic Dimensionality)	Все измерения данных должны быть равноправны. Дополнительные характеристики могут быть предоставлены отдельным измерениям, но поскольку все они симметричны, данная дополнительная функциональность может быть предоставлена любому измерению. Базовая структура данных, формулы и форматы отчетов не должны опираться на какое-то одно измерение.
7.	Динамическая обработка разреженных матриц (Dynamic Sparse Matrix Handling)	Инструмент OLAP должен обеспечивать оптимальную обработку разреженных матриц. Скорость доступа должна сохраняться вне зависимости от расположения ячеек данных и быть постоянной величиной для моделей, имеющих разное число измерений и различную разреженность данных.
8.	Поддержка многопользовательского режима (Multi-User Support)	Зачастую несколько аналитиков имеют необходимость работать одновременно с одной аналитической моделью или создавать различные модели на основе одних корпоративных данных. Инструмент OLAP должен предоставлять им конкурентный доступ, обеспечивать целостность и защиту данных.
9.	Неограниченная поддержка кроссмерных операций (Unrestricted	Вычисления и манипуляция данными по любому числу измерений не должны запрещать или ограничивать любые отношения между ячейками данных. Преобразования,

	Cross-dimensional Operations)	требующие произвольного определения, должны задаваться на функционально полном формульном языке.
10.	Интуитивное манипулирование данными (Intuitive Data Manipulation)	Переориентация направлений консолидации, детализация данных в колонках и строках, агрегация и другие манипуляции, свойственные структуре иерархии направлений консолидации, должны выполняться в максимально удобном, естественном и комфортном пользовательском интерфейсе.
11.	Гибкий механизм генерации отчетов (Flexible Reporting)	Должны поддерживаться различные способы визуализации данных, то есть отчеты должны представляться в любой возможной ориентации.
12.	Неограниченное количество измерений и уровней агрегации (Unlimited Dimensions and Aggregation Levels)	Настоятельно рекомендуется допущение в каждом серьезном OLAP инструменте как минимум пятнадцати, а лучше двадцати, измерений в аналитической модели. Более того, каждое из этих измерений должно допускать практически неограниченное количество определенных пользователем уровней агрегации по любому направлению консолидации.

Набор этих требований, послуживших фактическим определением OLAP, следует рассматривать как рекомендательный, а конкретные продукты оценивать по степени приближения к идеально полному соответствию всем требованиям.

Интеллектуальный анализ данных (Data Mining)

В последнее время оформилось новое направление в аналитических технологиях обработки данных — Data Mining, что переводится как "добыча" или "раскопка данных" Нередко рядом с Data Mining встречаются слова "обнаружение знаний в базах данных" (knowledge discovery in databases) и "интеллектуальный анализ данных". Их можно считать синонимами Data Mining. Возникновение всех указанных терминов связано с новым витком в развитии средств и методов обработки данных.

Цель Data Mining состоит в выявлении скрытых правил и закономерностей в наборах данных. Дело в том, что человеческий разум сам по себе не приспособлен для восприятия больших массивов разнородной информации. Человек к тому же не способен улавливать более двух-трех взаимосвязей даже в небольших выборках. Но и традиционная математическая статистика, долгое время претендовавшая на роль основного инструмента анализа данных, также нередко пасует при решении задач из реальной сложной жизни. Она оперирует усредненными характеристиками выборки, которые часто являются фиктивными величинами (типа средней температуры пациентов по больнице, средней высоты дома на улице, состоящей из дворцов и лачуг и т.п.). Поэтому методы математической статистики оказываются полезными главным образом для проверки заранее сформулированных гипотез (verification-driven data mining).

Современные технологии Data Mining (discovery-driven data mining) перелопачивают информацию с целью автоматического поиска шаблонов (паттернов), характерных для каких-либо фрагментов неоднородных многомерных данных. В отличие от оперативной аналитической обработки данных (online analytical processing, OLAP) в Data Mining бремя формулировки гипотез и выявления необычных (unexpected) шаблонов переложено с человека на компьютер.

Таблица: Примеры формулировок задач при использовании методов OLAP и Data Mining

OLAP	**Data Mining**
Каковы средние показатели успеваемости для различных контрольных групп учащихся (посещающих и непосещающих лекции)?	Как отражается посещаемость лекций на успеваемость в контрольных группах учащихся?
Каковы средние временные затраты студента на курсовой проект по сравнению с курсовой работой?	Какие характеристики курсового проекта и курсовой работы отвечают за различия в средних временных затрат студентов на их выполнение?
Какова средняя величина ежедневного пользования кафедрального Интернет-ресурса студентами в учебных и внеучебных целях?	Какие факторы влияют на распределение «учебные\внеучебные цели» ежедневного использования студентами кафедрального Интернет-ресурса?

В постановке задачи Data Mining специалисты видят решение задач типа "поиск эмпирических закономерностей", "эвристический поиск в сложных средах", "индуктивный вывод" и т. п. С развитием Интернет-пространства еще в большей степени возросла практическая ценность умения решать такие задачи. Во-первых, в связи с развитием технологий записи, передачи и хранения данных на людей обрушились колоссальные информационные потоки в самых различных областях, которые без продуктивной переработки грозят превратиться в полнейший и зловредный информационный хаос. И, во-вторых, средства и методы обработки данных стали доступными и удобными, а их результаты понятными любому человеку.

Типы закономерностей, выявляемых с использованием метода Data Mining

Выделяют пять стандартных типов закономерностей, которые позволяют выявлять методы Data Mining:

- ассоциация;
- последовательность;
- классификация;
- кластеризация;
- прогнозирование.

Ассоциация имеет место в том случае, если несколько событий связаны друг с другом. Например, исследование, проведенное в потоке учащихся, может показать, что 60% выступивших на семинарах успешно сдают зачет с первого раза, а при системе накопления баллов для зачета за выступления на семинарах успеваемость зачетной сессии с первого предъявления возрастает до 80%. Располагая сведениями о подобной ассоциации, педагогам легко оценить, насколько действенна предоставляемая возможность постепенной сдачи зачета еще на семинарах путем накопления баллов.

Если существует цепочка связанных во времени событий, то говорят о ***последовательности***. Так, например, опрос студентов в одном из московских вузов показал, что после покупки модема к компьютеру студентом-пользователем в 90% случаев в течение месяца приобретается Интернет-карточка, а в пределах трех месяцев изыскиваются наиболее эффективные провайдеры.

С помощью ***классификации*** выявляются признаки, характеризующие группу, к которой принадлежит тот или иной объект. Это делается посредством анализа уже классифицированных объектов и формулирования некоторого набора правил.

Кластеризация отличается от классификации тем, что сами группы заранее не заданы. С помощью кластеризации средства Data Mining самостоятельно выделяют различные однородные группы данных.

Основой для всевозможных систем ***прогнозирования*** служит историческая информация, хранящаяся в БД в виде временных рядов. Если удается построить математическую модель и найти шаблоны, адекватно отражающие эту динамику, есть вероятность, что с их помощью можно предсказать и поведение системы в будущем.

Классы и подклассы систем интеллектуального анализа данных

Предметно-ориентированные аналитические системы очень разнообразны. Наиболее широкий подкласс таких систем, получивший распространение в области исследования финансовых рынков, носит название "технический анализ". Он представляет собой совокупность нескольких десятков методов прогноза динамики цен и выбора оптимальной структуры инвестиционного портфеля, основанных на различных эмпирических моделях динамики рынка. Эти методы могут быть весьма просты (например, методы, использующие вычитание трендового значения), но могут иметь достаточно оригинальную математическую основу (например, теорию фракталов). Поскольку чаще всего теория "зашита" в эти системы, а не выводится на основании истории рынка, то требования статистической значимости выводимых моделей и возможности их интерпретации для них не имеют смысла.

Хотя последние версии почти всех известных **статистических пакетов** включают наряду с традиционными статистическими методами также элементы Data Mining, основное внимание в них уделяется все же классическим методикам — корреляционному, регрессионному, факторному анализу и другим. Недостатком систем этого класса считают требование к специальной подготовке пользователя. Также отмечают, что мощные современные статистические пакеты являются слишком "тяжеловесными" для массового применения в финансах и бизнесе.

Есть еще более серьезный принципиальный недостаток статистических пакетов, ограничивающий их применение в Data Mining. Большинство методов, входящих в состав пакетов опираются на статистическую парадигму, в которой главными фигурантами служат усредненные характеристики выборки. А эти характеристики при исследовании реальных сложных жизненных феноменов часто являются фиктивными величинами.

Нейронные сети — это большой класс систем, архитектура которых пытается имитировать построение нервной ткани из нейронов. В одной из наиболее распространенных архитектур, многослойном персептроне с обратным распространением ошибки, эмулируется работа нейронов в составе иерархической сети, где каждый нейрон более высокого уровня соединен своими входами с выходами нейронов нижележащего слоя. На нейроны самого нижнего слоя подаются значения входных параметров, на основе которых нужно принимать какие-то решения, прогнозировать развитие ситуации и т. д. Эти значения рассматриваются как сигналы, передающиеся в вышележащий слой, ослабляясь или усиливаясь в зависимости от числовых значений (весов), приписываемых межнейронным связям. В результате на выходе нейрона самого верхнего слоя вырабатывается некоторое значение, которое рассматривается как ответ, реакция всей сети на введенные значения входных параметров. Для того чтобы сеть можно было применять в дальнейшем, ее прежде надо "натренировать" на полученных ранее данных, для которых известны и значения входных параметров, и правильные ответы на них. Эта тренировка состоит в подборе

весов межнейронных связей, обеспечивающих наибольшую близость ответов сети к известным правильным ответам.

Основным недостатком нейросетевой парадигмы является необходимость иметь очень большой объем обучающей выборки. Другой существенный недостаток заключается в том, что даже натренированная нейронная сеть представляет собой черный ящик. Знания, зафиксированные как веса нескольких сотен межнейронных связей, совершенно не поддаются анализу и интерпретации человеком (известные попытки дать интерпретацию структуре настроенной нейросети могут выглядеть недостаточно убедительными).

Идея **систем рассуждений на основе аналогичных случаев** (case based reasoning — CBR) на первый взгляд крайне проста. Для того чтобы сделать прогноз на будущее или выбрать правильное решение, эти системы находят в прошлом близкие аналоги наличной ситуации и выбирают тот же ответ, который был для них правильным. Поэтому этот метод еще называют методом "ближайшего соседа" (nearest neighbour).

Системы CBR показывают очень хорошие результаты в самых разнообразных задачах. Главным их минусом считают то, что они вообще не создают каких-либо моделей или правил, обобщающих предыдущий опыт, — в выборе решения они основываются на всем массиве доступных исторических данных, поэтому невозможно сказать, на основе каких конкретно факторов CBR системы строят свои ответы. Другой минус заключается в произволе, который допускают системы CBR при выборе меры "близости". От этой меры самым решительным образом зависит объем множества прецедентов, которые нужно хранить в памяти для достижения удовлетворительной классификации или прогноза.

Деревья решения (decision trees) являются одним из наиболее популярных подходов к решению задач Data Mining. Они создают иерархическую структуру классифицирующих правил типа "ЕСЛИ... ТО...", имеющую вид дерева (это похоже на определитель видов из ботаники или зоологии). Для того чтобы решить, к какому классу отнести некоторый объект или ситуацию, требуется ответить на вопросы, стоящие в узлах этого дерева, начиная с его корня. Вопросы имеют вид "значение параметра А больше х?". Если ответ положительный, осуществляется переход к правому узлу следующего уровня, если отрицательный — то к левому узлу; затем снова следует вопрос, связанный с соответствующим узлом.

Популярность подхода связана с наглядностью и понятностью. Но очень остро для деревьев решений стоит проблема значимости. Дело в том, что отдельным узлам на каждом новом построенном уровне дерева соответствует все меньшее и меньшее число записей данных — дерево дробит данные на большое количество частных случаев. Чем больше этих частных случаев, чем меньше обучающих примеров попадает в каждый такой частный случай, тем менее уверенной становится их классификация. Если построенное дерево слишком "кустистое" — состоит из неоправданно большого числа мелких веточек, оно не будет давать статистически обоснованных ответов. Как показывает практика, в большинстве систем, использующих деревья решений, эта проблема не находит удовлетворительного решения. Кроме того, общеизвестно, и это легко показать, что деревья решений дают полезные результаты только в случае независимых признаков. В противном случае они лишь создают иллюзию логического вывода.

Эволюционное программирование

Логическим продолжением вышесказанного в отношении деревьев решений является метод эволюционного программирования. Проиллюстрируем современное состояние **эволюционного программирования** на примере системы PolyAnalyst. В данной системе гипотезы о виде зависимости целевой переменной от других переменных формулируются в виде программ на некотором внутреннем языке программирования. Процесс построения программ строится как эволюция в мире программ (этим подход немного похож на генетические алгоритмы). Когда система находит программу, достаточно точно выражающую искомую зависимость, она начинает вносить в нее небольшие модификации и отбирает среди построенных таким образом дочерних программ те, которые повышают точность. Таким образом система "выращивает" несколько генетических линий программ, которые конкурируют между собой в точности выражения искомой зависимости. Специальный транслирующий модуль системы PolyAnalyst переводит найденные зависимости с внутреннего языка системы на понятный пользователю язык (математические формулы, таблицы и пр.), делая их легкодоступными. Для того чтобы сделать полученные результаты еще понятнее для пользователя - не математика, имеется богатый арсенал разнообразных средств визуализации обнаруживаемых зависимостей. Для контроля статистической значимости выводимых зависимостей применяется набор современных методов, например рандомизированное тестирование.

Строго говоря, Data Mining — далеко не основная область применения **генетических алгоритмов.** Их нужно рассматривать скорее как мощное средство решения разнообразных комбинаторных задач и задач оптимизации. Тем не менее генетические алгоритмы вошли сейчас в стандартный инструментарий методов Data Mining, поэтому они и включены в данный обзор.

Пусть необходимо найти решение задачи, наиболее оптимальное с точки зрения некоторого критерия. Пусть каждое решение полностью описывается некоторым набором чисел или величин нечисловой природы. Скажем, если надо выбрать совокупность фиксированного числа параметров рынка, наиболее выражено влияющих на его динамику, это будет набор имен этих параметров. Об этом наборе можно говорить как о совокупности хромосом, определяющих качества индивида — данного решения поставленной задачи. Значения параметров, определяющих решение, будут тогда называться генами. Поиск оптимального решения при этом похож на эволюцию популяции индивидов, представленных их наборами хромосом. В этой эволюции действуют три механизма: отбор сильнейших — наборов хромосом, которым соответствуют наиболее оптимальные решения; скрещивание — производство новых индивидов при помощи смешивания хромосомных наборов отобранных индивидов; и мутации ≈ случайные изменения генов у некоторых индивидов популяции. В результате смены поколений, в конце концов, вырабатывается такое решение поставленной задачи, которое уже не может быть далее улучшено.

Генетические алгоритмы имеют ряд недостатков. Критерий отбора хромосом и сама процедура являются эвристическими и далеко не гарантируют нахождения «лучшего» решения. Как и в реальной жизни, эволюцию может «заклинить» на какой-либо непродуктивной ветви. И, наоборот, можно привести примеры, как два неперспективных родителя, которые будут исключены из эволюции генетическим алгоритмом, оказываются способными произвести высокоэффективного потомка. Это особенно становится заметно при решении высокоразмерных задач со сложными внутренними связями.

Решения поиска. Оптимизационные решения. Алгоритмы ограниченного перебора

Алгоритмы ограниченного перебора были предложены в середине 60-х годов М.М. Бонгардом для поиска логических закономерностей в данных. С тех пор они продемонстрировали свою эффективность при решении множества задач из самых различных областей, в том числе в задачах моделирования информационно-поисковых систем и поисковых подсистем сложных комплексных ИС.

Эти алгоритмы вычисляют частоты комбинаций простых логических событий в подгруппах данных. Примеры простых логических событий: X = a; X < a; X > a; a < X < b и др., где X — какой либо параметр, a и b — константы. Ограничением служит длина комбинации простых логических событий (у Бонгарда она была равна 3). На основании анализа вычисленных частот делается заключение о полезности той или иной комбинации для установления ассоциации в данных, для классификации, прогнозирования и т.п..

Описания распределенных систем через множество процессов

Процесс описывает возможное функционирование распределенной системы. Распределенные системы обладают многими ходами работы. Абстрактное описание динамического поведения распределенной системы может быть дано путем задания множеств процессов. Существует ряд формализмов для описания систем и ходов работ. Тремя представителями таких формализмов являются:

- сети Петри, графический метод описания;
- агенты, формальный язык описания;
- формулы логики предикатов для описания хода работы;
- другие хорошо и отчасти формализованные методы, языки и модели.

Другие методы описания для распределенных систем и их поведения дают языки программирования и программы. Если некая программа выполнятся системой, то протекает некоторый процесс, который складывается из множества событий, соответствующих действиям при выполнении программы. Программа также описывает операционным способом процесс, причем последовательные программы описывают последовательные процессы. Для описания общих, не последовательных процессов языковых средств обычных, последовательных языков программирования оказывается недостаточно. Чтобы разрешить проблему, возникающую при описании параллельно выполняющихся систем программ, например, для параллельной композиции, коммуникации между частями программы и синхронизируемого доступа к общесистемному информационному пространству – базам данных, применяют дополнительные языковые средства. Часто ИС содержит собственный «внутренний» язык, поддерживающий систему и описывающий ее действия.

Мера сложности информационных систем - сложные модели ИС

Понятие вычислимости (способности ИС своими действиями привести к результату) непосредственно зависит от выбранного понятия алгоритма действий системы, хотя в классической теории информатики считают, что все различные предложенные понятия алгоритма при определенных условиях могут быть эквивалентными. Сложность какой-либо задачи также зависит от выбранного понятия алгоритма, однако, здесь различные формализации этого понятия (детерминированные, недетерминированные алгоритмы) влекут богатство понятия **сложности систем**.

Существуют простые единицы измерения для *вы*числительных затрат и потребностей в памяти рекурсивных вычислительных предписаний. В теоретической информатике при этом понятие сложности общепризнанно основывается на концепции машины Тьюринга (Т-машины). В информатике утверждается также, что реальные ЭВМ по меньшей мере похожи на Т-машины - только в каждой ячейке памяти можно хранить конечное множество различных «знаков» (двоичные слова), а *для* процессора существует только конечное число различных состояний. Аналогично в больших распределенных информационных системах в каждом модуле можно хранить конечное число элементов информации, а для базы данных ИС существует конечное рано или поздно ограниченное число различных состояний; при этом в самом общем виде по своей структуре обобщенная большая ИС похожа на Т-машину.

Временная сложность

Чтобы получить реалистичные и стабильные рассмотрения сложности, рассмотрим большую информационную Т-систему с К-подсистемами, видя в ней некие условные аналогии с Т-машиной с k-лентами. Для каждой из k имеется своя СУБД.

На каждом шаге функционирования (например в ИС, поиск информационного элемента в К и ответ на единичный запрос) в зависимости от внутреннего состояния системы происходит как минимум единичное смещение этого состояния. Для каждого входного запроса (команды) той или иной длины система производит определенное число вычислений (действий). Число шагов в этих действиях обозначает длину соответствующих последовательностей шагов вычислений. Система может оказаться ограниченной во времени Т-системой, что означает существование ветви вычислений (поиска, действий), требующей определенное ограниченное число шагов вычислений для входа в систему запроса в самом общем недетерминированном случае. Это число шагов отражает сложность системы: чем оно менее, тем вероятнее видеть систему уменьшающейся сложности. Способность реализации как можно большего числа шагов в один и тот же отрезок времени есть характеристика мощности системы. Следовательно, сложность и мощность, способная освоить эту сложность, есть взаимосвязанные, в общем случае однонаправленные понятия в оценке уровня ИС.

Т-системы с множеством подсистем, а также распределенные ИС (если пренебречь воздействием на сложность ИС сетевых аспектов) в смысле вычислимости не мощнее систем, не имеющих в своей структуре других входящих систем, подсистем и распределенности, однако, они могут дать более реалистичные оценки затрат времени и мощности ИС, в известном смысле осуществляя полное или частичное горячее резервирование модулей системы по отношению к друг другу.

Если задача Р может быть решена Т-системой и эта система является T(n)-ограниченной по времени, то задача Р также называется T(n)-ограниченной по времени. По этому определению сложность системы тесно связана с представлением входа/выхода этой системы при рассмотрении ее в виде некоего «черного ящика».

Определенные задачи, хотя теоретически и являются вычислимыми, но имеют такую сложность, что *практически* они не могут быть вычислены с использованием конкретно взятой системы.

Следовательно, в связи с вычислимостью (то есть достигаемостью решения в пределах допустимой мощности ИС) наиболее реальный интерес представляют детерминированные Т-системы. Для вопросов сложности имеют значение и недетерминированные Т-системы, поскольку они характеризуют определенные классы сложностей. Для этого необходимо определить, что значит, для недетерминированной Т-системы принимать (воспринимать, допускать) запрос, команду. Т-система

принимает запрос или команду, если она приходит в заключительное состояние по некоторому пути вычисления, что означает завершенные, истинные, удовлетворяющие пользователя действия и ответы на запрос или команду в окончательном виде (то есть заключительное состояние).

Концепцию временной ограниченности можно обобщить и на недетерминированные Т-системы. Так, недетерминированная Т-система является *недетерминированной Т неограниченной по времени,* если эта оценка имеет место для каждого запроса, каждой входящей команде w при их недетерминированной обработке (то есть для каждого w существует допускающая его ветвь вычисления, по времени не превосходящего T(n)). Здесь уместно заметить, что по курсу информатики в машине Тьюринга аналогично определяется и *недетерминированная S(n) ленточная ограниченность.*

Полиномиальная и недетерминированная полиномиальная временная сложность систем. NP-полные проблемы

Множество задач, которые могут быть решены за полиномиально-ограниченное время при поддержке детерминированных систем, обозначим через Р:

$$P == \mathbf{u}\ DTIME\ (n^{\wedge}) \qquad [\wedge=i].$$
$$I > l \quad \text{Задачи из P}$$

множества называют эффективно решаемыми.

Ряд важных задач могут быть достаточно просто, эффективно решены при поддержке недетерминированных полиномиально-ограниченных по времени Т-систем. Класс этих задач, относящийся к так называемым NP-полным проблемам, обозначим через NP:

$$NP = \mathbf{u}\ NTIME\ (n^{\wedge}).$$
$$i > 1$$

Пример (задача из NP). Следующая задача входит в класс NP. Для заданного графа с k вершинами определить, имеется ли циклически замкнутая трасса, на которой лежат все вершины графа. Такая трасса называется *направленным обходом Гамильтона.*

Концепция недетерминированных систем является формулировкой в теории автоматов одного из классов алгоритмов, которые работают с методами поиска (например, применительно к Т-машинам в классической информатике). Любой недетерминированный алгоритм можно считать спецификацией задачи поиска, то есть свойственным информационно-поисковым системам и подсистемам мегасистем.

NP-полные проблемы (см. УМК каф. ТИССУ МИРЭА по курсу информатики и теории информации) требуют обращения с помощью алгоритмов с экспоненциальной сложностью. Это означает такие большие затраты на вычисления, которые при определенных постановках задач находятся на границе того, что практически может быть осуществлено, а в некоторых случаях даже за пределами этих возможностей. Такая ситуация более, чем ощутима, в больших, глобальных, особенно распределенных ИС с мало предсказуемой ресурсной емкостью системы в целом. Поэтому особенно важно для решения NP-проблем иметь возможно более эффективные алгоритмы.

NP-полные проблемы всегда можно понимать как **проблемы поиска.** Недетерминированной информационной системе мы можем предписать конечное дерево с ограниченной числом v степенью разветвления и с ограниченной полиномом p(n) его высотой. Детерминированный алгоритм должен по-существу обойти

(просмотреть) это дерево, содержащее 0(vP(^) вершин. Часто NP-полные проблемы представляют собой задачу оптимизации типа «найти кратчайший путь», «найти оптимальный ход» и т. п.

Для эффективной обработки - в рамках экспоненциальной сложности - для NP-полных проблем можно выбирать, в частности, следующие методы.

(1) ***Branch and Bound (искусный бэктрекинг)***. Поиск в пространстве решений организуется целенаправленно:

- в первую очередь проходятся легко вычисляемые ветви;
- своевременно распознанные как неинтересные ветви обрезаются и не проходятся полностью.

К тому же, в частности, пытаются пространство решений трактовать как многомерное пространство. Для многих задач это вытекает уже из самих их постановок (найди вектор определенной длины, найди множество определенной мощности).

(2) ***Приближенные методы***. В ряде постановок задач создания и реализации модели ИС требуется найти оптимальное решение (например, кратчайший путь). Однако иногда достаточно иметь примерно оптимальное решение, благодаря чему может быть редуцирована сложность задачи. В таких случаях требуется найти компромисс между затратами и неполной оптимальностью.

(3) ***Динамическое программирование.*** При динамическом программировании число подлежащих исследованию кандидатов при наивном поиске уменьшается за счет более искусного образа действий. Если, например, в дереве поиска встречаются идентичные кандидаты в различных его поддеревьях, то часто оказывается более эффективным (относительно времени вычислений и времени сетевых операций и ожиданий, однако при дополнительной потребности в памяти) запоминать в определенных таблицах уже обработанные задачи и в случае необходимости использовать полученные ранее результаты вычислений.

(4) ***Вероятностные алгоритмы.*** Эти алгоритмы сознательно используют случайные величины, порождаемые генератором случайных чисел. При этом имеется два варианта («а» и «б»):

(а) алгоритмы, которые дают корректный результат только с определенной вероятностью (которая при соответствующих затратах может быть выдержана сколь угодно близкой к единице);

(б) алгоритмы, которые завершаются в заданные границы времени только с определенной вероятностью.

(5) ***Ограничение класса задач.*** В некоторых применениях можно так ограничить рассматриваемый класс задач, что задачи из этого ограниченного класса допускают полиномиальную обработку.

(6) ***Эвристические методы.*** Как раз в случае NP-полных задач поиска в условиях дальнейшего расширения (масштабирования) распределенных мега БД по экспоненциальногому закону мы часто находим эвристические методы, которые удивительно хорошо работают, хотя неясно почему, и относительно этого нет каких-либо точных высказываний.

Для соответствующей работы с NP-трудными проблемами от программиста требуется особое искусство.

Искусный просмотр больших древовидных структур, оптимизационные решения с использованием динамического программирования и Гриди-алгоритмов

Рассмотрим теперь специальную постановку задачи. Пусть дано множество M в качестве листьев дерева и функция

f: M -» **O,** где O - множество с линейным порядком. Требуется найти элемент x из M, для которого f(x) минимальна. Точнее говоря, ищется элемент x **е** M, для которого справедливо

f(x) - min {f(y): yeM}.

Обратим внимание, что упомянутые выше NP-полные проблемы могут быть приведены к этой форме. Эффективный поиск в пространстве поиска состоит, в частности, в комбинации порождения множества и исследования его элементов при исключении запоминания всего множества. Если множество порождается рекурсивно, то принцип порождения определяет на дереве вызовов древовидное упорядочение элементов рассматриваемого множества. Схематически такую проблему можно представить неким заданным алгоритмом. Алгоритм рекурсивно порождает множество, в котором должен производиться поиск решения, а другой сопутствующий алгоритм берет это множество в качестве входа и ищет в нем элемент, для которого функция f принимает наименьшее или удовлетворительное значение. При этом заданы: предикат, который проверяет, является ли исследуемый элемент x нужным элементом просматриваемого массива; специальные функции, которые представляют альтернативы для вывода элементов, и, может быть специальная функция, обозначающая степень разветвления поиска или обработки. Эти функции соответствуют принятию решения о выборе в недетерминированном дереве вычислений Т-машины, а равно и Т-системы при ограничениях и допущениях, в основном схожих с применимыми к Т-машине.

Многие проблемы могут быть представлены по этой схеме. Сюда же относится и проблема выполнимости.

Рекурсивная структура функции вызова (поиска) дает возможность улучшить эффективность алгоритма поиска (выполняемость), если применить:

- умелое отсечение трасс, которые не обещают успеха (например, а/р-отсечение, бэктрекинг в Т-машинах);
- умелый выбор порядка просмотра (greedy и т.д.).

Если умело выбрать порядок просмотра, то ускоренно находится минимальное решение и своевременно отбрасываются не минимальные трассы.

Выбор для программы метода улучшения эффективности при решении сложных проблем требует умения, знаний и опыта. Полезен тщательный анализ проблемы. Чем больше мы знаем о проблеме и ее свойствах, тем лучше мы может применить оптимизации.

К числу высокоэффективных методов оптимизации относят динамическое программирование, гриди алгоритмы (англ. greedy - «жадный»), предназначенные решать задачу глобальной оптимизации с помощью локальных критериев оптимизации. Типичным примером является следующий алгоритм , по которому находится длина кратчайшего пути в графе. В этом алгоритме , исходя из начальной вершины, строится путь таким образом, что по мере надобности выбираются вершины

с минимальным рассеянием от её вершины Т.

Пример (алгоритм Дейкстры для вычисления кратчайшего пути в графе с нагруженными ребрами). Пусть дано множество вершин

V={l,...,n} и расстояния между ними d:VxV^Nu{oo}.

Предположим, что d есть всюду определенная функция. Если не существует ребра от вершины X к вершине У, то это выражается через:

d(x,y) = **oo**

Для вычисления длины кратчайшего пути служит следующее предписание (тип **enat** содержит натуральные числа и **oo**) **fct** dijkstra = (set **node** m, **node** x, **node** y) **enat**

if m = 0

then d(x, y)

else node w == **some node** z: z **e** m **л V node** b: b **e** m => d(x, z) < d(x,b); min(diJkstra(m\{w}, x, y), d(x, w)+dijkstra(m\{w},w,y) Используется следующий инициализирующий вызов: dijkstra(V, x, y). Корректность этого предписания можно показать следующим образом: для любого множества S с V вызов dijkstra (S, x, y) дает длину кратчайшего пути от х к у - с внутренними вершинами только из множества S. Это утверждение доказывается индукцией по n=|S|

Для n = О утверждение очевидно. Пусть утверждение справедливо для п; для любого множества S с |s|= n+1 пусть w есть вершина из S с наименьшим удалением от x: V *be* S: d(x, w) < d(x, b). Пусть x^ pi...-» pk^ y есть путь кратчайшей длины от х к у с внутренними вершинами pi **e** S для всех i, 1 < i < k. Или справедливо w **g** {pi, ..., pk}, или справедливо pi = we I <, i ^ k. Но тогда должно быть i = 1, так как в противном случае путь х -» pi -»... -» pk -» У был бы короче. Итак, справедливо: кратчайший путь не содержит вершины w, или он содержит w в качестве первой внутренней вершины. Так что мы получаем длину кратчайшего пути как минимум из длин кратчайшего пути, который не содержит w, и кратчайшего пути, который содержит w в качестве первой внутренней вершины.

Заметим, однако, что гриди-принцип не всегда применим, поскольку локальная оптимизация не всегда приводит к глобальной оптимизации. Пример тому мы получим, если попытаемся идею алгоритма Дейкстры перенести на целочисленно размеченные графы. Здесь могут существовать пути с отрицательными значениями, так что обход через далеко удаленные вершины может, тем не менее, вести к более короткому пути.

Работа с задачами большой сложности требует глубокого анализа постановки задачи. Путем упрощения постановки задачи, например через ограничения на область значений параметров, в некоторых случаях можно сделать применимыми менее сложные алгоритмы. Важно также, сколь многочисленны на самом деле входные данные для рассматриваемой задачи. При небольшой размерности могут непосредственно решаться (англ. brute force) даже NP-полные проблемы.

Методы описаний и формализмы спецификаций программных средств и систем

Существует много различных стилей и формализмов для описания требований, данных и алгоритмов, которые находят применение при проектировании, разработке и анализе программных средств и систем. Различные формализмы по-разному удовлетворяют общим требованиям к нотации, таким, как: • простота чтения и понимания,

- легкость овладения,
- мощность (границы возможностей нотации),
- выполнимость,
- эффективность выполнения.

Какие веса придать этим критериям, сильно зависит от специфики приложений.

Для формулирования эффективных алгоритмов, безусловно, требуется точно установить, что должно быть вычислено, не вникая сначала в то, *как* (по какому алгоритму) это будет вычисляться. Мы говорим о *спецификации задачи* или о *спецификации требований.*

Абстракция в спецификации

В программировании оказывается полезным различать *абстрактные вычислительные структуры* (называемые также *абстрактными типами данных)* и *структуры данных.* При этом, точнее говоря, речь идет исключительно о разных взглядах на типы и на функции, имеющиеся для них в распоряжении.

Абстрактные структуры данных представляют вид доступа для типа s или для семейства типов. При этом устанавливается, какие основные операции имеются в распоряжении для элементов данных этого типа. Для этого в сигнатуре задаются доступные для использования функции и их функциональности. Относительно символов функций делается различие между

- селекторами для доступа к составным частям элемента данных,
- функциями опроса для установления определенных свойств данных,
- дискриминаторами,
- функциями-конструкторами для построения элементов данных.

Наиболее прост частный случай, когда задается вычислительная структура для описания структуры доступа только для одного типа s. Конечно, эти вычислительные структуры опираются на другие типы, которые предполагаются заданными. Функции-конструкторы в качестве типа результата принципиально имеют тип s. Если аргументы функции-конструктора также имеют тип s, то описываемый тип будем называть рекурсивным. При рекурсивных типах также и определенные функции-селекторы имеют тип результата s.

Если в связи с типом s говорится о структуре данных, то тем самым имеется в виду внутреннее строение элементов данных типа s. Это определяет, каким образом структурируются элементы данных внутри себя и как конкретно они реализуются в памяти ЭВМ.

Различение аспектов *реализации* (англ. Glass Box View) и *доступа* (англ. Black Box View) является существенным для разработки программы. Аспект доступа определяет интерфейс (разрез) для типа или вычислительной структуры. Этого разреза достаточно для корректного использования типа и имеющихся для него операций. Для оценки эффективности нужны только данные о сложности по времени и по памяти используемых операций.

Аспект реализации касается лишь реализатора типа и содержит в себе все детали реализации. Аспект доступа представляет собой *абстракцию* аспекта реализации. Существует много структур данных для реализации одних и тех же абстрактных вычислительных структур. При использовании вычислительной структуры пользователю нужно знать лишь аспект доступа, реализация же может быть скрыта от него (англ. information hiding). Это имеет решающие преимущества:

- пользователь не обязан знать порой сложные, трудно просматриваемые детали реализации вычислительной структуры (информационной системы), ему

достаточно знать лишь ее абстрактный аспект доступа. Этот аспект служит для документирования;

- реализация может производиться параллельно с программированием, при котором используется эта вычислительная структура. Основой для взаимопонимания служит аспект доступа;
- реализация может быть изменена, и пользователь может даже не знать об этом, если только сохраняется аспект доступа.

Центральным для этого метода разделения аспектов доступа и реализации является понятие точки разреза, или ***интерфейса*** (англ. interface). Последовательное разделение этих аспектов является показателем хорошего стиля разработки программного продукта, особенно при создании сложных программных систем (информационных систем). Этот принцип используется не только применительно к структурам данных - он может быть перенесен и на другие программные единицы, такие, как процедуры, модули и классы.

Спецификация абстрактных, вычислительных структур

При разработке программных систем одной из важнейших задач является выбор и спецификация основных вычислительных структур. Для описания вычислительных структур предлагаются следующие возможности:

- моделирование с помощью заданных структур (способ, ориентированный на модели), например через множества, отношения, абстрактные машины;
- характеризация свойств через логику предикатов или через законы равенств ("алгебраическая спецификация").

Ниже кратко обрисована техника алгебраических спецификаций, так как она особенно хорошо подходит для описания аспектов интерфейса. Вычислительную структуру можно алгебраически специфицировать путем:

(1) задания сигнатуры,

(2) задания законов, характеризующих отображения.

Дадим несколько примеров алгебраических спецификаций. В алгебраической спецификации мы описываем абстрактную вычислительную структуру через ее сигнатуру и ее законы.

Спецификация функций

Одной из самых элементарных концепций математики являются функции и абстракции функций. Программы (вычислительные предписания) в простейшем случае реализуют (вычисляют) определенные функции. Тем самым спецификация программ соответствует спецификации функций.

Чтобы специфицировать функцию f, сначала задается ее функциональность. Например, пишется:

Fct f= (s1,…..,sn) Sn+1. Для заданной Σ -
вычислительной структуры могут быть специфицированы с помощью равенств.

При этом задается некоторое количество правил, которые устанавливают требования к f, но не обязательно описывают функцию однозначно. В этом случае говорят, что функция f не полностью специфицирована.

Обращается внимание на то, что для многих постановок задач вовсе не обязательно, чтобы спецификация задавала функцию однозначно. Здесь говорится о

степенях свободы в спецификации, о неполной спецификации или об открытости принятия решений по проекту.

Базы данных и информационные системы

Одной из центральных задач информатики является длительное хранение данных и контролируемый доступ к ним. Эти задачи в различных источниках имеют весьма многоплановое толкование и обозначение. Так, в публикациях специалиста с мировым именем М.Броя обсуждение этих задач ведется под терминами (компьютерные) *информационные системы* и *базы данных (БД).*

Аспекты баз данных в рамках задачи управления информацией в информационных системах охватывают описание имеющихся данных, управление ими как наиболее важной частью информационной системы и доступ к ним.

Информационная система содержит информацию, необходимую для управления и контроля задачами пользователя. При разработке какой-либо базы данных для определенного применения необходимо построить *модель данных,* в которой представлены все виды информации, которые будут храниться в базе данных. Но модель данных, независимо от задачи описания базы данных, также должна быть применима для всех релевантных данных для определенного применения.

Моделирование отношений сущность/связь

Модель данных можно задать с помощью спецификации вычислительной структуры. В народнохозяйственных применениях информатики встречаются большие множества однотипных данных. Примерами являются данные о наличии клиентов, товаров или сотрудников предприятия. Такие данные тоже можно охватить через вычислительные структуры. Однако для таких больших множеств данных удобнее применять специальные способы описания. Общим методом описания для информационных систем и баз данных, и в частности для моделирования данных с большим их числом, является так называемая *сущность/связь-модель* (англ. Entity/Relationship Model), или, короче, E/R-модель. При этом наличные данные представляются множествами элементов данных, а связи между основными элементами представляются через отношения.

При проектировании информационной системы создание модели нужно для того, чтобы охватить задачи управления, лежащие в основе системы. В моделировании данных охватывается вся информация, имеющая значение для применений.

Типичным образом модели данных и базы данных содержат определенные основные единицы информации, называемые *сущностями.* Каждая сущность имеет обозначение и представляет множество элементов (записи) определенного типа. Запись называется проявлением (воплощением) сущности. Тип определяет совокупность всех записей, которые могут рассматриваться как сущность. Между сущностями имеются определенные связи, называемые отношениями, которые устанавливают "семантические связи" между сущностями.

Диаграммы сущность/связь

На практике для описания E/R-моделей данных часто используют графическое изображение, поскольку оно более наглядно и понятно. Это особенно важно именно при моделировании данных, где должно быть особенно полное понимание между

информатиками и экспертами по применению. Это также имеет место и для применений баз данных.

Для графического представления E/R-моделей и баз данных используются E/R-диаграммы (англ. Entity/Relationship Diagram). Помещенный ниже рисунок дает пример такой диаграммы. В E/R-диаграмме множества сущностей изображаются прямоугольниками, а связи - ромбами. Часто добавляются дополнительные данные, которые описывают, например, между сколькими сущностями может иметь место определенное отношение, - здесь говорится о *количественных указаниях.*

Связь состоит из обозначения и указания фигурирующих сущностей, а также типа связи. В соответствии с числом вовлеченных сущностей говорится о n-местном отношении (или также об отношении степени n).

Обычно связи между сущностями обусловлены их свойствами, используемыми в данной области применений. Например, один сотрудник может участвовать в нескольких проектах, но может принадлежать только одному из отделов предприятия.

Многие ER-диаграммы позволяют характеризовать такие свойства отношений взаимосвязи, как:

- во всех допустимых состояниях сущностей элементы присутствуют в кортежах отношения в количестве, лежащем в некотором заданном интервале;
- отношение является однозначным справа, однозначным слева, взаимно однозначным (n:1-, 1:n- и 1:1-взаимосвязи).

В зависимости от мощности языка описаний для модели баз данных могут быть выражены более или менее сложные взаимосвязи.

Атрибуты могут быть также определены и для связей между множествами сущностей. Это значит, что для каждого элемента в связи (каждого элемента кортежа) специфицируются дополнительные значения.

Пример (атрибуты между взаимосвязями). Взаимосвязь "есть_руководитель_проекта" могла бы иметь атрибут "с: дата" и "предшественник: сотрудник".

Во многих приложениях встречаются специальные взаимосвязи, как, например, "есть_тот-то"-взаимосвязь, которая допускает классификацию сущностей. Если, например, существуют сущности "руководитель отдела" и "сотрудник", то между ними может иметь место взаимосвязь "есть_тот-то". Эта взаимосвязь разрешает перенесение атрибутов. Если какой-то руководитель отдела является сотрудником предприятия, то все атрибуты сотрудника наследуются и руководителем отдела (переносятся на него). Определенные конструкции "представления знаний", такие, как "фреймы" и формы "объектно-ориентированного" моделирования, используют подобного рода концепции наследования, иерархии и образования классов в качестве базы для спецификаций. В конце данной главы мы вернемся к этим вопросам под термином объектной ориентации.

Формы иерархизации и образования классов являются типичными для ряда областей науки, например, зоологии. Соразмерное применение этого подхода делает возможным далеко идущую структуризацию задач управления информацией.

К применению систем баз данных

В задачах управления информацией можно выделить следующие аспекты архитектуры баз данных:

- концептуальный аспект (логическая модель вовлекаемых баз данных);

- физический аспект (внутреннее представление баз данных в памяти и относящиеся к этому процедуры доступа);
- внешний аспект (возможность доступа для пользователя и, соответственно, для пользовательских программ).

Системы баз данных могут использоваться либо при непосредственных запросах и быть интерактивными, либо через определенные программы, которые вызываются с помощью процедур службы базы данных.

Система управления базой данных

Связь между описанными аспектами осуществляется системой управления базой данных. В задачи этой системы входит:

- обеспечение условий целостности,
- координация параллельно протекающих выполнении запросов пользователей,
- защита данных от неправомерного доступа и их сохранность от разрушения.

Запросы к базам данных и их изменение

Заданием E/R-модели характеризуется и множество возможных состояний базы данных. В каждый момент своей жизни база данных обладает некоторым состоянием. С помощью определенных операций можно осуществить опрос состояния базы и, соответственно, изменить это состояние.

Как правило, запросы к базе данных и изменения се состояния выражаются не в аспекте значений отдельных атрибутов, а через мощные операции над сущностями и связями. Типичный запрос к базе данных заключался бы в поиске: "найди фамилии и адреса всех сотрудников отделов x, y, z по специальностям a, b, c".

Такой запрос доставляет список, представляющий собой одноместное отношение, который затем может быть еще обработан с определенной точки зрения (например, просмотрен или распечатан). Языки запросов к базам данных предусматривают вполне определенный формат и логическую форму.

Аналогичным образом могут быть сделаны и изменения состояний и баз данных. Эти изменения могут состоять из следующих действий:

- внести новые сущности, воплощения и новые элементы связей,
- удалить определенные данные,
- изменить определенные компоненты.

Во всех указанных случаях могут возникнуть проблемы из-за нарушения условий целостности. Задача системы управления базой данных состоит в том, чтобы выявлять такие ситуации, указывать на них и в конце концов обеспечить целостность базы данных.

Для формулирования запросов к базе данных и требований на изменение ее содержимого базы данных предоставляют пользователю особый интерфейс. Он состоит из команд (как выбор, исключение или добавление данных) и из (ограниченных) возможностей задавать логические формулы для образования подмножеств данных. Для этого используются также некоторые аспекты алгебры отношений (реляционной алгебры), такие, как:

- проекция (вычеркивание определенных компонент кортежей в отношении),
- соединение (произведение отношений, причем делается соединение кортежей, совпадающих в общих атрибутах),
- селекция (выбор кортежей, удовлетворяющих заданному условию относительно значений атрибутов).

В случае языка запросов на основе исчисления отношений пользователь задает отношения с помощью логических выражений и соответствующим образом комбинирует их. При этом иногда допускается (в ограниченной форме) использование кванторов существования и всеобщности. Изменения же в состоянии базы данных делаются, как правило, поэлементно.

Имеется много самых различных концепций для языков запросов к базам данных и их изменения. Практически важно при этом, чтобы эти концепции укладывались в языки программирования, так как это допускает управление базами данных из программ.

Логическое программирование

Если в предыдущих разделах было описание только задач и вычислительных структур, но не алгоритмов, то теперь будет рассмотрен вопрос о том, насколько подходящими являются формализмы логики непосредственно для алгоритмических вычислений. Здесь это обсуждено на направлении, которому в предыдущие годы было уделено большое внимание, так называемого *логического программирования.*

В логическом программировании предикаты над заданной вычислительной структурой (как истинностные значения, числа и последовательности) описываются логическими формулами очень простого вида. Тогда запросы к логической программе можно сформулировать, основываясь на специфицированных предикатах, путем задания формул со свободными идентификаторами. Тогда система выполнения логической программы систематически ищет решение для запроса. Решение состоит в отыскании подстановок для свободных идентификаторов в запросе таким образом, что так модифицированные запросы логически следуют из спецификаций для предикатов. Этот способ действий используется в языке программирования Пролог, который нашел широкое практическое применение.

Решение задач в логическом программировании

Логическая программа состоит из формул вида (пусть С и В, - логические выражения, в простейшем случае **true и false,** а также предикаты на константах), называемых "условиями Хорна" (англ. "Нот - Klausein"):

$$B \Leftarrow B1 \wedge B2 \wedge \ldots \wedge Bn.$$

Выражения Bi называются *предпосылками* или *подцелями* (англ. subgools), а С называется *заключением.* Множество подцелей может также быть пустым или состоять только из **true;** тогда формулу мы называем *фактом,* а другие условия - *правилами.*

При формулировании Hom-Klauseln-программ будут использовать следующие соглашения. Идентификаторы будут записывать большими буквами, а обозначения функций (включая нуль-местные функции для представления констант) - малыми.

Унификация

При выполнении логических программ путем применения правил преобразования типичными являются операции над термами. Примером этому служит вопрос о применимости некоторого правила. На этот вопрос можно ответить с помощью методики *унификации.*

Унификация является операцией на множестве термов. Результатом унификации является подстановка, которая для каждого терма заданного множества в качестве результата выдает один и тот же терм. Такая подстановка называется

унификатором. Но, не для каждого множества существует такого рода унификатор - в этом случае множество называется *не унифицируемым.*

Объектно-ориентированное программирование

Объектная ориентированность пытается претворить в жизнь общие принципы разработки программного обеспечения и программ с помощью учета конкретных способов разработки и использования вполне определенных средств описания, что чрезвычайно актуально для информационных систем, поддерживающих базы знаний в образовании и науке. Исторически объектная ориентированность восходит к языкам программирования Simula-67 и Smalltalk. В настоящее время одним из наиболее употребительных на практике языков этого типа является C++.

Для обычных фаз разработки программ характерны следующие моменты объектной ориентированности:

- объектно-ориентированный анализ,
- объектно-ориентированный проект,
- объектно-ориентированное программирование.

Это образование понятий проистекает из классического разбиения процесса разработки программного обеспечения на отдельные фазы. На *фазе анализа* выясняется специфика применения с целью охватить требования к разрабатываемой программной системе. На *фазе проектирования* вырабатывается структура системы (ее архитектура). На *фазе реализации* программируются части системы, предусмотренные в проекте. В дальнейшем будет рассмотрено, прежде всего, объектно-ориентированное программирование (00-программирование).

При 00-программировании опираются на следующие концепции:

- инкапсуляция данных,
- классы и наследование,
- объекты и воплощение (динамическое создание новых объектов),
- вызов методов и обмен сообщениями.

Эти концепции далее будут объяснены. Объектная ориентированность, между прочим, является попыткой внедрить более эффективные и адекватные методы описания, модели и инструментарий в технологию программирования.

Объектная ориентированность нацелена на то, чтобы воплотить общие принципы проектирования программных систем и программ в конкретные методики разработки и моделирования со специфичной нотационной поддержкой. 00-программирование при этом делает акцент на следующие цели:

- модульность,
- унификация и систематика,
- модифицируемость и гибкость,
- повторная применимость (переносимость).

Принцип модульности преследует цель строить систему из замкнутых частей ("строительных кирпичиков"), для использования которых не нужно знать деталей их реализации, а достаточно знать лишь их действие на точках разреза (т. е. интерфейс). Детали реализации скрываются (англ. information hiding) от пользователя. Благодаря этому пользователь модуля не обременяется ненужными для него деталями и не может их использовать при применении модуля. Тем самым детали реализации модуля могут быть изменены без того, чтобы возникали какие-либо проблемы при его использовании, если только сохраняются требования спецификации на интерфейс. Это снова соответствует уже обсуждавшемуся принципу делать различие между аспектами доступа и реализации.

Принципы унификации и систематики состоят в том, что программная система строится из однообразных "кирпичей" (классов). По принципу модифицируемости и гибкости система строится таким образом, чтобы ее можно было легко приспосабливать к изменяющимся требованиям.

Для обеспечения повторной применимости систему стараются построить из модулей таким образом, чтобы ее реализацию можно было перенять из предыдущих проектов или же эту реализацию можно было применить в более поздних проектах. Очевидно, что эти принципы тесно переплетаются между собой.

Можно различать следующие два основных направления 00-программирования:

(а) объектная ориентированность в традициях программной инженерии (как она выступает, например, у Б. Майера или реализуется в языках Smalltalk, Eiffel, а также в вариантах языка C),

(б) объектная ориентированность в традициях "искусственного интеллекта" (проявляющаяся в 00-расширениях языка LISP или во фреймах, в частности при разработке экспертных систем).

В дальнейшем будет ориентация на направление (а), поскольку там более заметную роль играют важные для нас вопросы проектирования и методики.

При концепции объектной ориентированности делается особенное ударение на различие между статическим текстом программы, написанным с помощью классов, и динамикой ее выполнения путем образования воплощений классов в виде объектов.

В подходах к объектной ориентированности существуют заметные различия в моделях протекания процесса вычислений:

- последовательное выполнение (как в Simula, C++, Eiffel, Smalltalk),
- параллельное выполнение (00-SDL, GRAPES).

В 00-языках программирования с последовательной моделью выполнения, как и в классических императивных языках программирования (ЯП), вызовы процедур (которые здесь часто называют вызовом метода) обрабатываются последовательно. При параллельной же модели объекты действуют параллельно и обмениваются сообщениями. Впрочем, языки с параллельной моделью выполнения в настоящее время являются еще предметом исследования. Однако с учетом все возрастающей роли распределенных систем под терминами *вычислительные сети, системы клиент/сервер и распределенные информационные системы* значение таких языков становится все более важным.

Центральными понятиями для объектной ориентированности являются *классы* и *объекты.* Класс понимается как единица описания (аналог аксиоматической спецификации или модуля), состоящая из объявлений программных переменных, обозначений констант (в 00-программировании говорят об *атрибутах класса),* а также функций, процедур (здесь говорится о *методах)* и часто каналов коммуникаций. Таким образом, класс представляет собой совокупность:

- типов (типы данных),
- функций, процедур (или *методов),*
- переменных (или *атрибутов),*
- каналов коммуникаций,
- других классов (при определенных обстоятельствах).

Класс имеет обозначение, которое используется (также как тип) для объявления объектов этого класса.

Объект создается путем воплощения класса, что сравнимо с порождением значения указателя, которое понимается как ссылка на объект. Эта ссылка однозначно идентифицирует объект, и поэтому указатель является идентификатором объекта. С

указателем обращаются как с элементом данных. Его типом является связанный с ним класс.

Класс есть поименованная единица описания. В процессе выполнения 00-программы с помощью этой единицы генерируются объекты данного класса. Поэтому с этим классом во время выполнения программы можно связать множество генерируемых объектов данного класса.

Под термином *объектная ориентированность* между тем охватываются хорошо известные принципы software-инженерии. При этом в основе лежит принцип *модульности,* который подразумевает известные критерии software-систем:

- модульную декомпозицию (модульная разложимость системы на под системы),
- модульную композицию (модульная собираемость системы из подсистем),
- модульную понимаемость (независимая понимаемость подсистемы),
- модульную стабильность (модульная модифицируемость, локальность изменений),
- модульную инкапсуляцию (модульная защита, однозначно установленные права и методы доступа).

Модульность требует особенно тщательной спецификации и описания взаимодействия (интерфейса) между составными частями software-системы. Поэтому формульная спецификация поведения интерфейса представляет особый интерес.

Для образования интерфейса выдвигаются следующие пять принципов, которые сохраняют свою силу и вне объектной ориентации:

- синтаксически ясная и независимая формулируемость единиц (частей системы),
- небольшое число точек разреза (интерфейсов) (адекватная гранулированность проекта системы),
- простота интерфейса (простота точек разреза, их адекватный выбор),
- явное описание интерфейса,
- упрятывание информации о реализации (принцип инкапсуляции).

Честолюбивая цель объектной ориентированности состоит в обеспечении широкого повторного использования программ, проектов, архитектур, спецификаций и концепций с помощью схематизации образа действий. При этом внедряются следующие категории схем:

- схема объявлений типов,
- схема родственных вычислительных структур (типы и алгоритмы),
- схема для вычислительных предписаний.
- На переднем плане стоит осознание структурных и поведенческих общностей с целью:
- независимость представления (упрятывание информации),
- использование общностей родственных единиц.
- Многократное использование поддерживается следующими концепциями:
- перезарузки (англ. overloading) операторов,
- наследования,
- полиморфизма,
- инкапсулированных, параметризуемых, генерируемых структур.

В объектной ориентированности присутствует как параметрический полиморфизм, так и ad-hoc-полиморфизм (перекрытие).

В *параметрическом полиморфизме* используются те же самые тела функций и, соответственно, процедур, которые работают с объектами различных типов. В *ad-hoc-полиморфизме* существенно используются одинаковые обозначения функций и процедур, но с различными их телами. При этом идентификация вычислительных предписаний может осуществляться либо статически, во время трансляции, либо

динамически, в процессе выполнения программы, в зависимости от типов аргументов (англ. late binding). При этом получается тесная связь с подтипами, когда рассматриваются частичные отношения между множествами носителей и выбор вычислительного предписания осуществляется динамически при его применении к объектам.

Особенно важной концепцией в объектной ориентации является *наследование.* Основанием наследования является построение иерархии связей классов с помощью отношения "есть один из". Это особенно оправдывается при оформлении интерактивных оболочек, поскольку там все снова и снова встречаются похожие структуры данных и вычислительные предписания. Благодаря наследованию определенные описания могут быть схематично перенесены с одних заданных единиц описаний на другие. Это будет объяснено позднее на примере.

Существенными, техническими составными частями объектной ориентированности являются:

- концепция классов с отношениями наследования и быть частью (иерархия классов, отношение "является частью"),
- концепция коммуникации и устойчивость программных переменных (см. ниже),
- динамическое создание (воплощение) и удаление объектов.

Отношение "является частью" выражает то, что объекты одного класса являются объектами и другого класса в качестве его составной части. Технически это, как правило, означает, что класс обладает атрибутом с типом другого класса.

Отношение наследования "класс A есть один из классов B" выражает, что объект класса A является также и объектом класса B. Это значит, что класс A обладает всеми составными частями и свойствами (всеми атрибутами и методами), которые имеет класс B, и в данном случае еще некоторыми дополнительными. Класс A наследует составные части класса B. При наследовании делается различие между *единственным* и *множественным* наследованием. При единственном наследовании один класс наследует свойства самое большее от одного из классов. При множественном наследовании один класс наследует свойства от нескольких других классов.

Наследование означает, что все атрибуты и методы одного класса имеются в распоряжении и класса-наследника. Во многих ЯП наследуемые методы могут быть объявлены заново. Однако обычно требуется, чтобы синтаксический интерфейс при этом не изменялся.

Далее даются простые примеры классов и 00-концепций, чтобы разъяснить эти понятия.

Алгебраические спецификации, как спецификация POOL в разд. 5.1.2, описывают вычислительные структуры. Подкласс алгебраической спецификации можно образовать следующим образом: **spec** QUEUE =

import POOL,
sort bool, data, queue data,
subsort queue data inherits from set data,
fct deq == (queue data) queue data, fct
next = (queue data) data Axioms:
deq(add(empty, e)) = empty,
next(add(empty, e)) = e,
deq(add(add(q, d), e)) = add(deq(add(q, d)), e), ~

next(add(add(q, d), e) = next(add(q, d)) **end_of_spec spec** QUEUE 1 =
sort bool, data, queuel data,
subsort queuel data inherits from queue data Axioms:
any(s) = next(s) **end_pf_spec**

Переменная v типа **var queue data** может принимать значения типа **queue data** или типа **queuel data.** Тем самым результат вызова any(v) будет зависеть от типа переменной v. Если значение v имеет тип **queuel data,** то вызов any(v) соответствует вызову next(v), в противном случае - нет. Тогда говорится о *полиморфизме* и о *динамическом связывании* (англ. late binding).

Как правило, объекты имеют состояния, которые определяются значениями их атрибутов. Это ведет к понятию *устойчивости,* объект образует устойчивость данных, так, как содержит постоянно существующие программные переменные, которые доступны только с помощью методов (функций или процедур доступа). Эти переменные называются атрибутами.

Атрибуты в качестве значений могут содержать также ссылки на объекты. Это определяет структуру связей между объектами.

Операционно можно смоделировать устойчивость посредством спецификации большого глобального пространства состояний, в котором состояния отдельных объектов содержатся как частичные состояния. Каждый метод может изменять соответствующую часть.

Инкапсуляция программных переменных в объекты с помощью атрибутов ведет, строго говоря, только к специальным правилам видимости для программных переменных.

Модульной формализации устойчивости данных можно достигнуть, например, с помощью следующих математических концепций:

- автоматов ввода/вывода,
- потокообрабатывающих функций,

В частности, объект можно понимать как машину (систему) состояний с входом и выходом.

Объекты в ЯП параллельной моделью выполнения могут трактоваться как интерактивные компоненты. Интерактивная компонента осуществляет коммуникации с помощью сообщений (вызов метода тогда соответствует сообщению с возвратом ответа) и обладает состоянием, которое изменяется через обмен сообщениями. Отсюда становится ясным, каким образом формализация устойчивости данных может быть осуществлена с помощью автоматов ввода/вывода или потокообрабатывающих функций.

Объекты возникают во время выполнения программы как воплощения классов. Во многих 00-языках программирования существуют универсальные методы специально для создания объектов. Созданные объекты, как правило, идентифицируются ссылками. В более сложных примерах также и ссылки на объекты могут возвращаться в качестве результатов.

Формализация этого обращения с классами и объектами возможна с помощью явного представления ссылок и создания объектов. Это ведет к понятию состояния по аналогии с состоянием памяти

Концепция наследования касается таких составных частей класса, как:

- атрибуты (компоненты состояния),
- методы (функции и процедуры), как имена (ad-hoc-полиморфизм), как предписания (параметрический полиморфизм),

- свойства поведения.

Дополнительные отношения на классах и объектах, которые часто имеют место в 00-описаниях, это:

- является (чем-либо),
- знает, использует, клиент,
- является частью (чего-либо).

Эти отношения выражают определенные связи между объектами некоторого класса. Для разработки систем и software представляет интерес формализация этих отношений.

00-программирование является, несомненно, одним из наиболее интересных направлений для профессиональной разработки программ. Впрочем, здесь имеются еще нерешенные проблемы:

- интерфейс часто бывает недостаточно описан;
- в проектировании software концепция воплощения трудно осваивается и просматривается;
- нахождение классов часто бывает трудным и требует больших затрат;
- отсутствуют формальные модели;
- концепция модульности для программирования по большому счету недостаточна (классы для этого слишком мелки); связь с обычной software-инженерией зачастую описана неясно (E/R-моделирование, переход от последовательной модели выполнения программ к параллельной).

Тем не менее, от объектной ориентированности ожидаются большие выгоды в отношении снижения стоимости, повышения качества и повторной применимости software в процессе проектирования информационных систем.

Эффективные алгоритмы и структуры данных: алгоритмы сортировки и их сложность

Часто используемыми эффективными алгоритмами сортировки являются метод пузырька, наиболее простой для программирования, сортировка через выбор, метод вставки и несколько более трудоемкие для программирования метод быстрой сортировки, сортировка слиянием и сортировка через деревья выбора. Ниже приводятся вычислительные предписания для этих способов сортировки:

- insertsort метод вставки,
- selectsort через выбор,
- bubblesort метод пузырька,
- mergesort метод слияния,
- quicksort быстрая сортировка,
- heapsort через деревья выбора.

Сложность по времени, так же как и сложность по памяти оценивается в зависимости от длины n сортируемой последовательности. По мере необходимости будет сделано различие между *средней сложностью* и *сложностью в самом неблагоприятном случае.* Средняя сложность показывает, сколь велики затраты алгоритма в среднем для сортировки последовательности длины n, если предположить, что каждое упорядочение элементов (каждая их перестановка) имеет одну и ту же вероятность. Сложность в самом неблагоприятном случае передает максимальные затраты, возможные при сортировке. Эти затраты имеют место, когда элементы в исходной последовательности упорядочены самым неблагоприятным для данного алгоритма образом.

На самом деле затраты того или иного метода сортировки часто сильно зависят от упорядоченности элементов исходной последовательности, если последовательность играет заметную роль. В дальнейшем, считается, что элементы в исходной последовательности упорядочены случайным образом.

Методы сортировки insertsort и selectsort имеют среднюю временную сложность $O(n^2)$. При этом требуется n шагов вставки и выбора, а на i-м шаге затраты на вставку равны i, а на выбор n-i. В обоих случаях получается следующая формула для затрат:

$$\sum_{i=1}^{n} i=\sum_{i=0}^{n-1}(n-1)=n*(n-1)/2=O(n^2).$$

Если последовательность в процессе сортировки хранится целиком в оперативной памяти, то говорят о *внутренней сортировке*. Если же последовательность хранится во внешней памяти, например, организована в файлы, то говорят о *внешней сортировке.* Как правило быстрая сортировка (quicksort) наиболее удобна для внутренней сортировки больших последовательностей со случайным упорядочением элементов. Для внешней сортировки предпочтительнее метод слияния.

Пути в графах

Многие практические задачи информатики могут быть отображены на отношения и графы. При этом типичны проблемы, связанные с существованием путей в графах. Простейшей постановкой задачи, например, является достижимость (существование пути) в графе, которая родственна образованию транзитивного замыкания для отношения. Другие часто используемые алгоритмы решают вопрос о равенстве структур графов, например, установление изоморфности графов, или другие вопросы эквивалентности.

Деревья. Упорядоченные ориентированные и отсориентированные деревья

Общесистемным является взгляд на деревья как на графы, причем данные представляются в корне и листьях поддеревьев в качестве меток вершин, а функции доступа (селекторы) представляются ребрами.

Неориентированное дерево с этой точки зрения есть неориентированный связный ациклический граф. Два неориентированных дизъюнктивных дерева могут быть объединены путем добавления ребра между любыми двумя вершинами, по одной из каждого дерева, в единое неориентированное дерево.

Ориентированное дерево есть направленный связный ациклический граф, который содержит одну вершину, называемую корнем, такую, что каждая вершина графа может быть достигнута, исходя из корня, единственным путем. Корень дерева определен однозначно. Лист дерева в представлении дерева в виде графа является вершиной без наследников.

Упорядоченное дерево в графовом представлении есть ориентированное дерево, на поддеревьях которого задан порядок следования. Упорядоченные деревья с точки зрения вычислительной структуры могут быть описаны следующим объявлением типа:

sort ordtree = ordtree(m root, seq ordtree subtrees).

Число непосредственных поддеревьев соответствует длине последовательности в дереве в приведенном выше представлении типа. Это число называется *степенью ветвимости* дерева. Считается, что z-максимальная степень ветвимости какого-либо

дерева, если оно само и все его поддеревья имеют степень ветвимости не превосходящую z.

Двоичное дерево является частным случаем упорядоченного дерева. Двоичные деревья имеют максимальную степень ветвимости 2. В дальнейшем, если речь идет об ориентированных деревьях, будет употреблено просто «деревья».

Максимальное число вершин в путях по дереву называется *высотой дерева*. Высота пустого дерева, таким образом, равна нулю, а высота одноэлементного дерева есть 1. Высота hi(b) дерева b определяется самым длинным путем в дереве. Дерево называется *полным*, если все его поддеревья имеют одинаковую степень ветвимости и все пути доступа от корня к его листьям имеют одинаковые длины.

Пусть #b есть число вершин в дереве, а m—максимальная степень ветвимости. Для непустого, полного дерева справедливо $hi(b)-1 \le \log_m b$.

В соответствии с этим в полном дереве число его вершин растет экспоненциально с высотой дерева.

Если на вершинах дерева задан линейный порядок, то ориентированное дерево называется *отсортированным*, если прохождение по порядку следования вершин дает возрастающую последовательность их содержимого.

Представление деревьев массивами

В машине дерево может быть представлено в виде списка. В этом представлении для хранения дерева требуется дополнительное место для указателей на поддеревья в вершинах. Однако дерево можно так же представить в виде массива, для чего следует расположить дерево в массиве по слоям.

Путем использования заполняющих элементов можно представлять и неполные деревья. Впрочем, при этом представлении пропадает много места в памяти, если представляемое дерево является существенно неполным. Также и перестройка так представленных деревьев путем перестановки поддеревьев требует больших затрат и весьма затруднительна.

AVL-деревья

Отсортированное двоичное дерево есть двоичное дерево типа **tree m**, в котором все вершины в левом поддереве не больше, а в правом поддереве не меньше корня и все поддеревья также отсортированы. Тогда обход дерева в порядке следования вершин (левое поддерево, корень, правое поддерево) дает упорядоченную последовательность. Логическая функция проверяющая отсортированность двоичного дерева, имеет вид:

```
fct sortiert= (tree m t) bool: if
isempty(t) then true else
sortiert(left(t))∧ sortiert(right(t))∧
allnodes(left(t), (m x) bool: x≤root(t))∧
allnodes(right(t), (m x) bool: root(t)≤x) fi
```

При этом вызов allnodes(t, p) предписания allnodes проверяет, все ли вершины в дереве t выполняют предикат p.

```
fct allnodes= (tree m t, fct(m) bool p) bool:
    if isempty(t) then true else
allnodes(left(t), p)∧
```

allnodes(right(t), p)л
p(root(t))

fi

AVL-дерево есть отсортированное двоичное дерево, в котором во всех его поддеревьях высоты левого и правого поддеревьев отличаются не более, чем на единицу. Такое дерево называется также *сбалансированным.* Булевское предписание, проверяющее сбалансированность дерева, имеет вид: fct isbal=(tree m t) bool: if isempty(t) then true **else** isbal(left(t)Aisbal(right(t))A

=> 1<hi(left(t))-hi(right(t))<1 fi

AVL-деревья являются компромиссом между полными и произвольными деревьями. AVL-дерево b высоты hi(b) содержит по меньшей мере $2^{(hi(b)/3)}$-2 вершин. Это можно показать индукцией по высоте деревьев. В AVL-деревьях число вершин растет также экспоненциально с высотой дерева.

B-деревья

Другой концепцией для специальных деревьев с интересными свойствами для эффективного управления большими множествами данных являются B-деревья.

Хотя для невырожденных двоичных деревьев их высота растет логарифмически с ростом числа вершин, все же реализация с помощью указателей приводит к заметным дополнительным затратам памяти для целей управления. Компромиссом здесь являются так называемые B-деревья. B-дерево либо пусто, либо содержит, по меньшей мере, n и самое большее 2n непосредственных поддеревьев. B-дерево над типом m, на котором пусть задан линейный порядок, является тогда элементом приведенного ниже объявления типа **btree:**

sort btree=empty **I** cbt(btree first, seq cell rest)
sort cell=mc(m d, btree bt).

При этом для B-деревьев с порядком n предполагается, что все их последовательности поддеревьев содержат от n до 2n элементов. Элементы данных линейно упорядочены в последовательных вершинах. Таким образом, B-дерево есть упорядоченное дерево со степенью ветвимости между n и 2n.

Добавление элемента в B-дерево делается очень просто. Если для вершины последовательности не находится места (поскольку длина последовательности равна 2n), то эту последовательность можно разбить на две последовательности (длины n) и тем самым снова получить B-дерево.

На основании правил работы с массивами обеспечивается, что эти массивы всегда заполнены, по меньшей мере, наполовину. B-деревья используются, в частности, при хранении больших множеств данных во внешней памяти в связи с банками данных.

Эффективное представление множеств

Вычислительная структура *множество* встречается во многих приложениях. И хотя множества являются конечными, они все же могут иметь очень большую мощность. Эффективность алгоритмов, в которых встречается много операций доступа к большим множествам данных, в решающей мере зависит от того, могут ли быть быстро выполнены операции доступа к множествам. Поэтому для представления

больших множеств часто выбирают сложные структуры данных, на которых операции доступа могут быть реализованы эффективно.

В этой части рассматривается структура данных множества со следующими операциями доступа: включение элемента в множество, выяснение принадлежности элемента множеству и с определенными ограничениями - операцию исключения элемента из множества.

Если рассматриваются маленькие множества и необходимы операции пересечения и объединения, то можно порекомендовать представление множеств в виде битовых векторов. При этом подмножества основного множества представляются векторами битов, длина которых соответствует мощности основного множества. Если какой-либо элемент принадлежит множеству, то в соответствующий бит заносится значение **true**, в противном случае - значение **false.** Такое представление плохо подходит для очень большого основного множества, поскольку теряется много памяти, особенно в том случае, когда должны представляться подмножества с относительно малой мощностью. В этом случае предлагается рассматриваемое ниже представление, а именно методика хэширования.

Вычислительная структура множеств с доступом по ключу

В приложениях часто бывают нужны структуры данных, которые содержат большое количество данных и доступ к которым осуществляется с помощью ключей. Пусть дано множество ключей с помощью типа sort key.
fct key=(data) key. Выше приведенная функция осуществляет доступ к порциям данных типа **data**, содержащих в себе компоненту типа key.
При этом предполагается, что каждая порция данных идентифицируется своим ключом. Тогда обращаться к нужной порции данных можно путем задания ключа. Необходимо найти тип **store,** который позволяет заносить порции данных и получить эффективный доступ к ним. При этом используются следующие функции: fct emptystore=store, fct get=(store, key) data, fct insert=(store, data) store, fct delete=(store,key) store. Способ действия этих функций можно представить следующими равенствами: k = key(d) => get(insert(s, d), k) = d k ***Φ*** key(d) => get(insert(s, d), k) = get(s, k), delete(emptystore, k) = emptystore, k = key(d) => delete(insert(s, d), k) = delete(s, k), k ***Φ*** key(d) => delete(insert(s, d), k) = insert(delete(s, k), d).

Метод хэширования

Метод хэширования позволяет хранить множество элементов в линейном массиве длиной z. Для этого нужна функция расстановки («рассыпания»):

h: **key**->[0:z-1], которая каждый элемент типа key отображает на индекс в множестве [0:z-1]. Эта функция устанавливает, под каким индексом будет храниться данный элемент в массиве. Используем h(m) в качестве индекса (также называемого ключом) для запоминания элемента данных в массиве

sort store = [0:z-1] **array data** a.

Как правило, число элементов типа **key** значительно больше, чем z. Тогда функция h наверняка неинъективна. Возможно хранение элемента b с ключом m в массиве a под индексом h(m). Получаются следующие реализации функций:

fct emptystore = **store**: emptyarray,

fct get = (**store** a, **key** k) **data**: a[h(k)]

fct insert = (**store** a, **data** d) **store**: update(a, h(key(d)), d),

fct delete = (**store** a, **key** k) **store**: update(a, h(k), empty).

Здесь предполагается, что empty обозначает элемент типа **data,** который играет роль держателя места. Функции работают корректно, пока для всех встречающихся ключей значения функции расстановки различны. Возникает проблема, когда нужно запоминать два различных элемента с ключами m1 и m2, и при этом оказывается h(m1)=h(m2). В этом случае говорят о *коллизии.*

Для использования метода хэширования надо справиться со следующими проблемами:

- определения величины массива и тем самым числа z значений индексов,
- выбор функции расстановки h,
- определение способа разрешения коллизий.

При этом имеются экспериментальные данные относительно того, сколь большим должен быть выбран массив a, чтобы, с одной стороны, вероятность коллизий не была слишком велика и, с другой стороны, не пропадало слишком много памяти, если заняты не все позиции массива.

Размер массива должен быть выбран таким, чтобы массив был занят не более чем на 90%.

Для выбора функции расстановки следует обратить внимание, что во многих практических применениях множество возможных ключей значительно больше числа допустимых значений индексов. В частности, обычно приходится исходить из того, что должна запоминаться только небольшая часть значений ключей, но при этом заранее не известно, каковы эти значения. Тогда существует заинтересованность в том, чтобы функция расстановки по возможности равномерно отображала множество значений ключей на значения индексов. Таким образом, принимаются предположения статистики.

Если сами ключи также заданы в виде натуральных чисел, например из интервала от 0 до s-1, где число s значительно больше числа z, то в качестве функции расстановки можно просто взять

h(i) = i **mod** z.

Эта функция фактически обладает тем свойством, что значения ключей равномерно распределяются по области значений индекса. Таким образом, если с помощью трансформационного отображения возможно значения ключей однозначно отобразить на интервал натуральных чисел, то эту функцию можно использовать в качестве функции расстановки. Впрочем, если имеются какие-то дополнительные статистические данные о распределении ключей, отличном от равномерного, то следует использовать другую, более подходящую функцию расстановки. Следует также обратить внимание, что вычисление этой функции не должно быть слишком трудоемким.

Примером, когда невозможно исходить из равномерного распределения ключей, является запоминание в хэш-памяти слов из некоторого текста. При наивном подходе напрашивается следующий способ действий: последовательные буквы слова кодировать двоичными цифрами и слово текста хранить в полученной таким образом двоичной кодировке, а в качестве функции расстановки принять просто проекцию—

например, в качестве значения функции принять код первой буквы слова. Однако этот способ, как правило, неудачен, так как он не сможет обеспечить равномерного распределения ключей по области значений индекса.

Для разрешения коллизий следует поступить следующим образом. Если при возникновении коллизии оба ключа должны быть запомнены в хэш-памяти, то дополнительно к собственно индексу для занесения в хэш-память должен быть найден заменяющий индекс. Здесь речь идет об *открытой адресации* в методе хэширования.

Предлагается и следующий, принципиально иной способ действий. На каждый индекс в хэш-памяти предусматривается занесение не одного содержимого, а целого их множества. Это может быть реализовано, например, путем образования списка из заносимых значений. Речь идет о *непосредственном сцеплении*. В этом случае после определения индекса для ключа надо просмотреть этот список и проверить, было ли занесение по этому индексу, и если да, то заносимый элемент должен быть внесен в этот список. Такой способ требует контроля за переполнением хэш-массива, а потому необходимости выделения дополнительной памяти для размещения приводящих к коллизии элементов, которые не удается разместить непосредственно в хэш-памяти. В этом случае говорится о *закрытой адресации* в методе хэширования.

При открытой адресации дополнительные элементы при коллизии помещаются в самом хэш-массиве. Поиск места в хэш-массиве для занесения элемента в случае возникновения коллизии будем называть *зондированием.*

Если при вычислении значения индекса для заданного ключа выясняется, что по этому индексу уже занесен элемент данных с другим ключом, то по определенному правилу вычисляется следующий индекс, по которому и заносится элемент данных. Если по этому индексу был уже занесен элемент данных, то вычисляется следующий индекс и т.д. - до тех пор, пока не будет найдено свободное место в массиве.

Если обращаться к какому-либо элементу в хэш-массиве для его чтения, то может случиться так, что по индексу для заданного ключа находится элемент с другим ключом. В таком случае аналогично тому, как это делалось при занесении элементов данных, надо перебирать последовательность значений индекса, по которым мог быть записан этот элемент.

Различают *линейное* и *квадратичное* зондирование. При линейном зондировании позиции хэш-массива просматриваются с постоянным шагом, а при квадратичном, исходя из значения h(i),—значения индекса

h(i)+1 **mod** z, h(i)+4 **mod** z, h(i)+9 **mod** z,…,h(i)+j^2 **mod** z.

Простой способ линейного зондирования для определения нового значения индекса в случае коллизии состоит в том, что значение индекса (по модулю z) по мере необходимости увеличивается на 1, пока не будет найдено свободное место в массиве. Впрочем, этот способ имеет то недостаток, что при некоторой статистике скоплений могут возникнуть сгущения в хэш-массиве. Это может привести к тому, что значительные области массива будут интенсивно загружены, вследствие чего потребуется длительный поиск свободного места для заносимого элемента, в то время как другие области будут заняты очень слабо.

Более подходящей будет такая функция разрешения коллизий, которая равномерно распределяет ключи по остальному множеству свободных мест. Однако этот способ может быть очень дорогим. Поэтому на практике ищут компромисс -берут, например, функцию, которая как это показано выше, распределяет ключи квадратичным образом. Впрочем, может случиться, что в процессе поиска по хэш-массиву не будет найдено свободного места для размещения элемента. При квадратичном зондировании будет просматриваться по меньшей мере половина массива, если в качестве его длины взято простое число.

Метод хэширования весьма эффективен, если используются только рассмотренные операции над элементами множеств. Если же в определенных приложениях должны использоваться все элементы данных, например, обработка типа сортировки или такие операции над множествами, как объединение и пересечение, то этот метод менее удобен. Чтобы и в этих случаях можно было эффективно работать с хэш-массивами, необходима трудоемкая предварительная сортировка.

Анализ статистики показывает, что метод хэширования чрезвычайно эффективен, если нет слишком большого заполнения хэш-массива. Даже его заполнение на 90% при достаточно удачно выбранном способе зондирования для занесения или выбора нужного элемента в среднем требует 2,56 шага зондирования. Это значение существенно зависит от степени загрузки массива. Поэтому размер хэш-массива следует выбирать таким, чтобы в процессе работы он заполнялся не более чем на 90%.

Отсюда вытекают и недостатки метода хэширования. Если размер массива выбрать слишком большим по отношению к числу фактически хранимых в нем элементов, то значительная часть выделенной памяти будет просто пропадать. Если же размер массива окажется слишком малым, то будет возникать слишком много коллизий, для их устранения придется просматривать длинные списки элементов и по затратам времени метод окажется неэффективным. Недостаток прежде всего состоит в том, что размер хэш-массива должен быть выбран и зафиксирован предварительно и этот размер не может быть динамически подогнан к числу фактически заносимых элементов.

Другой недостаток состоит в том, что весьма сложна процедура удаления элементов - особенно в тех случаях, когда используется техника размещения элементов, вызывающих коллизию. В этом случае при удалении элемента придется осуществлять перезапоминание элементов, которые размещались в памяти в результате разрешения коллизии, а это влечет за собой весьма сложные преобразования соответствующих списков.

Существует много различных вариантов метода хэширования. Рафинированные способы хэширования получаются при так называемом “grid file”, когда в хэш-памяти размещается информация с двумерными и многомерными китчами с помощью функций хэширования. Для этого плоскость или пространство делят не растры точками, которые хранятся в таблице, и отсюда определяют функцию хэширования.

Ресурсы и виды информационных систем. Семантическая модель реальности и идеальности

Семантическая роль знания и информации с кибернетических позиций рассматривается в неживой, живой и социальной природе на основе классификации интеллектуальных технологических процессов и их расширительной трактовке. При этом отражается единство информационных процессов для всех видов материи, включая естественные и искусственные (созданные человеком) системы.

Сформируем тезисно-концептуальные положения по этому заключительному разделу, относящемуся к понятию семантики:

- Семантика есть интерпретация связи содержания с формой.
- Информация как семантическая сущность материи - понятие системное и выражается в информации об объекте , о цели и необходимом силовом воздействии на объект. Источниками и приёмниками информации могут быть элементы бинарной системы любого вида материи – объект и субъект.

- Семантическая сущность информации, циркулирующей в человеко-машинных системах, как в любых искусственных системах, созданных человеком, проявляется через человека. В машинах нет плана-содержания, того, что существует в любых естественных системах. В искусственно созданные человеком информационные системы информация привнесена человеком. В машинах отсутствует интеллект. Машинная технологическая информация имеет формальный, синтаксический характер, семантическая же сущность сообщений остаётся человеком.
- В современном обществе технические средства коммуникаций играют все возрастающую роль в построении информационного общества. На практике это частенько приводит к переоценки роли машин в системах “человек-машина”, что неизбежно сказывается на эффективности таких систем.
- Различные виды материи различаются только своей структурой. Поэтому структуру вещества как носителя свойства можно принять за язык, средствами которого кодируются хранящиеся знания, а структурные параметры энергетических процессов- за язык кодирования передающейся в системе информации.
- При познании окружающей действительности и самого себя человек интерпретирует природные, вещественные и энергетические структуры подобно интерпретации созданных им же знаковых и сигнальных систем, в том числе реализованных средствами вычислительной техники. Главное – извлечение из этих структур их концептуального содержания.
- Важнейшие средства передачи информации в коммуникативных процессах – физические среды, каналы связи и системы кодирования. В социальных системах средством общения является естественный язык. Он встроен в символьной и образной форме в процессы двухполушарного мышления человека, через которые проявляется его семантическая сила.
- В социальных системах физическая среда и каналы связи строятся человеком, а в качестве системы кодирования выступает естественный язык – важнейшее средство человеческого общения и орудие мысли. В работе ЭВМ участвуют искусственные языки, в основе которых лежит двоичная система исчисления. Формальная интерпретация сообщений на этих языках на стадии разработки осуществляется программистами, что создаёт семантический барьер между пользователем и машиной на стадии эксплуатации.
- Для уменьшения семантического разрыва максимальное усилие необходимо предпринять в области информационных гипертехнологий, комплексно моделирующих техническими средствами кибернетические и коммуникативные возможности человека, повышая когнитивность искусственных систем и вытесняя формальных посредников в общении человека с машиной.

Информационные технологии с позиций системного анализа и проектирования ИС

Дадим основные определения и положения, относящиеся к понятию информационных технологий с позиций настоящего курса, полагая, что развитие вопроса, особенно в прикладных отношениях, заложено в целом спектре соответствующих учебных дисциплин:

- ***Технология*** – это совокупность приёмов нацеленных на создание чего-либо. Это определение относиться и к созданию информации. Реализация принципа непрерывности развития алгоритмическими методами отличается крайней трудоёмкостью и поглощает большую часть времени квалифицированных специалистов.

- Дальнейшие успехи в области интеллектуализации информационных технологий, связаны с моделированием способности к приближённым рассуждениям. Вместе с тем построение когнитивной среды требует создание автоматов, способных воспринимать текстовую, визуальную, звуковую и тактильную информацию, семантически сопрягать различные её виды, осуществлять логический вывод на основе обобщённой информации и изменять свою деятельность согласно формирующимся целям и окружающей ситуационной обстановки.
- Развитые интеллектуальные технологические процессы составляют главный информационный конвейер такой высокоорганизованной системы, какой является человек. Однако все перечисленные автоматизированные процессы могут стать реальностью только при высоком уровне развития их аппаратного обеспечения.
- В компьютерной семантике ЭВМ рассматривается как активный партнёр человека, осуществляющий интеллектуальную деятельность, направленную на сигнальное взаимодействие с человеком в единстве синтаксических, семантических и прагматических характеристик. Фактически проблема состоит в совместном участии человека и машины в одной системе и различении выполняемых ими функций.
- Деятельность в психологии понимается как динамическая система взаимодействий субъекта с миром, в процессе которых происходит возникновение и воплощение в объекте психологического образа и реализация опосредованных им отношений субъекта в предметной действительности. Основными характеристиками деятельности являются предметность и субъективность, которые могут быть перенесены в понятие компьютерной деятельности. Уже на предварительной стадии рассмотрения данной проблемы можно отметить важное отличительное свойство компьютерной семантической системы – её свободу от прошлого, личностной детерминированности человеческого поведения, его исключительности.

Предметная область с позиций моделирования и проектирования ИС

- ***Предметная область*** – это та часть реального мира, которая является сферой проблемной ориентации автоматизированной системы.
- Машины могут оперировать только определённой формой описания предметной области, при этом всегда предполагается соотнесение описания на формальном языке с её описанием на естественном.
- Введение формального языка приводит к необходимости однозначного перевода описания на одном языке (естественном и формальном) в описание на другом и необходимости эквивалентной интерпретации.
- При определении предметной области объект должен иметь относительно целостный характер и обладать некоторым конечным набором свойств.
- При этом свойства объекта могут быть индивидуальными и общими, присущими и единичному экземпляру, и целому классу.
- Технические средства коммуникации в современном обществе играют большую роль. Обмен знаниями посредством передачи, кодирования смысла и сигналов привёл к необходимости разработки общих принципов обработки, хранения, передачи информации.
- В настоящее время понятие информации включает передачу сведений между людьми, наследственных кодов и т.п.
- Информация выступает фундаментальным понятием кибернетики.

- ***Информация***-это всеобщее свойство материи, проявляющееся в коммуникативных процессах, которые содержат в себе субъектно-объектные отношения.
- Информационные единицы бывают элементарными и составными. Элементарными единицами информации выступают **реквизиты** - логически неделимые элементы, соотносимые с определённым свойством отображаемого объекта или процесса. Различают **числовые** и **текстовые** реквизиты.
- Числовые реквизиты характеризуют количественные свойства явлений, полученные в результате подсчёта натуральных единиц, взвешиванием, измерением.
- Текстовые реквизиты отражают качественные свойства явлений. Дают характеристику тем обстоятельствам, при которых протекало то или иное экономическое явление или процесс, и были получены те или иные числовые значения.

Представление и структура данных в информационных системах

- Структура данных связана с присутствием различных компонент, различными отношениями между ними, порядком следования компонент, установленных отношением.
- Структуру данных можно определить как множество компонент и отношений между ними, задающих порядок подчинённости и связи компонент.
- Если множество компонент полностью определено и конечно, то структура называется абсолютной или финитной.
- Все структуры данных могут быть разделены на иерархические, неиерархические и смешанные.
- В иерархических структурах каждая компонента является либо главной, определяющей, либо зависимой, подчинённой. Любая зависимая компонента связана только с одной операцией.
- Иерархическую структуру, у которой хотя бы одна компонента является одновременно и главной и зависимой по отношению к некоторым другим компонентам, называют многоуровневой.
- Порядок следования компонент определяется множеством иерархических отношений между ними, т.е. порядок подчинения.
- В неиерархических структурах любая зависимая компонента непосредственно связана с несколькими главными компонентами. В них не существует только зависимых или только определяющих компонент.
- Смешанные структуры являются комбинацией иерархических и неиерархических структур.

Иерархические структуры

- Иерархические структуры данных и связи между ними отображаются в виде схем, называющихся деревьями.
- Дерево представляет собой иерархию элементов, называемых узлами.
- На самом верхнем ее уровне имеется только один главный узел — корень дерева.
- Любой узел, кроме корня, связан с одним узлом на более высоком уровне. Этот узел называется исходным.

- Ни один узел не имеет более одного исходного. Любой рассмотренный узел может быть связан с одним или несколькими элементами на более низком уровне. Эти элементы называются порожденными.

Представление знаний. Классификационные системы

- Для представления знаний используется модельный подход. В качестве мододелей применяются классификационные системы, системы основанные на отношениях, семантические сети, фреймы продукции.
- Классификационные системы с давних пор применяются для структурирования и обобщения знаний. В таких системах, с одной стороны, все сущности разбиваются по определенным признакам на некоторое число классов, с другой — группируются вместе. При классифицировании пользователю дается набор объектов, которые можно описать некоторым множеством признаков. Любой объект может принадлежать одному или нескольким классам из фиксированного множества.
- Некоторые классификационные системы широко применяются для представления знаний. Вся совокупность употребляемых при классификации слов называется лексикой.

Иерархические системы. Алфавитно-предметная классификация

- Иерархическая система классификации - такая система, в которой между классификационными группировками установлено отношение подчинений, как правило, родовидовое.
- Классификационное множество объектов делится по некоторому выбранному признаку (основание деления) на крупные группировки. Затем любая группировка в соответствии с выбранным основанием деления разбивается на ряд последующих группировок, которые в свою очередь распадаются на более мелкие, постепенно конкретизируя объект классификации, причем однозначности должны быть учтены отношения синонимии, омонимии и полисемии, свойственные словам естественного языка.
- Основными преимуществами иерархической системы являются большая информационная емкость и простота поиска группировки; недостатки заключаются в малой гибкости структуры и невозможности агрегировать объекты по любому сочетанию признаков.
- Алфавитно-предметной классификацией называется система классов, расположенных в алфавитном порядке их имен.

Информация как ресурс с позиций анализа ИС

Обобщая и преломляя с позиций анализа ИС результаты исследований философов, экономистов, психологов, специалистов по техническим наукам применительно к проблеме использования информации как ресурса развития предприятий и организаций, можно выделить следующие основные специфические особенности информации, обусловливающие ее отличие от других видов ресурсов:

- *Практически неубывающая потенциальная эффективность информации,* которая часто реализуется далеко не сразу, а спустя многие годы.

* *Тиражируемость* и *многократность использования*, что является одним из условий неубывающей потенциальной эффективности информации и приводит к относительной *независимости информации от её создателей*.
* *Неаддитивность, не коммутативность и не ассоциативность* информации.
* *Кумулятивности* информации.
* Зависимость фактической *реализуемости* и *эффективности* от степени использования информации.
* Сообщение становиться информацией только в случае, когда есть *источник, переносчик* и *приёмник*, который должен хотеть воспринять информацию и быть способным её принять и использовать.
* *Материя* и *информация,* парные философские категории, так что появление новой информации всегда сопутствует появлению новых форм существования материальных объектов и процессов и обуславливает, влияет на их развитие, появление, переход в другие состояния (исчезновение), а также отражает все эти изменения, предписывая окружающему миру адекватные им реакции и процессы. Таким образом информации присуща связующая роль между материей, пространством и временем. Информация предопределяет и отражает всеобщее движение, в энергетическом смысле являющееся результатом изменчивости (переменности) плотности потока времени. Одновременно информация первична и вторична по отношению к категориям материи, пространства и времени – в этом ее диалектическая сущность и материальное воплощение.

Основные виды и формы информационного обеспечения средствами ИС

По мере развития автоматизированных систем начинает все больше проявляться их взаимное влияние и взаимодействие различных сфер информационного обеспечения, и в настоящее время начинает все больше пониматься необходимость формирования единой информационной сферы, в которой информацию необходимо классифицировать по разным признакам. Информационная сфера предприятия (организации) должна включать в себя весь спектр различных видов информации, отображающей состояние и функционирование конкретного предприятия или организации, а информационная сфера учреждения или науки, кроме того, давать всю необходимую совокупность знаний, сведений, справок для обучения и творчества и дидактические средства для их применения при поддержке ИС.

Разными исследователями предлагались различные способы классификации информационного обеспечения. Так, с точки зрения взаимодействия предприятия (организации) с окружающей средой всю информацию (в основном документальную) принято делить на *входящую* и *исходящую.* В зависимости от сроков хранения различают *постоянную, условно-постоянную* (иногда обновляемую) и *переменную* (регулярно изменяющуюся). Разделяют информацию и *по уровням управления* (заводская, внутризаводская, цеховая, внутрицеховая), *по характеру деятельности* (конструкторско-технологическая, бухгалтерская, учетно-отчетная, плановая и т. п.). В автоматизированных системах информационное обеспечение делят на *машинное* (в памяти ЭВМ) и *внемашинное.*

Различные классификации предлагались и использовались в системах управления, как правило, для информации, создаваемой и хранящейся в форме документов (приказов, писем, справочно-табличных форм статистической отчетности и т. п.), т.е. в виде *документальной* информации.

Однако по мере развития автоматизированных средств появилась возможность регистрации и хранения информации в виде отдельных фактов (предметов, событий, операций и т. п.), то есть в виде массивов *фактографической* информации, в которых

данные могут сортироваться по различным признакам и выводиться в различных формах, удобных для решения той или иной управленческой, проектной, научной задачи или задачи обучения.

Такие классификации помогают формировать из фактографической информации документы - формы статистической отчетности, справки для руководителей различных служб системы управления и т. п.

Подобные преобразования *документальной* и *фактографической* информации приводили к необходимости классификации информационных массивов: *массивы входной информации* (структура которых соответствовала структуре потоков или документов входной информации), *архивные массивы* (в которые преобразовывались входные массивы для долговременного хранения и обобщения), *выходные массивы* (структуры которых соответствовали наиболее часто требуемым формам представления информации по регламентированным запросам).

При преобразовании массивов происходило и преобразование информации: из *документальной входной* – в *фактографическую архивную,* из *архивной* - снова в *документальную* информацию, представляемую в формах, соответствующих регламентным запросам.

При этом, поскольку преобразования связаны с пересортировкой, в основном, фактографических данных, они не вызывают принципиальных трудностей, и в этих случаях не обязательно разделение информации на документальную и фактографическую, хотя и полезно с точки зрения осознания происходящих преобразований и определения характера и структуры массивов.

Разделение на документальную и фактографическую информацию носит более принципиальный характер, когда речь идет о научно-технической информации (монографиях, статьях, отчетах. патентах, законодательных актах и т. п.). Такая информация формируется человеком всегда в виде текстов, то есть в форме документальной информации, а тексты (даже относительно структурированные) имеют ряд принципиальных особенностей (синонимия, омонимия, парадоксы), которые затрудняют извлечение из них фактографической информации, необходимой для решения проектных или управленческих задач.

Для научно-технической информации также предлагались различные классификации: по видам источников документальной информации (первичные и вторичные), видам информационных изданий (общегосударственные, отраслевые, внутрифирменные).

Из вышеизложенного следует, что разработка структуры информационных ресурсов предприятия или организации и даже отдельно взятого пользователя является важной и сложной задачей, от решения которой во многом зависит эффективность их деятельности. Эту задачу следует решать с учетом конкретных особенностей предприятия (организации). При этом, вероятно, необходимо разрабатывать многоаспектную классификацию, которая позволит более полно охарактеризовать информационные ресурсы организации (то есть учесть вид, характер. назначение информации» ее направленность на внутренние потребности или на экспорт и т. д.). Важно также учесть и многоаспектность проблемы организации сбора, хранения, поиска и представления информации, что можно сделать путем стратифицированного представления информационной инфраструктуры.

Классификации ИС призваны отражать эти многогранные особенности информации. Так, например, информационно-поисковые системы (подсистемы ИС), используемые в образовании и научных исследованиях, описываются классификациями по уровням систем\подсистем, видам документальной обработки, видам структур, по системам индексирования, режимам информационного

обслуживания, видам и разнообразию критериев поиска, тематическому профилю комплектования с очерчиванием границ предметной области, по формам носителей информации, способам хранения, доставки, защиты, переработки, уровням интеграции лексики и многим другим признакам и свойствам. По характеру использования информационные системы делят на две большие группы: управленческие (АСУ) и информационные (ИС).

Сбор информации

Обязательным компонентом любой систем обработки данных, любой информационной системы является входной поток первичной информации, поступающей из внешней среды. Анализ внешней информационной среды предполагает: выявление источников необходимой информации и связей этих источников с информационной системой потребителя в заданной предметной области, а именно:

- оценку надежности источников информации;
- оценку достоверности информации, которой обладает источник:
- определение объемов и формы представления информации у источника;
- выяснение условий и особенностей предоставления информации источником;

Источниками информации могут служить любые объекты окружающего мира: предметы живой и неживой природы, события, явления, процессы (физические, экономические, социальные и прочие), а также сами информационные системы различного назначения и принадлежности. Это могут быть люди, владеющие не только сведениями о себе, но и профессиональными знаниями в определенной предметной области. Это могут быть предприятия и организации, включая средства массовой информации (СМИ), а также плоды их материальной и духовной деятельности.

Выбор источников информации и методов ее сбора во многом определяется результатами анализа внешней информационной среды. Критериями выбора источника служат:

- надежность и доступность источника;
- необходимость и достоверность предоставляемой информации;
- стоимость информационных услуг;
- совместимость формы существующего у источника представления информации с требованиями информационной системы потребителя.

Получение информации потребителем от источника в зависимости от организационных и экономических взаимоотношений между ними может осуществляться на безвозмездной или коммерческой основе по взаимному согласию или без него.

Информация директивного характера (законодательные акты, постановления, распоряжения и т.п. как правило, поступает принудительно по установленным каналам связи и не требует организации ее сбора. Такая информация регистрируется, аннотируется и систематизируется. Учетно-отчетная информация, предоставляемая физическими и юридическими лицами друг другу в соответствии с порядком, установленным законодательно, например, в виде деклараций о доходах или товарно-транспортных накладных, сопровождающих потоки ресурсов, также не нуждается в специальных мероприятиях сбора со стороны потребителя, участвующего в передаче информации.

Среди методов сбора нерегламентированной информации можно выделить:

- непрерывный мониторинг процессов и явлений;
- статистическое обследование информационных объектов;

- приобретение информации по подписке;
- электронный поиск в информационных сетях;
- разведка.

Примерами мониторинга могут служить: непрерывная автоматическая запись параметров технологического процесса, ежедневный учет объемов продаж или курсов валют. В любом случае, наблюдаемые данные должны фиксироваться в форме, удобной для контроля и последующей обработки.

Из перечисленных методов сбора особый интерес представляет статистическое обследование. Целью таких обследований, как правило, является получение оценок параметров или характеристик внешней или внутренней среды управляемой системы (например, предприятия или государства), необходимых для улучшения качества управления. В основе любого статистического обследования лежит получение информации от большого количества независимых, но однотипных источников, например, таких как человек или какой-нибудь продукт массового производства. Для получения удовлетворительных оценок нет необходимости обследовать всю совокупность этих источников, т.е. проводить полную перепись. Достаточно ограничиться информацией, взятой у некоторой представительной выборки, а оценку получить последующей статистической обработкой в соответствии с существующими методами.

Прием и регистрация собираемой первичной информации на входе информационной системы могут производиться вручную. автоматизированным способом или автоматически. При *ручном* способе фактографические данные об информационных объектах фиксируются на специальных бланках и в журналах в установленной табличной форме, а поступающие документы регистрируются и сохраняются в виде оригиналов или копий. При *автоматизированном* способе регистрация осуществляется на машинном носителе (например, винчестере ЭВМ) путем диалога оператора и компьютера. *Автоматическая* регистрация выполняется без участия человека и предполагает прямое подключение информационной системы к источнику. Например, при регистрации пассажиропотоков метро встроенными в турникеты фотоэлементными устройствами, подключенными к компьютеру информационной системы.

В этом случае входная информация представляет собой кодированный сигнал. При любом способе регистрации входная информация должна контролироваться с целью недопущения ее дублирования или обнаружения в ней ошибочных данных.

Собранная таким образом информация представляет собой первичные данные информационной системы. Эти данные подлежат дальнейшей форматизации и обработке в пакетном или диалоговом режиме с целью получения производных данных, используемых потребителями для принятия управленческих или проектных решений.

При проектировании конкретной информационной системы проблема выбора способа сбора информации при наличии альтернатив должна решаться в контексте общей проблемы технико-экономического обоснования. Это связано с возможной зависимостью затрат на последующую обработку информации от формы ее представления, диктуемой избранным способом сбора.

Обработка информации

Накопленная первичная информация, прошедшая предварительную обработку в форме отбора, систематизации и сохранения в базе данных в форматированном виде, может использоваться для решения многих задач анализа, планирования,

прогнозирования и управления в социально-экономических, технических, геофизических и др. системах. Постановка этих задач, как правило, связана с необходимостью получения на основе исходных накопленных данных производной информации, помогающей пользователю информационной системы принимать правильные решения. Такой производной информацией могут служить различные экономические, технические и др. показатели, параметры математических моделей, значения факторов, бездействующих на управляемый социально-экономический, технический или иной процесс, и многое другое.

В зависимости от способа получения, достоверности и объема исходной информации при ее обработке могут применяться как методы статистического анализа, так и математические методы обработки детерминированных данных. Отбор и загрузка информации на обработку обычно производится или в пакетном режиме (для статистической информации) или в диалоговом режиме (для детерминированной информации) с использованием методов поиска необходимых данных.

Среди статистических методов обработки можно выделить две группы методов: предварительной обработки и статистического анализа. К первым методам относят группировку и сводку данных. Группировка данных представляет собой выделение части данных из общей совокупности по одному или сочетанию нескольких признаков. Сводка (суммирование) данных обеспечивает их количественную оценку как по всей совокупности этих данных (простая сводка), так и по отдельным группам этой совокупности (групповая сводка).

Методы статистического анализа позволяют получать оценки более «тонких», чем дает сводка и группировка, характеристик экономических процессов и явлений. В числе таких характеристик:

- показатели роста, прироста, колебаний и других тенденций изменения во времени различных экономических факторов, выявляемых в результате анализа временных рядов:
- индексы потребительских и оптовых цен (в форме Ласпейреса, Пааше или Фишера), промышленного производства, состояния рынка ценных бумаг (индекс Доу-Джонса) и другие индексы, составляющие суть индексного анализа;
- прогнозные оценки экономических показателей в условиях неопределенности и риска, получаемые с использованием байесовского подхода;
- оценки величин и степени зависимости различных факторов на основе многофакторного регрессионного и корреляционного анализа и многое другое .

В отличие от статистических методов математические методы обработки детерминированной информации применяются для численного решения на основе этих данных задач планирования, моделирования и оптимизации с использованием типовых экономико-математических моделей объектов предметной области, обслуживаемой связанной с нею информационной системой. Наиболее часто, для решения этих задач, используются методы математического программирования, теории массового обслуживания, теории оптимального управления .

Программное обеспечение, реализующее эти методы обработки, может создаваться под конкретную информационную систему или в виде прикладных программных пакетов (ППП), не зависящих от содержания предметной области и предназначенных для использования специалистами, способными формализовать решаемые задачи на языке представления данных. Тому яркий пример – информационная система «1С», созданная для торгово-бухгалтерских учетов и отчетностей и обладающая собственным встроенным языком поддержки «1С».

Обобщая, отметим основные составляющие программного обеспечения современных и перспективных систем обработки данных:

- операционные системы (ОС);
- системы управления базами данных (СУБД);
- средства разработки приложений* (системы программирования):
- инструментальные средства технологии сквозного проектирования (CASE-технологии).

Программные обеспечение (ПО) систем обработки данных (СОД) включает в себя программные средства и документацию, необходимую для эксплуатации этих программных средств (Руководство пользователя. Руководство системного программиста и др.). ПО разделяют на общесистемное (базовое) и прикладное.

Общесистемное {базовое) ПО предназначено для организации процесса обработки информации в СОД и включает в себя операционную систему (ОС), сервисные программы, системы программирования (комплексные средства разработки программ на языках высокого уровня) и программы технического обслуживания.

Прикладное ПО предназначено для решения конкретных задач СОД. В него входят программные средства общего назначения и специальные программные средства для данной СОД. К *средствам общего назначения* относятся системы управления базами данных (СУБД), табличные процессоры, текстовые и графические редакторы и др. *Специальные* программные средства могут быть как разработаны для конкретной системы обработки данных, так и приобретены готовыми на отечественном рынке. При этом необходимое ПО может быть приобретено как “целиком” (если оно удовлетворяет всем необходимым требованиям), так и “собрано” из фрагментов готовых продуктов (возможно, с использованием услуг специалистов, называемых системными интеграторами).

Освоение современных ОС упростилось благодаря однотипности выполнения всех основных операций и наглядности выполняемых действий.

0бъектно-ориентированный подход позволяет при работе в среде современных ОС (например, Windows 2000) концентрировать внимание на объектах, представляющих файлы, папки, дисководы, программы, документы и т.д. Объектам на основании их свойств предоставляются системные ресурсы.

Системы управления базами данных. Система управления базами данных (СУБД), по определению, это комплекс программ и языковых средств, предназначенных для создания, ведения и использования баз данных. До 1995 года значительная часть ПО ИС разрабатывалась с использованием таких СУБД реляционного типа, как Clipper (вер. 5.0 и ниже) и FoxPro (вер. 2.5 и ниже). Для операционных систем Windows в наибольшей степени отвечающими требованиям СОД являются СУБД Visual FoxPro (версия 3.0 и выше) и СУБД MS Access из встроенного в Windows пакета Microsoft Office. Эти СУБД являются мощными и удобными средствами для разработки приложений баз данных с архитектурой клиент-сервер. В современных системах осуществлен *переход к базе данных,* в которой содержатся все включенные в нее таблицы, их индексы, постоянные связи между таблицами, хранимые процедуры, правила проверки значений полей и действия, выполняемые при добавлении новой записи, удалении и обновлении записи, называемые *триггерами.* Введены *новые средства для обработки данных с помощью SQL* (Structured Query Language - Структурированного Языка Запросов). Введена поддержка значений NULL (в дополнение к FALSE и TRUE) Для полей базы данных, предоставлены средства переноса баз данных на SQL-сервер и поддержки работы с удаленными источниками данных и многие другие сервисы.

Одновременно с наличием возможности процедурного пошагового программирования введены средства *объектно-ориентированного программирования.* При объектно-ориентированном подходе реальные предметы и понятия заменяются их

моделями, то есть определенными формальными конструкциями. Формальный характер моделей позволяет определить формальные зависимости между ними, формальные операции над ними и, в конечном итоге, получить формальную модель разрабатываемой программной системы как композицию моделей ее компонентов. Такой подход обеспечивает возможность модификации отдельных компонентов программного обеспечения без изменений остальных и повторного использования отдельных компонентов при перепроектировании системы.

Основными понятиями объектно-ориентированного программирования являются *класс, объект, свойство {атрибут), метод, событие.* Так, любой элемент управления или объект является в Visual FoxPro экземпляром класса. Например, элемент управления, создающий группу командных кнопок в экранной форме, принадлежит классу с именем CommandGroup, а объект “панель инструментов” - классу Toolbar. Класс содержит информацию о внешнем виде и поведении объекта, иными словами, описывает свойства (атрибуты) и методы обработки событий. Событие же представляет собой действия пользователя или операционной системы Windows, которые распознает объект. Таким образом, управление объектом осуществляется посредством обрабатываемых им событий. При создании нового объекта он наследует характеристики своего класса. Наследование позволяет определять также новые классы (производные, или дочерние) на основе существующих (родительских) классов и добавлять собственные свойства дочерних классов. В связи с этим средствами Visual FoxPro можно с существенно меньшими затратами, чем в более ранних версиях, создавать сложные программные системы.

Дальнейшее развитие получили средства *визуального программирования.* Разработан новый подход к созданию приложения в целом (в Visual FoxPro - организация проекта с применением диспетчера, представляющего проект в виде дерева и дающего возможность переключения между компонентами приложения и средствами разработки этих компонент) и использованию мастеров и построителей. Мастера (Wizard) позволяют полностью сконструировать любую новую компоненту, включая проектирование баз данных, отчетов, экранных форм, а с помощью построителей в экранную форму может быть встроен любой элемент управления.

На новом уровне организована поддержка *OLE-технологии* (Object Linking and Embedding - Связывания и Встраивания Объектов), добавлена возможность встраивания OLE-объектов в экранные формы и сохранения их в полях базы данных.

Реализована возможность *технологии перемещения и сброса объектов* (drag-and-drop), возможность перемещения таблиц и полей данных в экранные формы непосредственно из диспетчера проекта или из окна базы данных, использование контекстного меню.

Многие современные систем, в том числе приложения под Windows, имеют интегрированную среду разработки (IDE), то есть включают в себя редакторы кода, редакторы визуальных компонентов, компиляторы, отладчики, средства помощи и т.п. В современных системах используются преимущественно объектно-ориентированные языки программирования высокого уровня и встроенные в них возможности работы с базами данных. Активно реализуется также возможность использования языка SQL (и, следовательно, возможность создания баз данных с удаленным доступом).

Таким образом, одной из основных концепций в современных системах является концепция *объектно-ориентированного программирования.* Одним из ключевых понятий при этом является понятие компонентов, то есть готовых шаблонов для всех стандартных элементов приложений Windows (стандартных диалогов, окон, кнопок, списков и др.), поставляемых с системами; на их основе можно создавать свои собственные компоненты. Компоненты предоставляют программисту уже готовый

интерфейс с Windows, в них введено понятие события, которое программист обрабатывает вместо перехвата сообщений.

Еще одним современным понятием является понятие свойства. Можно сказать, что свойства в этих системах выступают в качестве простых переменных, но при этом во время проектирования приложения значения большинства из них отображены на экране и их можно менять (сразу же наблюдая результат изменения), а во время исполнения их можно рассматривать как функции - при этом система сама заботится об их выполнении вне зависимости от реального представления данных, с которыми они работают.

С современными системами поставляется библиотека визуальных компонентов (VCL), в которой содержатся шаблоны всех стандартных визуальных элементов Windows (а также многих специальных), так что программисту остается лишь незначительно изменять их по своему вкусу. Сам программист может создавать подобные шаблоны - и система не будет делать никаких различий между “родными” компонентами и добавленными (надо сказать, что эта возможность широко используется - множество фирм на Западе занимаются созданием компонентов, так что при наличии времени и доступа в Internet можно собрать практически любую не слишком сложную программу из уже готовых «заготовок»). Кроме того, естественно, при помощи систем можно создавать (и регистрировать) свои собственные DLL и статические библиотеки,

Важной особенностью систем является возможность использования объектов OLE (или DDE) - то есть можно, например, редактировать в своем приложении документ Word средствами самого Word. Хотя Delphi и C++Builder и не создавались как системы для работы с Internet и Web-дизайна, в них есть некоторые возможности и для этого. Современные системы должны обеспечивать совместимость, переносимость и масштабируемость.

Программное обеспечение для разработки ИС. Для решения задач разработки, сопровождения и модернизации информационных систем создаются технологии сквозного проектирования (ТСП). Эти технологии представляют собой набор компонент - программных продуктов и методов разработки, основные из которых и являются предметом нашего рассмотрения.

Интегрированный CASE-инструментарий. Предназначен для коллективной разработки, охватывающей все этапы жизненного цикла системы (от подготовки технического задания до генерации программного кода, внедрения и эксплуатации приложений).

Процесс проектирования начинается с формализации общих требований, предъявляемых к информационной системе, и предусматривает постепенную конкретизацию замысла с использованием механизмов декомпозиции и перехода к детальным техническим решениям, вплоть до получения готового программного кода, состава и топологии аппаратных средств проектируемой системы. В каждом продукте среда проектирования поддерживается удобным и легким в работе универсальным графическим редактором. Результатом выполнения проекта является:

• база данных проектируемой системы, включающая все необходимые физические объекты (таблицы, триггеры, хранимые процедуры), построенная с учетом проектируемой политики поддержания целостности данных;

• исходный код приложений информационной системы - программы, реализующие пользовательский интерфейс и логику приложений.

Разработка проекта поддерживается репозитарием, который обеспечивает централизованное хранение данных, проверку данных на взаимную непротиворечивость и полноту, сопровождение версий разработок. Репозитарий

поддерживает реальный многопользовательский режим для групп разработчиков. В качестве репозитария используется специальная проектная база данных.

Настраиваемые кодогенераторы. Позволяют с учетом возможностей CASE-инструментов получать исходный программный код приложений, доступных для последующего редактирования.

Средства реинжиниринга и репроектирования. Дают возможность не только "прочитать" структуру имеющейся базы данных, но и установить значения по умолчанию для управляющих параметров кодогенерации. С помощью удобного графического интерфейса среде MS Windows можно управлять процессом кодогенерации и получать новые версии приложений.

Среда разработки приложений. Представлена в первую очередь продуктами для информационных систем, построенных в архитектуре "клиент-сервер" с использованием языков программирования современного поколения,

СУБД и операционные системы. В основном используются серверы реляционных баз данных (SQL-серверы) и ОС UNIX, в качестве СУБД могут использоваться серверы Informix. ОС UNIX является основой реализации любого сложного проекта, поскольку органично сочетает все необходимые сервисы и предоставляет платформу для функционирования и интеграции современных программных продуктов. Для решения локальных задач обработки информации наряду с UNIX возможно применение и MS Windows NT,

Обеспечивающая часть ИС *(АСУ)*. При проектировании структуры обеспечивающей части ИС (АСУ) необходимо выбрать виды обеспечения и организовать взаимодействие между ними таким образом, чтобы они обеспечивали реализацию задач, входящих в подсистемы функциональной части ИС. При выборе технического и программного обеспечения учитываются особенности предприятия, его финансовые возможности, объемы информационных массивов, квалификации сотрудников и т.п.

В истории развития ИС первоначально был период, когда после разделения обеспечивающей части на составляющие отдельно разрабатывалось организационное (ОргО), информационное (ИО), техническое (ТО) и программное обеспечение (ПО). В результате такой независимой организации разработки структур этих видов обеспечения возникла проблема их совместимости. Поэтому в последующем был принят принцип единства ОргО, ТО, ИО и ПО.

Некоторые обобщения по выбору архитектур и моделей ИС

Итак, новыми компьютерными технологиями, на которые опираются при моделировании и проектировании ИС, являются экспертные и интеллектуальные системы, методы искусственного интеллекта, базы знаний, базы данных, нейронные сети, нечеткие системы. Современные информационные технологии позволяют создавать новое знание, выявляя скрытые закономерности, прогнозируя будущее состояние систем. Эффективным методом моделирования образовательных технологий, средством информационно-аналитического обеспечения обучения, творчества и научных исследований является метод имитационного моделирования, который позволяет исследовать и отображать комплексные системы обучения, творческого поиска, проектирования и управления с помощью экспериментального подхода. Это дает возможность на модели проиграть различные стратегии развития, сравнить альтернативы, учесть влияние многих факторов, в том числе с элементами неопределенности, одновременно с улучшением самой системы за счет этого не

прекращать каждодневную текущую информационную поддержку учебно-творческой деятельности множества коллективных и индивидуальных пользователей.

В числе других эффективных решений этих задач методы и технологии Data Mining, которые позволяют в автоматическом и полуавтоматическом режиме отыскивать скрытые зависимости и взаимосвязи в огромных массивах информации. Перспективно также применение комбинированных методов принятия решений в сочетании с технологиями Data Mining, методами искусственного интеллекта и компьютерным моделированием, различными имитационно-оптимизационными и экспертными процедурами.

Общие принципы построения системы поддержки принятия решений в научных исследованиях, учебных проектах и работах

Проектируемые для нужд образования и науки ИС в числе других решаемых задач могут быть нацелены на информационную, методическую и инструментальную поддержку процессов подготовки и принятия управленческих и проектных решений в процессе курсового, дипломного проектирования, выполнения научных поиска и исследований.

Для достижения этой цели в рамках ИС решаются следующие задачи:

- формирование и ведение аналитической базы данных соответствующих показателей, характеристик, справочных данных по интересующим предметным областям, научным и методическим подходам;
- комплексный анализ сложившейся ситуации и имеющегося опыта в интересующей сфере знаний и проектов, сравнительный анализ уровня известных решений и достижений с целью выделения диспропорций и точек воздействия на улучшение;
- динамическое моделирование происходящих, исследуемых и создаваемых процессов и явлений, связанных с ними социальных процессов; планирование расходов на реализацию поставленных задач; разработка социально и экономически сбалансированной технической политики и образовательной стратегии (в учреждениях образования);
- выполнение многовариантных прогнозных расчетов сценарного и целевого типа развития исследуемой сферы на основе комплекса динамических моделей;
- анализ и прогноз влияния макроэкономических и региональных факторов на развитие исследуемой и моделируемой ситуации в образовании, науке, творчестве, культуре и т.п. с оценкой научных, экономических, социальных, экологических и других важнейших последствий.

Предполагаемый состав информационно-аналитической системы, поддерживающей принятие решений в учебных и научных проектах

В **аналитической базе данных** ИС как минимум должны быть разработаны отчетные формы экспресс-просмотра и анализа информации в разных разрезах, аналитические формы для агрегирования, ранжирования, кластеризации и других видов обработки информации. Должен быть разработан блок отчетных форм, ориентированных на мониторинг различной поддерживающей поисковую и проектную работу информации, включая доступ к справочникам и методикам, программам и техническим заданиям, множеству прототипных решений и проектов, глоссариям расшифровки терминов, стандартам, нормативным актам, регламентам и рекомендациям, учебникам, задачникам, учебным пособиям и т.п. Отчеты должны

представляться в табличном и графическом виде, а также в обзорно воспринимаемом виде на уровне «витрин», демоверсий, подсказок, помощи.

Информационная система должна включать регулярно обновляемую подсистему **краткосрочного, среднесрочного и долгосрочного прогнозирования** развития исследуемой предметной области. Подсистема прогнозирования позволит оперативно оценить динамику изменения анализируемых показателей в течение жизненного цикла проектируемого изделия или создаваемой методики и в случае резких отклонений от планируемого курса своевременно принять соответствующие решения. Расчет многовариантных среднесрочных и долгосрочных прогнозов развития должен основываться на комплексной динамической имитационной модели исследуемой области, которую обслуживает создаваемая для нужд образования и науки ИС и, следовательно, создатели этой ИС обязаны изначально заложить в структуру ИС соответствующую подсистему с реальным информационным наполнением в виде методологии и первоначальных результатов трехступенчатого прогнозирования.. При формировании такой модели все основные социально-экономические значимые процессы, относящиеся к модели, могут отражаться во взаимосвязи с помощью разностных, балансовых и статистических уравнений.

Программные средства ИС должны позволять произвольно определять интервал прогнозирования, формировать сценарии вариантов прогнозов, просматривать и визуально сравнивать результаты расчета вариантов прогнозов. Результаты должны представляться в форме разнообразных табличных, графических и текстуальных отчетов.

Система должна быть открытой, с возможностью дальнейшего расширения функций ИС в необходимом направлении в режиме наращивания ее ресурсов, то есть без перепроектирования. Реализация и внедрение ИС должна предлагаться заказчику или руководителю работ в типовом, типизированном варианте или в виде, пригодном для сертификации проекта, со стандартным ограниченным набором отчетных форм и решаемых задач мониторинга и прогнозирования, а также сопровождающей проект документации. Разработка такой ИС непременно связана с проведением обследования предметной области, апробацией и внедренческой реализацией одного из вариантов проекта. Система может быть выполнена на основе того или иного аналитического комплекса или авторского инфологического решения, в среде "клиент-сервер" с использованием любой SQL-совместимой СУБД (Oracle, Informix, MS-SQL Server и других.). При этом для реализации в учебных, образовательных целях Минобразования РФ и головное учреждение науки и образования ГНИИ ИТТ «Информика» рекомендуют активно использовать возможности интегрального пакета Lotus Notes (Domino), широко применяющегося в учреждениях образования Европы, Америки и Японии. Внедряемая в образовательных и научных целях ИС должна максимально использовать современные возможности Интернет\Экстранет/Интранет-технологий.

Некоторые особенности обучения, творчества и научных исследований с позиций проектировщика ИС (с позиций системного подхода)

Методологической основой моделирования систем обучения, творчества и научного поиска, равно как и систем социально-экономического развития тех или иных объектов\субъектов, является системный анализ, центральной процедурой которого служит построение обобщенной (единой) модели объекта, отражающей важнейшие факторы и взаимосвязи реальной системы. В отношении образования это положение подкрепляется тем, что образование есть формирование гармоничной личности человека, где функции обучения представляют важнейшую составную

инструментальная часть формирования личности. Профессиональное образование, включая высшее, также направлено на дальнейшее формирование достойных членов социума, но наделенных необходимым избранным этими членами общества набором профессиональных знаний, умений и навыков, правами на профессиональную деятельность, позволяющими им плодотворно и квалифицированно трудиться в интересах общества в целом, своего государства, своего региона, своего учреждения, рабочего коллектива, своей семьи и в своих личных интересах. Оба вида образования неразрывно связаны с творчеством, как составной частью развития личности наряду с обучением, причем в профессиональном образовании творческая компонента четче обозначена в целевом отношении и призвана вносить большую долю созидательных признаков. Эта творческая часть образования, как и сама постановка обучения (включая центральную научную компоненту - дидактику) самым неразрывным образом связана с наукой, имеющей схожие с образованием системные признаки, позволяющие рассматривать науку, образование и творчество как целостную общественно-государственную систему. На практике на местах это рассмотрение связано с созданием комплекса моделей с развитыми динамическими и информационными связями между моделями всех уровней. В свою очередь комплексное многоуровневое иерархическое моделирование образования, творчества и научных исследований позволяют решать задачи всеобъемлющей эффективной информационно-аналитической поддержки этих видов профессиональной деятельности на уровне моделирования, проектирования, внедрения, эксплуатации соответствующих информационных систем (ИС).

Видение комплексной модели образования, творчества и науки позволяет просматривать и реализовывать в ИС необходимые структурные связи и обеспечивать информационный обмен и переработку информации в этих связях. В обобщенном смысле эти связи столь разнообразны и многогранны (в зависимости от того или иного подхода к проблеме, запроса, решаемой задачи), что речь может идти исключительно о многосистемных, многомодульных, древовидных моделях и решениях, образующих комплексную мегамодель образования, творчества и науки.

Здесь же следует выделить центральное звено образования – обучение, для которого определяющими в предметной области (то есть изучении того или иного предмета, курса на том или ином уровне обучения) являются две ведущие компоненты:

1. методология, дидактика обучения;
2. содержание и объем той части предметной области, которая предписана действующими нормативными актами к изучению и освоению на данном этапе по данному виду образования (в высшей школе (ВШ) эти характеристики описываются в отношении всех изучаемых предметов действующими государственными стандартами на специальности профессиональной подготовки специалистов, бакалавров и магистров).

В свою очередь содержание и объем осваиваемой предметной области можно разделить на минимально необходимую обязательную часть (в упомянутых выше стандартах ВШ строго регламентируется) и более свободную и индивидуализируемую по желанию обучающихся и обучающих развивающую часть. Развивающая часть осваивается наиболее ярко и продуктивно в дополнительном образовании, а в рамках базового образования реализуется учащимися в их учебном и внеучебном творчестве, курсовом и дипломном проектировании и т.п. Иными словами, эта часть относится к наиболее творческой компоненте обучения и тесно смыкается с наукой; своими результатами и методологией обогащает как науку, так и само образование.

Из приведенного выше модельного подхода к обучению в той или иной предметной области и научному развитию этой предметной области и методики ее

освоения следует, что вся эта совокупность творческой и образовательной деятельности в отношении указанной предметной области представляет некий комплекс, который должен поддерживаться специально созданной ИС, входящей наряду с другими аналогичными по устройству ИС других предметных областей в единую комплексную мегаинформационную многоуровневую систему образования и науки – сначала на уровне специальностей, направлений, затем (выше) на уровне факультетов, учреждений образования, ведомств, отрасли и т.п. Эта совокупность состава и методологии предметной области в своей обязательной и, следовательно, относительно неизменной постоянной части в высшей школе описывается так называемым Учебно-методическим комплексом (УМК), в состав которого входят основные организационно-методические и нормативные документы (в том числе выписки из стандарта по специальности, учебные планы и рабочие программы по дисциплинам и т.п.) и базовое в минимально необходимом объеме описание изучаемой предметной области (конспекты лекций, учебные пособия и учебники, задачники по курсу, методические указания и рекомендации по различным видам обязательной учебной деятельности в рамках изучения курса, например методические указания и рекомендации по курсовому проектированию или по выполнению лабораторного практикума и т.п.).

По кафедре ТИССУ МИРЭА и в объединении секторов НИТ и Информатики МГДД(Ю)Т сложилась практика формирования и размещения таких УМК по предметным областям изучения и творчества непосредственно в ядре ИС наряду с базовыми программными средствами (ПС), предопределяющими эффективное функционирование информационной системы. В информационном окружении ядра ИС размещаются динамично видоизменяемые многомодульные компоненты информационного расширения предметной области (различные справочники, глоссарии, сборники статей, рефератов, докладов, тематическая адресация в Интернет, подборки учебных и творческих работ – то есть своего рода тематическая электронная библиотека). Разумеется, информационное окружение ядра ИС по любой из дисциплин может в той или иной мере использоваться функционалами ИС других дисциплин. Таким образом, формируется многомодульная (гипертекстовая) информационная среда обучения и творчества по заданному направлению или заданной специальности. Основные обновления информсреды предметных областей в сфере НИТ и ДО публикуются в МГДД(Ю)Т в виде научно-методических сборников «Информсреда образования».

Отображая методологию и наполнение предметной области, ИС по специальности или направлению обучения и творчества, несет и другие функции, традиционно возлагаемые на дидактические системы. В их числе могут быть статучет и интеллектуализируемый анализ успеваемости, блок распорядительных документов с анализом их выполняемости, отдел кадров, учет ценностей и многое другое. Таким образом, комплексная система может быть декомпозирована на более частные системы и подсистемы в соответствие со своей сложной внутренней структурой.

Обсуждая обучение и особенно творчество как объект системной информатизации можно в самом обобщенном смысле считать, что система отображающая эти функционалы характеризуется иерархичностью управления и активностью отдельных ее подсистем, взаимодействие элементов в рамках которой рассматривается с учетом характера воздействий внешних быстро обновляемых и видоизменяющихся факторов на внутреннюю структуру. Интересно, что в соответствии с принципом единства и борьбы противоположностей, давление внешних видоизменений на сложившуюся в образовании систему уравновешивается традиционным консерватизмом, присущим образованию во все времена.

Образование и творчество, как объект моделирования, характеризуются:

- несовершенством теоретических знаний, размытостью теории развития образования и творчества на системном уровне;
- качественным характером знаний о системе, большой долей экспертных знаний при описании, структуризации объекта моделирования;
- высоким уровнем неопределенности исходной информации. (Различают внутреннюю и внешнюю неопределенность. Внутренняя неопределенность – это совокупность тех факторов, которые не контролируются лицом, принимающим решение полностью, но он может оказывать на них влияние. Внешняя неопределенность определяется характером взаимодействия с внешней средой или зависимостью от внешних неуправляемых внутрисистемных решений – это те факторы, которые находятся под слабым контролем лица принимающего решение рамках данной системы (например, сменяемость перечня и состава специальностей ВШ, регулируемая ведомством);
- совокупная взаимосвязь образования, творчества и науки представляет собой сложную динамическую систему;

Компьютерное моделирование систем обучения, творчества и научного поиска

Из приведенного выше описания особенностей обучения, творчества и научных исследований с позиций системного подхода напрашивается вывод, что в качестве метода моделирования этих систем, а точнее, комплексной многофункциональной мегасистемы, целесообразно выбрать **метод компьютерного моделирования**, поскольку он позволяет адекватно отразить структуру рассматриваемой сложной динамической системы, привнести в модель факторы неопределенности. Для образовательных и научных технологий очень привлекательно то обстоятельство, что метод компьютерного моделирования обеспечивает итеративный процесс разработки модели, характеризующийся постепенным углублением знаний о системе с участием экспертов и специалистов предметной области. Безусловно, в сферах образования и науки такими специалистами в первую очередь являются видные ученые, профессора и доценты вузов, передовые педагоги-новаторы, научные лидеры соответствующих научных школ и областей знаний, которые только во взаимодействии со специально подготовленными технологами – специалистами в сфере системотехники и проектирования ИС могут создать и реализовать средствами ИС действенные модели современной эффективной информационно-аналитической поддержки научных исследований, творчества и обучения в соответствующих областях науки и образования. Здесь же следует отметить, что подготовка специалистов высшей школой по специальностям «Информационные системы», «Информационные технологии в образовании» и «Информационные системы в научных исследованиях» и им подобных нацелена именно на решение этой важной народнохозяйственной задачи. Уместно отметить также, что достаточные темп и продуктивность работ в указанном направлении могут быть достигнуты только в случае массового участия в качестве специалистов и экспертов по моделированию, проектированию и сопровождению систем сотрудников, преподавателей, студентов и аспирантов учреждений образования и науки. Флагманом в развитии этого направления деятельности безусловно должна стать высшая школа России, причем немалый вклад в части поиска и апробаций новых решений в этом направлении может принадлежать системе дополнительного образования, отличающейся высокой концентрацией творчески одаренных и ищущих людей, относительной свободой творчества, углубленным развивающим образованием и инноватикой.

Компьютерное моделирование – это метод решения задачи анализа или синтеза сложной системы на основе использования ее компьютерной модели. Суть компьютерного моделирования заключена в получении количественных и качественных результатов на основе имеющейся модели.

Под компьютерной моделью понимают:

- условный образ объекта или некоторой системы, описанный с помощью взаимосвязанных компьютерных таблиц, блок-схем, диаграмм, графиков, рисунков, анимационных фрагментов, гипертекстов и т.д. и отображающий структуру и взаимосвязи между элементами объекта – ***структурно-функциональная модель***;
- отдельные программы, совокупность программ, программный комплекс, позволяющие с помощью последовательности вычислений и графического отображения их результатов воспроизводить (имитировать) и текстуально описывать процессы функционирования объекта при условии воздействия на него различных (включая случайные) факторов – ***имитационные модели***.

Компьютерное моделирование имеет ряд преимуществ по сравнению с другими подходами. В частности, оно дает возможность учитывать большое количество переменных, предсказывать развитие нелинейных процессов, возникновение синергетических эффектов, имеющих достаточно сложные и не всегда предсказуемые проявления в дидактике. Компьютерное моделирование позволяет не только получить прогноз, но и определить, какие управляющие воздействия приведут к наиболее благоприятному развитию событий, например, к быстрейшему разрешению той или иной научной проблемы или к интенсивной подготовке заданного числа специалистов по новой востребованной специальности.

Качественные выводы, сделанные по результатам компьютерного моделирования, позволяют обнаружить такие свойства сложной системы, как ее структуру, динамику развития, устойчивость, целостность, производительность взаимодействия с другими системами, объектами и субъектами и другие.. Количественные выводы в основном носят характер прогноза некоторых будущих или объяснения прошлых значений переменных и факторов, характеризующих систему. Одно из основных направлений использования компьютерного моделирования – поиск оптимальных вариантов внешнего воздействия на объект с целью получения наивысших показателей его функционирования, в том числе в результате планируемых целенаправленных видоизменений системы как путем видоизменения связей в ней, так и путем изменения ее наполнения обрабатываемой информации.

Компьютерное моделирование – эффективный метод решения задач анализа и синтеза сложных систем. Методологической основой компьютерного моделирования является системный анализ (в то время как у моделирования на ЭВМ фигурируют те или иные разделы теории математических моделей), - именно поэтому в ряде источников наряду с термином «компьютерного» используется термин «системного моделирования», а саму технологию системного моделирования призваны осваивать системные специалисты - аналитики.

Однако, ситуацию не стоит представлять так, что традиционные виды моделирования противопоставляются компьютерному моделированию. Наоборот, доминирующей тенденцией сегодня является взаимопроникновение всех видов моделирования, симбиоз различных информационных технологий в области моделирования, особенно для сложных приложений и комплексных проектов по моделированию. Так, например, имитационное моделирование включает в себя концептуальное моделирование (на ранних этапах формирования имитационной модели), логико-математическое (включая методы искусственного интеллекта) – для целей описания отдельных подсистем модели, а также в процедурах обработки и

анализа результатов вычислительного эксперимента и принятия решений; технологии проведения, планирования вычислительного эксперимента с соответствующими математическими методами привнесена в имитационное моделирование из физического (натурного) моделирования; наконец, структурно-функциональное моделирование используется при создании стратифицированного описания многомодельных комплексов.

Имитационное моделирование

Становление компьютерного моделирования связано с имитационным моделированием. ***Имитационное моделирование*** – один из видов компьютерного моделирования, использующий методологию системного анализа, центральной процедурой которого является построение обобщенной модели, отражающей все факторы реальной системы, в качестве же методологии исследования выступает вычислительный эксперимент.

Имитационная модель строится строго целенаправленно, поэтому для нее характерно адекватное отображение исследуемого объекта, логико-математическая модель системы представляет собой программно реализованный алгоритм функционирования системы. При имитационном моделировании структура моделируемой системы адекватно отображается в модели, а процесс ее функционирования имитируется на построенной модели. Под имитацией понимают проведение на компьютерах различных серий экспериментов с моделями, которые представлены в качестве некоторого набора (комплекса) компьютерных программ. Сравнение характеристик (конструкций, управлений) моделируемого объекта осуществляется путем вариантных просчетов. Особую роль имеет возможность многократного воспроизведения моделируемых процессов с последующей их статистической обработкой, позволяющая учитывать случайные внешние воздействия на изучаемый объект. На основе набираемой в ходе компьютерных экспериментов статистики делаются выводы в пользу того или иного варианта функционирования или конструкции реального объекта или сущности явления.

В ряде случаев формировать решения с помощью формальных методов не удается – эксперт должен быть включен в процесс принятия решения. Он становится активным компонентом информационной системы; детализирует проблему и модель, осуществляет постановку направленного вычислительного эксперимента на модели, генерацию и ранжирование альтернатив, выбор критериев для принятия решений, а также формирует рациональный вариант управления с помощью базы знаний. Принятие решений в условиях риска, например, требует ведения диалоговых процедур формирования статистически достоверных результатов и поэтапного сопоставления их с функцией цены риска. Необходимо осуществлять прямое участие эксперта в формировании оптимального множества вариантов решений и в процедурах вариантного синтеза.

Таким образом, имитационное моделирование значительно расширяет возможности и эффективность работы лиц, принимающих решения, предоставляя им удобный инструмент и средства для достижения поставленных целей. Имитационное моделирование реализует итерационный характер разработки модели системы, поэтапный характер детализации моделируемых подсистем, что позволяет постепенно увеличивать полноту оценки принимаемых решений по мере выявления новых проблем и получения новой информации.

Имитационная модель не дает оптимального решения подобно классическому решению задач оптимизации (минимизации), но она является удобным для системного

аналитика вспомогательным средством для поиска решения определенной проблемы. Область применения имитационных моделей практически не ограничена, это могут быть задачи: исследования структур сложных систем и их динамики, анализа узких мест, прогнозирования и планирования и т.д. Главным преимуществом имитационного моделирования является то, что эксперт может ответить на вопрос: «Что будет, если … », то есть с помощью эксперимента на модели вырабатывать стратегию развития.

В последнее время ведутся работы по разработке систем, способных оказать помощь эксперту при ответе на обратный вопрос «Что надо, чтобы …». Это можно назвать как «**целевое моделирование**», при котором на вход системы подаются показатели целевого состояния, а также перечень возможных регуляторов с указанием диапазона и шага их изменения. Система в автоматическом или полуавтоматическом режиме находит сочетание значений этих регуляторов для достижения заданного целевого состояния.

Итак, преимущества системно-динамического моделирования заключаются в следующем: системно-динамический подход начинается с попытки понять ту систему причин, которая породила проблему и продолжает поддерживать ее. Для этого собираются необходимые данные из различных источников, включая литературу, информированных людей (менеджеров, потребителей, конкурентов, экспертов, в образовании привлекают контрольные группы обучающихся и т.д.) и проводятся специальные количественные исследования. После того как элементарный анализ причин проблемы произведен, формальная модель считается построенной. Первоначально она представляется в виде логических диаграмм, отражающих причинно-следственные связи, которые затем преобразуются в сетевую модель, изображенную например графическими средствами системы “Ithink”. Затем эта сетевая модель автоматически преобразуется в ее математический аналог – систему уравнений, которая решается численными методами, встроенными в систему моделирования. Полученное решение представляется в виде графиков и таблиц, которые подвергаются критическому анализу. В результате модель пересматривается (изменяются параметры некоторых узлов сети, добавляются новые узлы, устанавливаются новые или изменяются существовавшие ранее связи и т.д.), затем модель вновь анализируется и так до тех пор, пока она не станет в достаточной мере соответствовать реальной ситуации. После того как модель построена, в ней выделяются управляемые параметры и выбираются такие значения этих параметров, при которых проблема либо снимается, либо перестает быть критически важной, либо переходит в режим поэтапного итерационного разрешения.

В процессе моделирования постепенно углубляется понимание проблемы участвующими в нем людьми. Однако их интуиция о возможных последствиях предлагаемых управленческих решений часто оказывается менее надежной, чем подход, связанный с тщательным построением математической модели. Иногда поведение таких систем оказывается настолько сложным, что его понимание лежит вне возможностей человеческой интуиции. **Компьютерное моделирование** – одно из наиболее эффективных имеющихся в настоящее время средств для поддержки и уточнения человеческой интуиции. Хотя модель и не является совершенно точным представлением реальности, она может быть использована для принятия более обоснованных решений, чем те, которые мог бы принять человек. Это гибкое средство, которое усиливает возможности человека, использующего ее для более глубокого понимания проблемы.

Таким образом, в сфере современных информационных технологий **имитационное моделирование** приобретает в мировых научных исследованиях и практической деятельности, включая обучение, весьма весомое значение. С помощью

имитационного моделирования эффективно решаются задачи самой широкой проблематики, - в области стратегического планирования, бизнес-моделирования, менеджмента (моделирование различного рода финансовых проектов, управление производством), реинжиниринга, проектирования (актуально применение имитационного моделирования в области инвестиционно-технологического проектирования, а также моделирования и прогнозирования социально-экономических и информационно-социальных систем, к которым в первую очередь можно отнести образование.

В качестве метода моделирования многосложных многоступенчатых инвариантных образовательных систем целесообразно выбрать модели системной динамики. Концепция системной динамики позволяет моделировать динамические процессы на высоком уровне агрегирования. В основе нее лежит представление о функционировании динамической системы, как совокупности потоков (накоплений знаний и навыков, формирования признаков подготавливаемого специалиста, описаний изучаемых предметных областей, реализации проектирования продукции, опирающегося на изучение предметных областей, человеческого фактора и т.п.

Модели образования и науки в самом общем виде представляют собой модели ресурсного типа (затратные механизмы м стоимость обучения, передаваемые и остаточные знания и навыки, успеваемость, характеристики интенсивности и насыщенности учебного процесса, выходные результаты и продукция обучения и т.п.). Ресурсы этих моделей видоизменяются, устаревают и обновляются, восполняются и иссякают, то тесть во многих случаях отвечают понятию ресурсов модели ресурсного типа, что вообще характерно для социально-экономических моделей. Поэтому состояние обобщенной социально-экономической системы, будь то образование, творчество или наука, в принципе можно описать переменными. Внешние воздействия, например, воздействие изменений на рынках труда на факторы и показатели подготовки специалистов, управленческие решения определяют динамику (темп) моделируемой системы (скорость подачи и изъятия ресурсов).

На основании обработки знаний экспертов выявляются все факторы, действующие в рассматриваемой системе, и причинно-следственные соотношения между ними. С помощью современных систем моделирования (таких, например, как IThink, VENSIM, DYNAMO и других) модель формируется на идеографическом уровне. Получаемые системные потоковые диаграммы являются формой структуризации знаний эксперта, в информационной сети которых вырабатывается рассогласование (дисбаланс) по различным видам потребностей и потребления ресурсов.

В блоках принятия решений на основе этой информации выдаются управляющие воздействия на различные виды объектов. Основной целевой задачей является установление баланса использования ресурсов в системе. Модели системной динамики применяются вместе с дифференциальными уравнениями балансового типа, а также в сочетании с принципами и методами логистики, основанными на оптимизации, управлении, интеграции потоков в сложных системах.

Таким образом, при разработке моделей социально-экономических систем аналитик должен учитывать некоторые особенности, о которых было сказано выше. При этом в ряде случаев достаточно не прибегая к поиску оптимальных стратегий развития (иногда это слишком сложно, дорого или невозможно вовсе) оказывается вполне достаточным найти приемлемое решение, отвечающее поставленной цели анализа и моделирования системы развития, некоторого компромиссного варианта, позволяющего учесть цели отдельных подсистем и обеспечить комплексное устойчивое развитие системы в целом.

Выбор инструментальной среды моделирования

Современные тенденции в области имитационного моделирования связаны с развитием проблемно-ориентированных систем, созданием встроенных средств для интеграции моделей в единый модельный комплекс; технологический уровень современных систем моделирования характеризуется большим выбором базовых концепций формализации и структуризации моделируемых систем, развитыми графическими интерфейсами и анимационным выводом результатов. Имитационные системы имеют средства для передачи информации из баз данных и других систем, или имеют доступ к процедурным языкам, что позволяет легко выполнять вычисления, связанные с планированием факторных экспериментов, автоматизированной оптимизацией и другими.

В качестве доминирующих базовых концепций формализации и структуризации в современных системах моделирования в числе других используются:

- для дискретного моделирования – системы, основанные на описании процессов (process description) или на сетевых концептах (network paradigms), - (Extend, Arena, ProModel, Witness, Taylor, Gpss/H-Proof и др.);
- для систем, ориентированных на непрерывное моделирование – модели и методы системной динамики, - (Powersim, Vensim, Dynamo, Stella, Ithink и др.)

Причем, в мощных системах, с целью расширения их функциональности присутствуют альтернативные концепции формализации. Так, например, в системах Powersim, Ithink встроен аппарат дискретного моделирования, и, наоборот, в системах Extend, ProcessModel реализована поддержка, правда, довольно слабая, непрерывного моделирования.

Большинство систем моделирования имеют удобный, легко интерпретируемый графический интерфейс, системные потоковые диаграммы или блок-схемы реализуются на идеографическом уровне, то есть рисуются, параметры моделей определяются через подменю. Сохраняются элементы программирования (на языках общего назначения или объектно-ориентированных) для отдельных элементов модели или создания специализированных блоков подготовленным пользователем, так называемое авторское моделирование (например, в системе Extend существует встроенный язык Modl для создания специализированных блоков).

Имитационные системы становятся все более проблемно-ориентированными. Известны системы моделирования производственных систем различного назначения (TOMAC, SIRE и другие), медицинского обслуживания (MEDMODEL), в области телекоммуникаций (COMNET) и другие. Для этого в проблемно-ориентированные системы моделирования включаются абстрактные элементы, языковые конструкции и наборы понятий, взятые непосредственно из предметной области исследований. Определенные преимущества имеют системы моделирования, декларирующие свою проблемную ориентацию, например, пакет Rethink, ориентирующийся на реинжиниринг. Все это, конечно, влияет на доступность и привлекательность имитационного моделирования.

В современных системах моделирования появляется некоторый инструментарий для создания стратифицированных моделей. **Стратификация систем**, являясь общим принципом системного моделирования, реализуется в технологии имитационного моделирования либо путем детализации, итерационной процедуры эволюции имитационной модели, - либо путем создания комплекса взаимосвязанных моделей, с развитыми информационными и имплицитными связями между моделями. Стратифицированные модели представляют собой машинно-ориентированные

понятия, предполагающие конструирование баз данных и знаний, над которыми определены вычислительные процессы решения задач системного анализа и принятия решения. Разработчики систем моделирования используют различные подходы для реализации стратифицированных моделей. Ряд программных продуктов, такие как AUTOMOD, ProModel, TAYLOR, WITNESS и др. поддерживают интеграцию моделей на основе создания вложенных структур. В системах Arena, Extend реализован подход к стратификации, основанный на построении иерархических многоуровневых структур. Наиболее перспективным является структурно-функциональный подход, реализованный, например, в системах моделирования Ithink, Rethink, базирующийся на методологии структурного анализа и проектирования. При такой технологии есть возможность для реализации нескольких уровней представления моделей, - высоко-уровневое представление в виде блок-схем, с использованием CASE - средств, а на нижнем уровне модели могут отображаться, например, потоковыми схемами и диаграммами.

Методология научного исследования в компьютерном моделировании, предполагающая организацию и проведение **вычислительного эксперимента на имитационной модели**, требует серьёзной математической и информационной поддержки процесса системного моделирования, особенно в части вычислительных процедур, связанных с планированием эксперимента, оптимизацией, организации работы с большим объёмом данных в процедурах принятия решений. Многие системы моделирования обеспечены средствами для интеграции с другими программными средами, осуществляют доступ к процедурным языкам, связанным с кодом имитационной модели, для реализации специальных вычислений, доступа к базам данных (подход Simulation Data Base).

В более мощных пакетах осуществляется интеграция через дополнительное программное обеспечение со специализированными блоками различного назначения. Это могут быть блоки анализа входных данных, гибкие средства анализа чувствительности, позволяющие осуществлять многократные прогоны с различными входными данными (в системах GPSS/H-PROOF, ProModel и др.). Перспективно создание систем моделирования с функционально широкими, ориентированными на специфику имитационного моделирования, блоками оптимизации (в этом смысле показательны системы WITNESS, TAYLOR). Интеграция программных систем, кстати, может осуществляться и на других уровнях, например, имитационное моделирование плюс логистики, что актуально, в частности, при реализации ресурсных моделей балансового типа.

Реализуемый в ряде систем многопользовательский режим, применение интерактивного распределенного моделирования, разработки в области взаимодействия имитационного моделирования с Интернетом, расширяют возможности имитационного моделирования, позволяя отрабатывать совместные или конкурирующие стратегии различным учреждениям, научным и образовательным школам и т.п.

Некоторые практические рекомендации по анализу и разработке социально-экономических моделей, поддерживаемых ИС, проектируемыми в интересах образования и научных исследований

Фундаментальные представления по созданию динамических моделей социально-экономического типа подробно рассмотрены в работах Дж. Форрестера и других известных авторов, а потому в данном материале не приводятся. Здесь уместнее сделать несколько реплик практического свойства, которые надо иметь ввиду в

процессе дипломного проектирования и при выполнении курсового проекта по дисциплине «Проектирование информационных систем».

Прежде всего, следует сосредоточить внимание на четком и однозначном понимании **цели** анализа и моделирования системы.

Согласно Форрестеру в число возможных целей могут входить:

- на модели предсказать поведение сложной системы,
- определить эффективность административных, управленческих программ в долгосрочной перспективе, показать противоречия между долгосрочными и краткосрочными целями;
- решить проблему пересмотра и усовершенствования подготовки по специальностям образования, отдельным предметам и проблемам и т.п.

В сложных системам образовательных технологий, особенно в случае третьей из упомянутых целей, присутствуют различные подсистемы (или входящие частные системы), например, подсистемы учебных дисциплин в системе специальности в целом или подсистемы специальностей в системе целостного направления (группы смежных специальностей) или подсистемы «предметная область обучения», «кадры», «успеваемость», «обратные связи в обучении» и т.п. – в зависимости от того, какой идеологии подчиняется структурирование системы. Подсистемы в проекте следует выделить, ранжировать и описать.

Далее полезно сосредоточить внимание на выявлении воздействия внешней среды и внешних факторов, ранжировать и описать их.

Важным моментом, заслуживающим пристального внимания, является утверждение Дж.Форрестера и других исследователей о том, что природа комплексных систем **контринтуитивна**. Таковые, настаивают они, включают взаимодействие слишком многих переменных, чтобы человеческий ум мог удерживать их в правильном порядке и обозримом виде.

По этой причине в данной ситуации при принятии решений необходимо использовать алгоритмы, а не полагаться на интуитивные суждения. Это неизбежно, когда приходится иметь дело со сложной динамической моделью с множеством прямых и обратных связей между подсистемами. Такой подход позволяет сравнивать между собой различные альтернативные программы развития социально-экономического объекта или системы. Следовательно, сравниваемые программы должны быть единообразно формализованы и описаны с учетом возможных отношений между ними и вводимых ограничений на их действие.

В результате реализации такого подхода на первоначальных этапах системного анализа вырисовываются контуры создаваемой комплексной системы (далее «комплекса»). В состав такого комплекса входят:

Аналитическая база данных

Аналитическая база данных в составе комплекса предназначена для хранения иерархически структурированной многомерной динамической информации. В наиболее общем виде комплекс может быть ориентирован на многомерное представление данных, однако допускается прямая обработка традиционных реляционных таблиц или логическое преобразование таблиц в многомерные объекты.

Интерфейс

Интуитивно понятный интерфейс в составе комплекса предоставляет пользователю удобные средства составления нерегламентированных запросов, генерации отчетов, анализа и преобразования данных. При этом особенностями аналитической базы можно считать:

- Возможность использования в качестве хранилища данных таблиц пользователей, пополняемых внешними по отношению к комплексу средствами. Таким образом,

существенно облегчается взаимодействие между средствами сбора информации и средствами ее анализа (различные блоки программного комплекса и уже используемое программное обеспечение других производителей); • Развитые возможности по преобразованию данных. В частности, предусматривается ряд механизмов, позволяющих производить вычисления, сравнения и агрегирование данных базы в реальном времени, что обеспечивает актуальность данных и не требует дополнительных объемов дискового пространства.

Источники данных

Необходимость в постоянной "подпитке" информацией из различных гетерогенных источников данных обуславливает целесообразность наличия в рамках комплекса инструментария гибкой настройки на внешние информационные массивы. Уместно ставить вопрос о поддержке шлюзов к SQL-ориентированным СУБД (Oracle, Informix, MS SQL Server и др.), СУБД класса персональных (Access, FoxPro, Paradox), электронным таблицам (Excel) и файлам (txt). При этом желательна загрузка информации из таблиц и файлов произвольной структуры в таблицы, используемые комплексом и предполагающие специальные представления данных.

Имитационное моделирование

Интерфейс блока моделирования предоставляет пользователю набор визуальных и программных средств для построения моделей исследуемых процессов и явлений. Графические средства и макроязык комплекса призваны обладать достаточной гибкостью и в то же время доступностью пользователю – непрограммисту, каковым в массовом плане является учащийся, преподаватель, научный работник, инженер, методист, управленческий работник. Хорошо, если основой всего блока является макроязык, ориентированный на обработку многомерных матриц и векторов. Кроме того, такой язык обладает стандартным набором управляющих конструкций, содержит специальные наборы математических функций, функций обработки файлов, статистических вычислений и обработки матриц с учетом иерархической структуры информации из аналитической базы. Так же в макроязык встраивается блок, позволяющий конструировать специальные диалоговые формы, которые могут быть вызваны в процессе моделирования и использованы для доопределения различных параметров моделей или вывода промежуточных результатов и которые фактически незаменимы в образовательных технологиях и творчестве.

Комплексный подход к построению социально-экономических моделей предполагает сочетание имитационных, целевых, оптимизационных и статистических методов моделирования и прогнозирования.

Многовариантные расчеты

Возможность выполнения многовариантных сценарных и целевых расчетов ("Что будет, если…?" и "Что нужно, чтобы…?" - анализ) на основе имитационных, оптимизационных и целевых моделей позволяет находить различные варианты стратегий, сравнивать между собой потенциальные результаты их реализации, ранжировать варианты по произвольным экономическим и иным критериям.

Конструктор отчетов

Конструктор отчетов комплекса может быть выполнен в форме электронной таблицы, что позволяет пользователю достаточно легко создавать аналитические и нормативные отчеты произвольной формы. Конструктор может иметь поддержку многомерного представления данных и специальные функции, обеспечивающие его тесную интеграцию с другими компонентами комплекса. Кроме этого, конструктор может располагать достаточно развитой подсистемой построения графиков и

диаграмм, возможностей импорта и экспорта отчетов в различные форматы, в том числе в файлы Excel и Html, что очень важно для интеграции создаваемой ИС в мировую информационную систему Интернет.

Электронные карты ГИС-технологий и т.п.

Наряду с табличным и графическим представлением, некоторые данные, отражающие научные и образовательные проблемы и результаты, могут быть отображены на электронных картах территорий, что в ряде случаев значительно упрощает их визуальный анализ (например, распределение обучающихся по той или иной специальности по стране или региону). При такой постановке задачи комплекс должен предоставлять средства простотой подготовки топографической основы. Для это в состав комплекса вводится специальный формат хранения картографической информации, который обеспечивает компактность и быстрое отображение. В технологиях ГИС имеется ряд средств, позволяющих импортировать карты из общепринятых форматов и использовать их в составе комплекса.

Приведенным выше перечнем средств и компонент, разумеется, описание возможного состава комплекса не исчерпано. Состав это может видоизменяться и расширяться в зависимости от поставленной перед проектировщиком ИС задачи. Так, например, может возникнуть потребность в реализации средств поддержки видеоконференций, создания мобильных ИС (функционирование которых не зависит от среды их использования) и многое другое.

В завершение настоящего рекомендательного раздела можно еще раз обратить внимание на то, что теоретическая модель комплекса для образовательных и научных технологий, как и всякая сложная модель, социально-экономического состояния или развития основывается на структурных закономерностях, отражающих связи между подсистемами и элементами этих подсистем. В образовании она строится, скорее всего, по выделенным на стадии анализа **морфологическим признакам. Система связей в такой модели в ряде случаев строится по схеме древовидного графа, а ядром системы является участок наивысшей концентрации или значимости элементов системы** или ее подсистем (например, УМК в подсистемах обслуживания отдельных учебных дисциплин той или иной специальности ВШ, рассматриваемой как система в целом). Наряду с главным ядром на системном уровне аналогично могут формироваться дополнительные ядра подсистемного уровня, занимающие центральное место в своей подсистеме. Все они имеют соответствующее гибкое информационное окружение. Все эти элементы (как ядер, так и окружений) связаны между собой определенными связями, которые, в свою очередь, могут являться элементами системы. Характер и оптимизация взаимоотношений и связей между всеми поименованными элементами и подсистемами анализируются с помощью методов морфологического анализа с использованием соответствующих методик минимизации и упрощений, широко описанных в специальной литературе.

Сообразно приведенному выше подходу проектанту ИС в результате анализа и синтеза системы вырисовывается перспектива выполнения программной реализации и исследования имитационной модели образовательных и научных технологий. Выстраивается агрегатированная модель, может быть получена и детальная модель. Строятся системные потоковые диаграммы; описываются подсистемы, для каждой из которых определяются и описываются уровни, потоки, входные переменные, промежуточные переменные, выходные данные; осуществляется параметризация имитационной модели.

Задача параметризации модели состоит в установлении производящих функций темпов потоков, то есть в составлении уравнений темпов, структура которых описана информационной сетью потоковой диаграммы. Составление уравнений темпов модели

системной динамики представляет собой процесс перевода вербальных описаний взаимозависимостей факторов моделируемой проблемной ситуации на язык четких количественных соотношений. Далее следуют испытания и дальнейшие исследования имитационной модели

Для повышения уровня доверия к результатам моделирования могут быть проведены оценки чувствительности и формальные процедуры верификации.

Чувствительность модели показывает, как сильно изменяется отклик при изменении того или иного фактора или регулятора.

Процедуры верификации проводятся для обеспечения правильности структуры модели и взаимосвязей между ее компонентами. При проведении этих процедур на модели воссоздают некоторые различные ситуации в предметной области и наблюдают поведение основных исследуемых показателей.

Далее, если отсутствует постановка иных специальных задач, переходят к расчету и обеспечению эффективности разработки и внедрения информационной системы (системы), к принятию мер по обеспечению ее полного жизненного цикла вплоть до полной ликвидации.

В завершение этого небольшого раздела следует напомнить о целесообразности широкого использования возможностей современного CASE-инструментария в процессе создания и выбора ПС информационных систем, рассмотрению которого в качестве средства построения ИС посвящены соответствующие тематические выпуски МГДД(Ю)Т и МИРЭА «Информсреда образования» (см. список источников). Уместно также отметить, что важнейшей частью проекта по созданию и сопровождению ИС является проектирование, наполнение и эффективное использование баз данных (БД) и баз знаний (БЗ), выбор систем управления ими (СУБД). Этим важным вопросам также посвящены соответствующие публикации МГДД(Ю)Т и МИРЭА (см. список источников).

Эффективность разработки и внедрения системы

Основной целью создания имитационной модели является анализ и прогнозирование основных показателей исследуемой и обслуживаемой сферы. В учреждениях образования и науки системы разрабатываются и внедряются для информационно-аналитической поддержки процесса организации и управления функционирования полноценных и высокоэффективных образовательных и научных технологий, в том числе с целью повышения эффективности процесса принятия управленческих решений и улучшения их качества подготовки обучающихся и выполняемых проектов и исследований.

Источниками эффективности разработки и внедрения таких систем являются:

- сокращение трудоемкости обработки информации при прогнозировании и анализе;
- сокращение времени составления прогнозов социально-экономического, функционального развития системы обучения и научных исследований;
- сокращение времени анализа альтернативных стратегических решений и выбора наиболее рационального, включая вопросы долгосрочного, среднесрочного, ближайшего и оперативного планирования и контроля за исполнением планов и программ работ и мероприятий;
- предотвращение возможного ущерба (как экономического, так и социального) от принятия неверных, нерациональных управленческих решений и ошибок в планировании, информационно-методическом и иных видах обеспечения учебного и творческого процессов, что особенно важно при разработке стратегических решений;

- достижение стабильного социально-экономического развития учреждения образования и\или науки, их подразделений и функционально-тематических направлений (научных школ, специальностей и т.п.) как следствие принятия наилучших управленческих решений.

Сетевой график разработки

Для достижения наилучших результатов проектных работ, контроля и оперативного управления их в процессе анализа, моделирования, проектирования, внедрения и эксплуатации системы на самой ранней стадии проектных работ следует составить календарный план работ, который может быть формализован в виде сетевого графика проекта. Так, например, для реализации такого сетевого графика создания и эксплуатации сложных ИС может быть использован моделирования на основе PERT–сетей, согласно которому ход разработки системы представляется в виде сетевого графика в терминах работ (дуги) и событий (вершины)

Сетевым графиком называется графическое изображение комплекса работ в виде ориентированного графа без контуров с дугами, имеющими одну или несколько числовых характеристик, отображающими технологическую взаимосвязь между работами.

Работа – это процесс в составе проекта, происходящий во времени, поэтому можно говорить об объеме работы, выполненному к моменту времени. Термин «работа» может иметь следующие значения:

1. действительная работа – или просто работа, то есть производственный или творческий процесс, требующий затрат труда, времени и материальных ресурсов;
2. зависимость (фиктивная работа) – работа, не требующая затрат труда, времени и ресурсов.

Действительную работу и ожидание на сетевом графике принято обозначать сплошной стрелкой, а фиктивную – пунктирной.

Событие – означает определенное состояние в процессе выполнения проектных работ, то есть событие – это определенный результат предшествующих работ, дающий возможность начать другие работы.

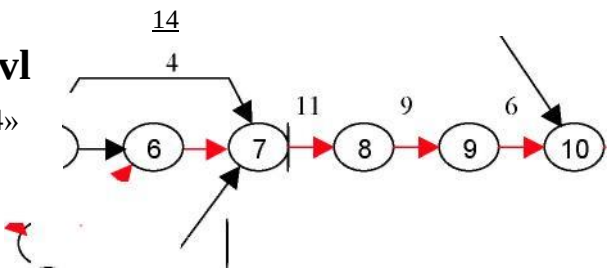
Пример иллюстрации хода разработки системы представлен ниже в форме сетевого графика, помещенного в таблицу, отражающую детализированный перечень проектных работ и событий.

24 7 8 |15 / 11 22

o)—Ki)------ ^CO-^vL/^vl

17^4»

/ 19



D ---- 7

Сетевой график работ по разработке и внедрению ИС (информационной системы).

Детализированный перечень работ и событий

Работа				**Предшествующее событие**		**Последующее событие**		$t_{ож}$
Код			Наименование	Код	Наименование	Код	Наименование	(дней)
0	–	1	Анализ проблемной области, ознакомление с прототипами, оценка целесообразности создания системы	0 1 1	Получено задание на создание системы	1 2 10 3	Постановка задачи	24 7 14 8
1	–	2	Формулирование проблемы, определение целей моделирования	2	Постановка задачи	4 5 6	Содержательное описание реальной системы	15 17 11
1	–	10	Экономическое обоснование разработки и внедрения системы	3 3	Постановка задачи	7 6 7	Проектная документация	4 19 7
2	–	3	Концептуальное описание системы, разработка концептуальной модели	4 4	Содержательное описание реальной системы	7 8	Концептуальная модель	22 11
3	–	4	Формализованное описание	5	Концептуальная модель		Формальная модель	
3	–	5	Сбор и анализ исходных данных	5	Концептуальная модель		Исходные данные	
4	–	6	Разработка (программирование) имитационной модели	6 7	Формальная модель		Имитационная модель	
4	–	7	Определение критериев эффективности и управляющих параметров		Формальная модель		План направленного вычислительного эксперимента	
5	–	6	Параметризация компонентов модели		Исходные данные		Имитационная модель	
5	–	7	Планирование направленного вычислительного эксперимента		Исходные данные		План направленного вычислительного эксперимента	
6	–	7	Оценка адекватности и верификация имитационной модели		Имитационная модель		План направленного вычислительного эксперимента	
7	–	8	Проведение исследования на имитационной модели		План направленного вычислительного эксперимента		Выходная статистика	

8	–	9 Анализ и интерпретация результатов моделирования	8 9	Выходная статистика	9	Результаты исследования, выводы	9
9	–	10 Документирование проекта	10	Результаты исследования, выводы	10	Проектная документация	6
10	–	11 Апробация и внедрение системы, ее модернизации, ликвидация ИС		Проектная документация , акты	11	Сдана в эксплуатацию	6

Приведенный пример весьма условен, так как в реальной жизни ИС в сферах образования и науки могут происходить различные важнейшие видоизменения и события, например, сертификация программных средств проекта, работы по интеграции разработанной ИС в иную систему (мегасистему), масштабирование и многое другое, что неизбежно влечет корректировку сетевого графика работ по проекту на протяжении полного жизненного цикла информационных систем. В практике курсового и дипломного проектирования ИС по кафедре ТИССУ МИРЭА перед проектировщиками ставятся также такие обязательные для всех задачи, как расчет и обеспечение надежности и защищенности ИС, обеспечение сетевого взаимодействия, в том числе в сети Интернет, публикуемость результатов работ и изготовление демоверсии проектируемого продукта и другие актуальные для образовательных нужд и технологий задачи (см. Методические указания кафедры ТИССУ МИРЭА по курсовому проектированию по дисциплине «Проектирование ИС»). Последовательность реализации таких задач также должна найти отражение в сетевом графике разработки.

Ожидаемая продолжительность выполнения работ определяется на основе пессимистической, оптимистической и вероятной оценок как:

$$t_{ож}=(t_{мин} + 4*t_{вер} + t_{макс})$$

$$Д_{исп}=((t_{мин} – t_{макс})/6)^2.$$

Построение сетевого графика, как правило, является начальным моментом осуществления работ: главная задача – это последующая оптимизация графика с целью повышения общей экономической эффективности всего цикла «проектирование – реализация – внедрение».

Методика расчета продолжительности выполнения разработки по сетевым графикам основана на оценке так называемого критического пути. Любая последовательность работ в сетевом графике, в которой конечное событие каждой работы совпадает с начальным событием следующей за ней работы, называется путем.

Путь сетевого графика, имеющий начало в исходном событии, а конец в завершающем, называется полным путем. Путь, обладающий максимальной длительностью из всех имеющихся полных путей, называется критическим. Критический путь показывает время, необходимое для выполнения всего комплекса работ по проекту. Вероятность завершения работ в срок определяется как:

$$\mathbf{P} = \Phi(z), \quad z = \frac{дир \quad крит}{\&^{o2}_{крит}}$$

где z - аргумент нормальной функции распределения вероятностей;
Тдир - директивный срок завершения комплекса работ;

$T_{крит}$ – ожидаемый ранний срок завершения комплекса работ.

Обычно оптимизация начинается с процедуры выравнивания сетевого графика – «снятия» ресурсов с работ, не лежащих на критическом пути, и «переброску» их на работы, лежащие на критическом пути, – и так до тех пор, пока все пути не станут критическими. При этом принято считать, что снятие единицы ресурса с работы приводит к ее увеличению на единицу времени, а назначение ресурса к ее сокращению на единицу времени. Предполагается также, что нельзя сокращать или увеличивать работу более, чем вдвое, а переброска ресурсов с одной работы на другую ведет к увеличению стоимости работ пропорционально количеству переброшенных ресурсов.

На основе полученных результатов оптимизации сетевого графика составляется календарный план хода разработки и внедрения системы, который включается как обязательная составная часть в состав технического задания (ТЗ) на проектирование информационной системы (ИС).

Ниже приведен пример календарного графика проекта в табличной форме, как и сетевой график предполагающий возможности расширений и корректировок.

Календарный план разработки и внедрения системы, связанной с реализацией имитационного моделирования образовательных технологий (условно)

Код			**Наименование работы**	**Начало**	**Окончание**
0	– 1		Анализ проблемной области, ознакомление с прототипами, оценка целесообразности создания информационной системы (системы, ИС)		
1	–	2	Формулирование проблемы, определение целей моделирования		
1	–	10	Экономическое обоснование разработки и внедрения системы		
2	–	3	Концептуальное описание системы, разработка концептуальной модели		
3	–	4	Формализованное описание		
3	–	5	Сбор и анализ исходных данных		
4	–	6	Разработка (программирование) имитационной модели		
4	–	7	Определение критериев эффективности, воздействующих и управляющих параметров		
5	–	6	Параметризация и вертификация компонентов модели		
5	–	7	Планирование направленного вычислительного эксперимента		
6	–	7	Оценка адекватности и верификация имитационной модели, морфологический анализ и решение задач минимизации (рационализации) модели		
7	–	8	Проведение исследования на имитационной модели		
8	–	9	Анализ и интерпретация результатов моделирования		
9	–	10	Документирование проекта, включая выпуск инструкций пользователя и программиста		
10	–	11	Внедрение, модернизации, эксплуатация и ликвидация системы		

Календарный план проекта подкрепляется расчетом стоимости этого проекта, где в первую очередь оцениваются и, по возможности, сокращаются затраты на эксплуатацию помещений и энергетических установок, приобретение, установку и эксплуатацию специального оборудования и программных средств (ПС), обучение и задействование в проект персонала, но не в ущерб качеству и эффективности проекта, уровню подготовки участников, современности оборудования и ПС, лицензионных чистоты и права используемых ПС и обеспечению мер по безопасности труда.

Для проектов, связанных с некоторой производственной отдачей, оценивается и достигается максимально возможная экономическая рентабельность, но опять же не в ущерб вышеприведенному перечню ограничений. При этом нельзя забывать о социальном и научном значении создания и использования высокоэффективных ИС в образовании и науки, об их высокой сложности и малой изученности как объекта системного анализа, поскольку такие системы относятся к сложным слабоструктурированным социально-экономическим системам с множеством прямых и обратных связей, имеющих нелинейный характер. Поведение таких систем сложно предсказуемо и не всегда согласуется с нашим жизненным опытом и интуицией.

Список использованных источников литературы

1. Форрестер Дж. Динамика развития города. – М., Прогресс, 1974г. – 285 стр.;
2. Форрестер Дж. Мировая динамика – М., Наука, 1978г.;
3. Лычкина Н.Н. Системы принятия решений в задачах социально-экономического развития регионов – Компьюлог №2(32) – М., 1999г.;
4. Лычкина Н.Н. Моделирование социально-экономического развития регионов; материалы научно-практического семинара кафедры информационных систем, под ред. Ю.М. Черкасова, 2000 г.,
5. Лычкина Н.Н. Современные тенденции в имитационном моделировании. – Вестник университета, серия Информационные системы управления №2 – М., ГУУ., 2000г.;
6. Попков Ю.С., Посохин М.В. и др. Системный анализ и проблемы развития городов. – 1983г.;
7. Кугаенко А.А. Основы теории и практики динамического моделирования социально-экономических объектов и прогнозирования их развития. – М., Вузовская книга 1998г. – 392 стр.;
8. Ефимова М.Р., Петрова Е.В., Румянцев В.Н. Общая теория статистики: Учебник. – М.:ИНФРА-М, 1998г.;
9. Введение в OLAP и многомерные базы данных. – Михаил Альперович;
10. Data Mining – интеллектуальный анализ данных. – В.А.Дюк, Санкт-Петербургский институт информатики и автоматизации РАН;
11. Способы аналитической обработки данных для поддержки принятия решений. – Л. В. Щавелёв;
12. Оперативная аналитическая обработка данных: концепции и технологии. – Л.В. Щавелёв, Ивановский государственный энергетический университет;
13. Создание систем поддержки принятия решений на основе хранилищ данных. – В.Львов;
14. Митичкин С.А., статья «Компьютерное моделирование развития городов» журнал «Новые рынки», №2, 2002 г
15. Митичкин С.А., тезисы выступления на научно-практической конференции молодых ученых «Проблемы управления в России», секция Информационные системы в управлении, 2002 год
16. Митичкин С.А. Дипломный проект на тему «Компьютерное моделирование жилищно-коммунальной сферы» (рук. доц. Лычкина Н.Н.).

Контрольные вопросы

1. Что такое Информатика?
2. Что называют «Теоретической Информатикой»?
3. На какие традиционные науки опирается информатика?
4. Что такое Информация?
5. Назовите формы представления информации.
6. Что такое Информационная Система?
7. Что называют Системой?
8. Какие термины применяют для описания процессов, протекающих в системе?
9. Что такое «Данные»?
10. Что такое «База данных»?
11. Что такое «База знаний»?
12. Какие виды связей бывают в базах знаний?
13. Что такое «Общность»?
14. Что такое «Банк данных»?
15. Что такое «Безопасность данных»?
16. Что такое «Агрегат данных»?
17. Назовите и опишите два типа агрегатов данных.
18. Что такое «Обучающая система»?