МИНИСТЕРСТВО КУЛЬТУРЫ И ИСКУССТВ УКРАИНЫ

ХАРЬКОВСКАЯ ГОСУДАРСТВЕННАЯ АКАДЕМИЯ КУЛЬТУРЫ

Кафедра информационно-документных систем

Основы проектирования информационных систем

Учебно-методические материалы

Харьков, ХГАК, 2005

УДК 004.414.2(072)
ББК 73.068.4р30-2
О-75

Печатается по решению совета факультета документоведения и информационной деятельности
(протокол № 10 от 4 апреля 2005 р.)

Рекомендовано кафедрой информационно-документных систем
(протокол №12 от 3 марта 2005 р.)

Составитель:
Н. С. Кравец

Основы проектирования информационных систем:
С76 Учебно-методические материалы / Харьк. гос. акад. культуры; Сост.: Н. С. Кравец — Х.: ХГАК, 2005.— 28 с.

УДК 004.414.2(072)
ББК 73.068.4р30-2

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Бурная информатизация общества, автоматизация технологических процессов, широкое использование вычислительной техники, средств связи и телекоммуникаций ставит перед современным менеджером, инженером и служащим целый комплекс взаимосвязанных задач по повышению эффективности бизнес - процессов принятия и выполнения решений.

На сегодня без использования современных автоматизированных информационных управляющих систем трудно представить себе ни учебный процесс в школе, институте, университете, ни эффективную работу практически в любой фирме, на предприятии, в банке или в госучреждении. И практически везде информационная система представляет собой интегрированную систему, ядро которой составляет база данных.

Задачи, связанные с обработкой данных, распространены в любой сфере деятельности. Они ведут учет товаров в супермаркетах и на складах, начисляют зарплату в бухгалтериях и т. д.. Невозможно представить себе деятельность современного предприятия или организации без использования автоматизированных информационных систем. Эти системы составляют фундамент информационной деятельности во всех сферах, начиная с производства, управления финансами и телекоммуникациями и заканчивая управлением семейным бюджетом. Поэтому современный специалист в области информационных технологий и документоведения должен знать основные принципы, методы и средства построения и принципы функционирования автоматизированных информационных систем, чтобы иметь возможность эффективно использовать их в своей профессиональной деятельности.

Данные учебно-методические материалы относятся к курсу «Основи проектування інформаційних систем» для студентов специальности «Документознавство та інформаційна діяльність».

1. ОБЩЕТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

1.1. Теоретические сведения

Проектирование информационных систем всегда начинается с определения цели проекта. Основная задача любого успешного проекта заключается в том, чтобы на момент запуска системы и в течение всего времени ее эксплуатации можно было обеспечить:

– требуемую функциональность системы и степень адаптации к изменяющимся условиям ее функционирования;

– требуемую пропускную способность системы;

– требуемое время реакции системы на запрос;

– безотказную работу системы в требуемом режиме, иными словами готовность и доступность системы для обработки запросов пользователей;

– простоту эксплуатации и поддержки системы;

– необходимую безопасность.

Производительность является главным фактором, определяющим эффективность системы. Хорошее проектное решение служит основой высокопроизводительной системы.

Проектирование информационных систем охватывает три основные области:

– проектирование объектов данных, которые будут реализованы в базе данных;

– проектирование программ, экранных форм, отчетов, которые будут обеспечивать выполнение запросов к данным;

– учет конкретной среды лил технологии, а именно: топологии сети, конфигурации аппаратных средств, используемой архитектуры, параллельной обработки, распределенной обработки данных.

В реальных условиях проектирования — это поиск способа, который удовлетворяет требованиям функциональности системы средствами имеющихся технологий с учетом заданных ограничений.

К любому проекту предъявляется ряд абсолютных требований, например, максимальное время разработки проекта,

максимальные денежные вложения в проект и т.д. Одна из сложностей проектирования состоит в том, что оно не является такой структурированной задачей, как анализ требований к проекту или реализация того или иного проектного решения.

Подходы к проектированию информационных систем

При проектировании информационных систем используют локальный или системный подходы.

Сущность **локального** подхода к проектированию состоит в последовательном наращивании задач, решаемых в системе управления. В этом случае проектирование информационной системы состоит из решения задач, ориентированных на удовлетворение потребностей конкретных подразделений или требований, связанных с реализацией конкретных условий управления. При этом данные организовывают в отдельные логически структурированные файлы. Этот метод имеет серьезные недостатки:

- избыточность информации;
- противоречивость;
- недостаточная скорость обработки;
- отсутствие гибкости;
- низкая стандартизация программного обеспечения.

Системный подход, будучи общей методологической базой проектирования информационных систем, основан на концепции интеграции данных, которые описывают все сферы деятельности объекта управления. Этот метод предусматривает рассмотрение все элементов и составляющих процесса проектирования в их взаимосвязи, взаимозависимости и взаимном влиянии в интересах оптимального достижения как отдельных, так и общих целей создания информационной системы. Данный подход исходит из обязательной необходимости анализа элементов и составляющих процесса проектирования в их взаимосвязи на основе широкого использования современных методов исследования.

Классификация методов проектирования автоматизированных информационных систем

Методы проектирования информационных систем — это

разные способы их создания, которые поддерживаются соответствующими средствами проектирования.

Рис. 1 Классификация методов проектирования информационных систем.

Декомпозиция

В 60-70-е годы было разработано много методов, помогающих справиться с растущей сложностью программ. Наибольшее распространение получило структурное проектирование по методу сверху вниз. Метод был непосредственно основан на топологии традиционных языков высокого уровня типа FORTRAN или COBOL. В этих языках основной базовой единицей является подпрограмма, и программа в целом принимает форму дерева, в котором одни подпрограммы в процессе работы вызывают другие подпрограммы. Структурное проектирование использует именно такой подход:

алгоритмическая декомпозиция применяется для разбиения большой задачи на более мелкие.

Тогда же стали появляться компьютеры еще больших, поистине гигантских возможностей. Значение структурного подхода осталось прежним, но как замечает Стейн, "оказалось, что структурный подход не работает, если объем программы превышает приблизительно 100000 строк". В последнее время появились десятки методов, в большинстве которых устранены очевидные недостатки структурного проектирования. Большинство этих методов представляют собой вариации на одни и те же темы. Их можно разделить на три основные группы:

1. метод структурного проектирования сверху вниз;
2. метод потоков данных;
3. объектно-ориентированное проектирование.

В каждом из этих подходов присутствует алгоритмическая декомпозиция. Следует отметить, что большинство существующих программ написано, по-видимому, в соответствии с одним из этих методов. Тем не менее, структурный подход не позволяет выделить абстракции и обеспечить ограничение доступа к данным; он также не предоставляет достаточных средств для организации параллелизма. Структурный метод не может обеспечить создание предельно сложных систем, и он, как правило, неэффективен в объектных и объектно-ориентированных языках программирования.

В методе потоков данных программная система рассматривается как преобразователь входных потоков в выходные. Метод потоков данных, как и структурный метод, с успехом применялся при решении ряда сложных задач, в частности, в системах информационного обеспечения, где существуют прямые связи между входными и выходными потоками системы и где не требуется уделять особого внимания быстродействию.

Классификация средств разработки информационных систем

Среди средств разработки информационных приложений можно выделить следующие основные группы:

– традиционные системы программирования;

– инструменты для создания файл-серверных приложений;

– средства разработки приложений клиент-сервер;

– средства автоматизации делопроизводства и документооборота;

– средства разработки Internet/Intranet-приложений;

– средства автоматизации проектирования приложений.

Рассмотрим кратко отличительные черты и область применения каждой группы инструментальных средств.

Традиционные системы программирования представлены средствами создания приложений на языках третьего поколения 3GL: C, Pascal, Basic и др. Среди них по способам подготовки и выполнения программных модулей различают системы компилирующего и интерпретирующего типа. Инструментальные средства программирования могут быть представлены набором отдельных утилит (редактор текстов, компилятор, компоновщик и отладчик) или интегрированной средой программирования.

Процедурные языки программирования являются традиционными, они лишь претерпели изменения от неструктурных до структурных языков программирования. Объектно-ориентированное программирование - сравнительно новое направление, однако оно в концептуальном плане более привлекательно, позволяет рассматривать и реализовывать информационные и функциональные свойства объектов в неразрывной связи. Свойствами объектно-ориентированных языков, обуславливающими их преимущества, являются сокрытие деталей реализации объекта (инкапсуляция), наследование процедурных и информационных частей от объектов-родителей, полиморфизм как возможность настройки на различные типы данных и др. Примерами объектно-ориентированных систем программирования являются C++ и Object Pascal.

Основой разработки **файл-серверных приложений** для локальных сетей ПК является инструментальное окружение различных "персональных" СУБД: FoxPro, Clipper, Paradox, Clarion, Paradox, dBase и т. п. Такие средства, как правило, реализованы в виде диалоговой интегрированной среды, предоставляющих три уровня доступа.

Группу инструментальных средств для создания информационных приложений с архитектурой **клиент-сервер** можно разделить на следующие подгруппы: среды разработки приложений для серверов баз данных, независимые от СУБД инструменты для создания приложений клиент-сервер, средства поддержки распределенных информационных приложений.

В качестве примера можно назвать инструменты Informix/4GL, Oracle*Forms и др.

В группе средств **автоматизации делопроизводства** и документооборота можно выделить следующие подгруппы:

– средства автоматизации учрежденческой деятельности Office Automation;

– системы управления электронными документами EDMS;

– средства обеспечения коллективной работы Groupware;

– средства автоматизации документооборота Workflow.

Новый быстро развивающийся сектор **Internet/Intranet-приложений** включает следующие группы средств разработки информационных систем:

– традиционные средства разработки гипертекстовых информационных систем для публикации и поиска документов (HTML; программы подготовки и встраивания гипермедиа-материалов (графики, аудио, видео); браузеры - программы просмотра и интерпретации гипертекста и гипермедиа; модули расширения браузеров);

– средства для организации шлюза в другие приложения из традиционных Internet/Intranet-приложений (интерфейсы CGI и API Web-серверов);

– развитые средства программирования Internet/Intranet-приложений (Java, JavaScript, Tcl и др.).

Средства автоматизации проектирования приложений (CASE-средства) предназначены для анализа предметной области, для проектирования и генерации программ информационных приложений. Могут существовать в виде отдельных утилит или интегрированной среды проектирования. Системы CASE реализуют либо структурные, либо объектно-ориентированные методы анализа, проектирования и программирования. Кроме того, для каждой методики

проектирования несколько различаются нотации схематических описаний. Различают следующие виды систем автоматизации проектирования приложений: независимые CASE-системы (например IDEF/Design); системы, интегрированные с СУБД (WestmountI-CASE for Informix и Oracle Designer/2000); системы проектирования БД (SILVERRUN и ERWin/ERX).

Характерными особенностями CASE-систем являются: наличие графических редакторов схем проекта, хранение описаний проектов в репозитарии объектов и генерация описания структуры и свойств БД, а также модулей приложения.

Инструментальные программные средства разработки информационных приложений должны обеспечивать следующие важные свойства:

- поддержку многоплатформенности;
- независимость от производителя;
- унификацию средств разработки;
- создание надежного и качественного программного обеспечения;
- поддержку разработанного ПО на протяжении всего времени жизни;
- проектирование с использованием различных современных методик;
- ведение версий;
- поддержку Web-технологии.

В условиях быстрой эволюции вычислительной техники, смены операционных систем, схода с рынка ряда фирм-производителей, не вполне ясных альтернатив "клиент-сервер или мейнфрейм", "Unix или Windows NT", "закрытые сети или Internet" устойчивость информационно-вычислительной среды может быть обеспечена разумным консерватизмом в сочетании с трезвым анализом тенденций современного рынка.

1.2. Порядок выполнения лабораторной работы

1. Построить схему алгоритма проектирования АИС для заданной предметной области методами:

- сверху-вниз;
- модульным методом;
- структурным методом.

2. Сравнить преимущества и недостатки полученных схем.

1.3. Вопросы для самостоятельной проработки

1. Методы и средства проектирования информационных систем, их содержание, классификация, преимущества и недостатки.
2. Методы и модели принятия проектных решений.
3. Методы оценки качества АИС.

2. ТЕХНОЛОГИЧЕСКИЕ АСПЕКТЫ ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

2.1. Теоретические сведения

Таблица 1. Этапы разработки информационной системы

№	Наименование этапа	Основные характеристики
1	**Разработка и анализ бизнес - модели**	Определяются основные задачи АИС, проводится декомпозиция задач по модулям, и определяются функции, с помощью которых решаются эти задачи. **Метод решения**: Функциональное моделирование. **Результат**: Концептуальная модель АИС. Аппаратно-технический состав АИС.
2	**Формализация бизнес - модели, разработка логической модели бизнес -процессов.**	Разработанная концептуальная модель формализуется, т.е. воплощается в виде логической модели АИС. **Метод решения**: Разработка диаграммы "сущность-связь" (ER (Entity-Reationship) - CASE- диаграммы). **Результат:** Разработанное информационное обеспечение АИС: схемы и структуры данных для всех уровней модульности АИС, документация по логической структуре АИС.
3	**Выбор лингвистического обеспечения, разработка программного**	Разработка АИС. Разработанная на втором этапе логическая схема воплощается в реальные объекты. **Метод решения:** Разработка

	обеспечения АИС.	программного кода с использованием выбранного инструментария. **Результат**: Работоспособная АИС.
4	**Тестирование и отладка АИС**	Корректировка информационного, аппаратного, программного обеспечения; разработка методического обеспечения (документации разработчика, пользователя) и т.п. **Результат**: Оптимальный состав и эффективное функционирование АИС. Комплект документации: разработчика, администратора, пользователя.
5	**Эксплуатация и контроль версий**	Контроль версий, т.е. добавление новых и развитие старых модулей с выводом из эксплуатации старых. **Результат**: Наращиваемость и безизбыточный состав гибкой, масштабируемой АИС.

ER-диаграммы

На втором этапе проектирования информационной системы производится разработка логической модели бизнес-процессов. На этом этапе вся информация, собранная о системе, формализуется и уточняется. Информация собирается и фиксируется в двух взаимосвязанных формах:

– функции — информация о событиях и процессах;

– сущности — информация о вещах, имеющих значение для организации и о которых что-то известно.

Двумя классическими результатами анализа являются:

– иерархия функций, которая разбивает процесс обработки на составные части;

– модель "сущность-связь" (Entry Relationship model, ER-модель), которая описывает сущности и их атрибуты и связи (отношения) между ними.

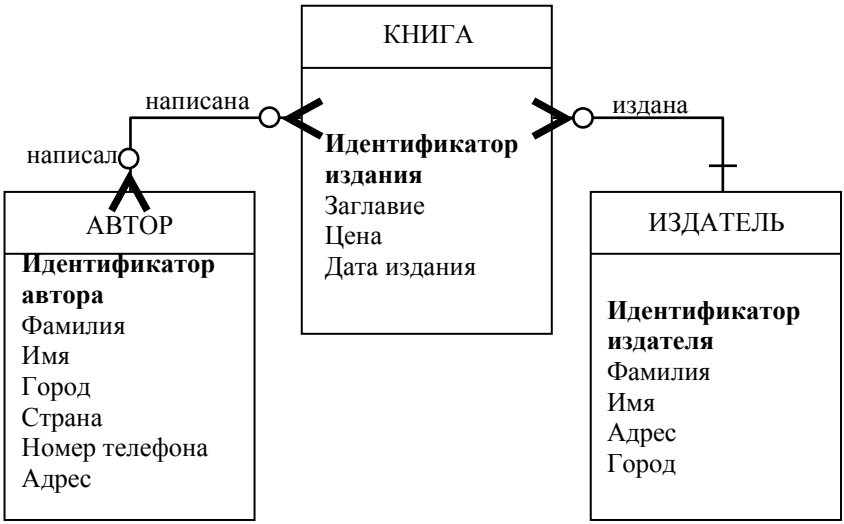
При проведении структурного анализа часто применяются такие методологии:

– Диаграммы "сущность-связь" (Entity relationship Diagrams, ERD), которые служат для формализации информации о сущностях и их отношениях.

– Диаграммы потоков данных (Data Flow Diagrams, DFD), которые служат для формализации представления функций системы.

– Диаграммы переходов состояний (State Transition Diagrams, STD), которые отражают поведение системы, зависящее от времени.

ER-диаграмма содержит информацию о сущностях системы и способах их взаимодействия, включает идентификацию объектов, важных для предметной области (сущностей), свойств этих объектов (атрибутов) и их отношений с другими объектами (связей). Во многих случаях модель сложна и содержит



множество объектов.

Рис. 2 Пример ER-диаграммы

Сущность изображается в виде прямоугольника, вверху которого располагается имя сущности, например, "Издание". В прямоугольнике могут быть перечислены атрибуты сущности, атрибуты, набранные полужирным шрифтом, являются ключевыми.

Отношение изображается линией между двумя сущностями. Одиночная линия означает "один", "птичья лапка" — "многие"; вертикальная черта означает "обязательно", а кружок — "не обязательно".

2.2. Порядок выполнения лабораторной работы

1. Построить концептуальную модель АИС.
2. Формализовать концептуальную модель АИС с помощьюER-диаграмм.

2.3. Вопросы для самостоятельной проработки

1. Участники процесса проектирования, их права и обязанности.
2. Предпроектная документация.
3. Технология технорабочего проектирования АИС.
4. Содержание проектной документации на информационную систему.
5. Менеджмент проекта информационной системы.

3. ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ НА ОСНОВЕ ПЕРСПЕКТИВНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

3.1. Теоретические сведения

Одна из современных тенденций развития информационных технологий состоит в создании специализированных инструментальных средств, ориентированных на определенный класс задач, которые могут быть интегрированы в общий состав информационной системы. В таком случае решение (проектирование) задачи (комплекса задач) с помощью специализированного инструментального средства значительно упрощается. К таким инструментальным средствам относятся электронные таблицы.

Электронная таблица является очень эффективным средством проведения численного моделирования ситуации или объекта. Изменяя в различных сочетаниях значения входных параметров, можно наблюдать за изменением расчетных

параметров и анализировать результат.

Задачи, решаемые с помощью табличного процессора:

– расчеты по установленным форматам в регламентном режиме, когда один раз определяется шаблон таблицы, а после этого выполняют расчеты, изменяя данные;

– моделирование результатов принятия решения, по типу "что будет, если" (формулами задаются зависимости, а по результатам нескольких расчетов выбирается оптимальный вариант);

– представление табличных данных в графической форме;

– использование табличного процессора как большого матричного калькулятора (например, для статистического анализа).

3.2. Порядок выполнения лабораторной работы

1. Создание проекта АИС с помощью электронных таблиц и его оценка.

3.3. Вопросы для самостоятельной проработки

1. Технология проектирования информационных систем с использованием электронных таблиц.
2. Интеграция электронных таблиц с другими инструментальными средствами информационной системы.
3. Особенности электронных таблиц нового поколения.
4. Технология проектирования информационных систем на сетях ЭВМ.

4. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОЕКТИРОВАНИЕ

4.1. Теоретические сведения

Объектно-ориентированное проектирование

В основе объектно-ориентированного проектирования (object-oriented design, OOD) лежит представление о том, что программную систему необходимо проектировать как совокупность взаимодействующих друг с другом объектов, рассматривая каждый объект как экземпляр определенного класса, причем классы образуют иерархию. Объектно-ориентированный подход отражает топологию новейших языков

высокого уровня, таких как Smalltalk, Object Pascal, C++ и др..

Объектно-ориентированное проектирование — это методология проектирования, соединяющая в себе процесс объектной декомпозиции и приемы представления логической и физической, а также статической и динамической моделей проектируемой системы.

В данном определении содержатся две важные части: объектно-ориентированное проектирование 1) основывается на объектно-ориентированной декомпозиции; 2) использует многообразие приемов представления моделей, отражающих логическую (классы и объекты) и физическую (модули и процессы) структуру системы, а также ее статические и динамические аспекты.

Именно объектно-ориентированная декомпозиция отличает объектно-ориентированное проектирование от структурного; в первом случае логическая структура системы отражается абстракциями в виде классов и объектов, во втором — алгоритмами.

Объектно-ориентированная технология основывается на так называемой **объектной модели**. Основными ее принципами являются: абстрагирование, инкапсуляция, модульность, иерархичность, типизация, параллелизм и сохраняемость. Каждый из этих принципов сам по себе не нов, но в объектной модели они впервые применены в совокупности.

Абстракция и инкапсуляция дополняют друг друга: **абстрагирование** направлено на наблюдаемое поведение объекта, а инкапсуляция занимается внутренним устройством. Чаще всего инкапсуляция выполняется посредством скрытия информации, то есть маскировкой всех внутренних деталей, не влияющих на внешнее поведение. Обычно скрываются и внутренняя структура объекта, и реализация его методов.

Инкапсуляция, таким образом, определяет четкие границы между различными абстракциями. Инкапсуляция - это процесс отделения друг от друга элементов объекта, определяющих его устройство и поведение; инкапсуляция служит для того, чтобы изолировать контрактные обязательства абстракции от их реализации.

Модульность — это свойство системы, которая была

разложена на внутренне связные, но слабо связанные между собой модули.

Иерархия — это упорядочение абстракций, расположение их по уровням. Основными видами иерархических структур применительно к сложным системам являются структура классов (иерархия "is-a") и структура объектов (иерархия "part of").

Типизация — это способ защититься от использования объектов одного класса вместо другого, или, по крайней мере, управлять таким использованием.

Сохраняемость — способность объекта существовать во времени, переживая породивший его процесс, и (или) в пространстве, перемещаясь из своего первоначального адресного пространства.

Унифицированный язык моделирования UML

UML (Unified Modeling Language) — это язык для визуализации, специфицирования, конструирования и документирования артефактов программных систем.

UML пригоден для моделирования любых систем: от информационных систем масштаба предприятия до распределенных Web-приложений и даже встроенных систем реального времени. Его концептуальная модель включает в себя три основных элемента: базовые строительные блоки, правила, определяющие как эти блоки могут сочетаться между собой, и некоторые общие механизмы языка.

Принципы моделирования

Использование языка UML основывается на следующих общих принципах моделирования:

1. ***абстрагирование*** - в модель следует включать только те элементы проектируемой системы, которые имеют непосредственное отношение к выполнению ей своих функций или своего целевого предназначения. Другие элементы опускаются, чтобы не усложнять процесс анализа и исследования модели;
2. ***многомодельность*** - никакая единственная модель не может с достаточной степенью точности описать различные аспекты

системы. Допускается описывать систему некоторым числом взаимосвязанных представлений, каждое из которых отражает определенный аспект её поведения или структуры;

3. ***иерархическое построение*** – при описании системы используются различные уровни абстрагирования и детализации в рамках фиксированных представлений. При этом первое представление системы описывает её в наиболее общих чертах и является представлением концептуального уровня, а последующие уровни раскрывают различные аспекты системы с возрастающей степенью детализации вплоть до физического уровня. Модель физического уровня в языке UML отражает компонентный состав проектируемой системы с точки зрения ее реализации на аппаратурной и программной платформах конкретных производителей.

Строительные блоки UML:

- сущности;
- отношения;
- диаграммы.

Сущности в UML

В UML определены четыре типа сущностей: ***структурные***, ***поведенческие***, ***группирующие*** **и** ***аннотационные***. Сущности являются основными объектно-ориентированными элементами языка, с помощью которых создаются модели.

Структурные сущности - это имена существительные в моделях на языке UML. Как правило, они представляют статические части модели, соответствующие концептуальным или физическим элементам системы. Примерами структурных сущностей являются «класс», «интерфейс», «кооперация», «прецедент», «компонент», «узел», «активный класс».

Поведенческие сущности являются динамическими составляющими модели UML. Это глаголы, которые описывают поведение модели во времени и в пространстве. Существует два основных типа поведенческих сущностей:

– ***взаимодействие*** - это поведение, суть которого заключается в обмене сообщениями между объектами в рамках конкретного контекста для достижения определенной цели;

– ***автомат*** - алгоритм поведения, определяющий последовательность состояний, через которые объект или взаимодействие проходят в ответ на различные события.

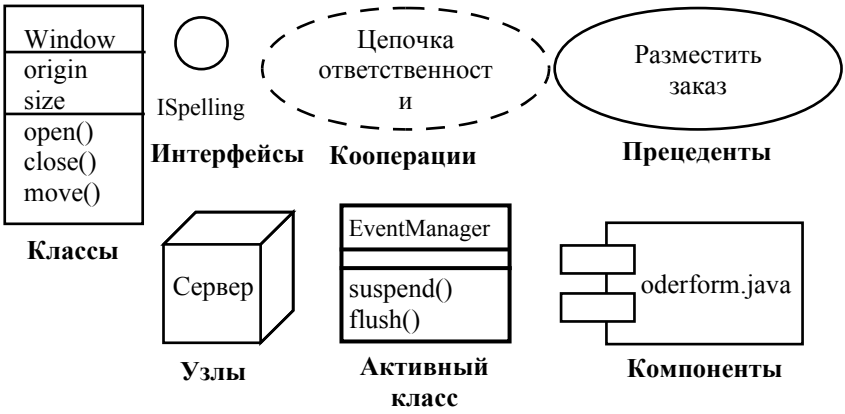


Рис. 3 Примеры структурных сущностей

Группирующие сущности являются организующими частями модели UML. Это блоки, на которые можно разложить модель. Такая первичная сущность имеется в единственном экземпляре - это пакет.

Пакеты представляют собой универсальный механизм организации элементов в группы. В пакет можно поместить структурные, поведенческие и другие группирующие сущности. В отличие от компонентов, которые реально существуют во время работы программы, пакеты носят чисто концептуальный характер, то есть существуют только в процессе разработки.

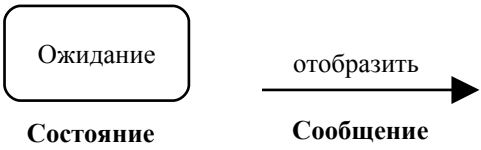


Рис. 4 Изображение поведенческих сущностей

Аннотационные сущности – это пояснительные части модели UML: комментарии для дополнительного описания, разъяснения или замечания к любому элементу модели. Имеется только один базовый тип аннотационных элементов - примечание. Примечание используют, чтобы снабдить диаграммы комментариями или ограничениями, выраженными в виде неформального или формального текста.

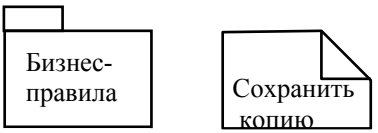


Рис. 5 Примеры группирующей и аннотационной сущностей

Отношения в UML

В языке UML определены следующие типы отношений: ***зависимость***, ***ассоциация***, ***обобщение*** и ***реализация***. Эти отношения являются основными связующими конструкциями UML и также как сущности применяются для построения моделей.

Зависимость (dependency) - это семантическое отношение между двумя сущностями, при котором изменение одной из них, независимой, может повлиять на семантику другой, зависимой.

Ассоциация (association) - структурное отношение, описывающее совокупность смысловых или логических связей между объектами. Агрегирование — частный случай ассоциации, предназначенный для моделирования отношения типа "часть\целое".

Обобщение (generalization) - это отношение, при котором объект специализированного элемента (потомок) может быть подставлен вместо объекта обобщенного элемента (предка). При этом, в соответствии с принципами объектно-ориентированного программирования, потомок (child) наследует структуру и поведение своего предка (parent).

Реализация (realization) является семантическим отношением между классификаторами, при котором один

классификатор определяет обязательство, а другой гарантирует его выполнение. Отношение реализации встречаются в двух случаях:

– между интерфейсами и реализующими их классами или компонентами;

– между прецедентами и реализующими их кооперациями.

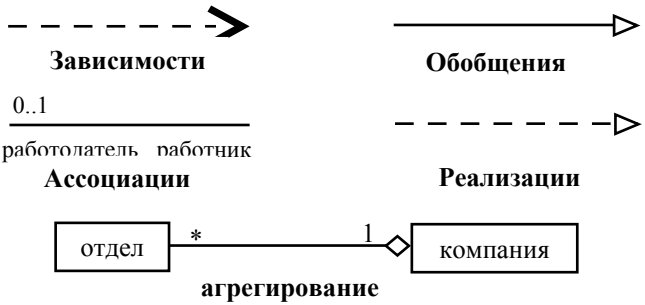


Рис. 6 Примеры отношений

Общие механизмы UML

Для точного описания системы в UML используются, так называемые, ***общие механизмы***:

– ***спецификации*** (specifications);
– ***дополнения*** (adornments);
– ***деления*** (common divisions);
– ***расширения*** (extensibility mechanisms).

UML является не только графическим языком. За каждым графическим элементом его нотации стоит ***спецификация***, содержащая текстовое представление соответствующей конструкции языка. Например, пиктограмме класса соответствует спецификация, которая описывает его атрибуты, операции и поведение, хотя визуально, на диаграмме, пиктограмма часто отражает только малую часть этой информации. Более того, в модели может присутствовать другое представление этого класса, отражающее совершенно иные его аспекты, но, тем не менее,

соответствующее спецификации. Таким образом, графическая нотация UML используются для визуализации системы, а с помощью спецификаций описывают ее детали.

Практически каждый элемент UML имеет уникальное графическое изображение, которое дает визуальное представление самых важных его характеристик. Нотация сущности «класс» содержит его имя, атрибуты и операции. Спецификация класса может содержать и другие детали, например, видимость атрибутов и операций, комментарии или указание на то, что класс является абстрактным. Многие из этих деталей можно визуализировать в виде графических или текстовых ***дополнений*** к стандартному прямоугольнику, который изображает класс.

При моделировании объектно-ориентированных систем существует определенное ***деление*** представляемых сущностей.

Во-первых, существует деление на классы и объекты. Класс - это абстракция, а объект - конкретное воплощение этой абстракции. В связи с этим, практически все конструкции языка характеризуются двойственностью «класс/объект». Так, имеются прецеденты и экземпляры прецедентов, компоненты и экземпляры компонентов, узлы и экземпляры узлов. В графическом представлении для объекта принято использовать тот же символ, что и для класса, а название подчеркивать.

Во-вторых, существует деление на интерфейс и его реализацию. Интерфейс декларирует обязательства, а реализация представляет конкретное воплощение этих обязательств и обеспечивает точное следование объявленной семантике. В связи с этим, почти все конструкции UML характеризуются двойственностью «интерфейс/реализация». Например, прецеденты реализуются кооперациями, а операции - методами.

UML является открытым языком, то есть допускает контролируемые ***расширения***, чтобы отразить особенности моделей предметных областей. ***Механизмы расширения*** UML включают:

– ***стереотипы*** (stereotype) - расширяют словарь UML, позволяя на основе существующих элементов языка создавать новые, ориентированные для решения конкретной проблемы;

– ***помеченные значения*** (tagged value) - расширяют свойства основных конструкций UML, позволяя включать дополнительную информацию в спецификацию элемента;

– ***ограничения*** (constraints) - расширяют семантику конструкций UML, позволяя создавать новые и отменять существующие правила.

Совместно эти три механизма расширения языка позволяют модифицировать его в соответствии с потребностями проекта или особенностями технологии разработки.

Виды диаграмм UML

Графические изображения моделей системы в UML называются ***диаграммами***. В терминах языка UML определены следующие их виды:

– диаграмма вариантов использования или прецедентов (use case diagram)
– диаграмма классов (class diagram)
– диаграммы поведения (behavior diagrams)
– диаграмма состояний (statechart diagram)
– диаграмма деятельности (activity diagram)
– диаграммы взаимодействия (interaction diagrams)
– диаграмма последовательности (sequence diagram)
– диаграмма кооперации (collaboration diagram)
– диаграммы реализации (implementation diagrams)
– диаграмма компонентов (component diagram)
– даграмма развертывания (deployment diagram)

Каждая из этих диаграмм конкретизирует различные представления о модели системы. При этом, диаграмма вариантов использования представляет концептуальную модель системы, которая является исходной для построения всех остальных диаграмм. Диаграмма классов является логической моделью, отражающей статические аспекты структурного построения системы, а диаграммы поведения, также являющиеся разновидностями логической модели, отражают динамические аспекты её функционирования. Диаграммы реализации служат для представления компонентов системы и относятся к ее физической модели.

Из перечисленных выше диаграмм некоторые служат для

обозначения двух и более подвидов. В качестве же самостоятельных представлений используются следующие диаграммы: ***вариантов использования***, ***классов***, ***состояний***, ***деятельности***, ***последовательности***, ***кооперации***, ***компонентов*** и ***развертывания***.

Для диаграмм языка UML существуют три типа визуальных обозначений, которые важны с точки зрения заключенной в них информации:

– ***связи***, которые представляются различными линиями на плоскости;

– ***текст***, содержащийся внутри границ отдельных геометрических фигур;

– ***графические символы***, изображаемые вблизи визуальных элементов диаграмм.

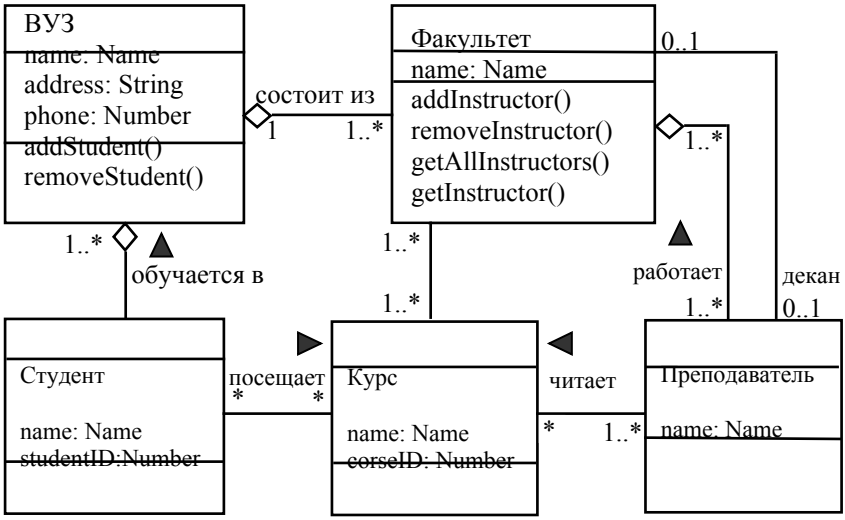


Рис. 7 Пример диаграммы классов UML

При графическом изображении диаграмм рекомендуется придерживаться следующих правил:

– каждая диаграмма должна быть законченным представлением некоторого фрагмента моделируемой предметной области;

– представленные на диаграмме сущности модели должны быть одного концептуального уровня;

– вся информация о сущностях должна быть явно представлена на диаграмме;

– диаграммы не должны содержать противоречивой информации;

– диаграммы не следует перегружать текстовой информацией;

– каждая диаграмма должна быть самодостаточной для правильной интерпретации всех ее элементов;

– количество типов диаграмм, необходимых для описания конкретной системы, не является строго фиксированным и определяется разработчиком;

– модели системы должны содержать только те элементы, которые определен

4.2. Порядок выполнения лабораторной работы

1. Построение для проектируемой АИС с помощью языка UML моделей: сущностей системы, классов, структурных отношений.
2. Построение диаграммы классов.
3. Реализация проекта системы в виде таблиц Exel.

4.3. Вопросы для самостоятельной проработки

1. Методология объектно-ориентированного проектирования.
2. Жизненный цикл АИС.
3. Язык UML (Unified Modeling Language).
4. Концептуальная модель UML.

СПИСОК ЛИТЕРАТУРЫ

ОСНОВНАЯ

1. Проектування інформаційних систем: Посібник/ За редакцією В.С. Пономаренка. ─ К.: Видавничий центр «Академія»; 2002. ─ 488 с. (Альма-матер)
2. Буч. Г. Объектно-ориентированное проектирование с примерами применения. ─М.:Конкорд, 1992. ─ 519 с.
3. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя: Пер. с англ. ─ М.:ДМК, 2000. ─ 432 с.

ДОПОЛНИТЕЛЬНАЯ

1. Береза А. М. Основи створення інформаційних систем: Навч. посібник/ А. М. Береза; Київ. Нац.екон. ун-т. ─ 2-ге вид., перероб. і доп. — К.: КНЕУ, 2001. ─ 214 с.
2. Гринберг А. С. Информационные технологии управления: Учеб. Пособие/ А. С. Гринберг, Н. Н. Горбачёв, А. С. Бондаренко. ─ М.: ЮНИТИ; 2004. ─ 479 с.
3. Компьютеризация информационных процессов на промышленных предприятиях/ В.Ф. Сытник, Х. Срока, Н.В. Ерёмина и др. ─ К.: Техника; Катовице: Экономическая академия, 1991. ─ 216 с.
4. Коноваленко М. К. Стратегическое планирование инноваций: бизнес-план научно-исследовательских и опытно-конструкторских работ (НИОКР).– Харьков: Торсинг.– 1998.– 76 с.
5. Мельник Л. Г. Экономика информации и информационные системы предприятия: Учеб. пособие/ Л. Г. Мельник, С. Н. Ильяшенко, В. А. Касьяненко. ─ Сумы.: Унив. кл, 2004. ─ 399 с.
6. Павленко Л. А. Корпоративні інформаційні системи: Навч. посібник/ Л. А. Павленко. ─ Х.: ВД ІНЖЕК, 2003. ─ 260 с..
7. Сендзюк М. А. Інформаційні системи в державному управлінні: Навч. посібник/ М. А. Сендзюк. ─ Київ. Нац. екон. ун-т., ─ К.: КНЕУ, 2004. ─ 339 с..
8. Системный анализ в экономике и организации производства./Под общ. Ред. С.А. Валуева, Н.В. Волковой. – Л.: Политехника.– 1991.– 396 с.
9. Шегда А. В. Основы менеджмента.– К.:Товаринство «Знання», КОО.– 1998.– 512 с.

28

Учебное издание

Основы проектирования информационных систем

Учебно-методические материалы

С о с т а в и т е л ь :

кандидат технических наук, доцент ***Кравец Н. С.,***

Печатается в авторской редакции

Компьютерный набор и верстка Н. С. Кравец

План 2005.
Подписано к печати 00.00.2005. Формат 60x84/16.
Гарнитура «Times». Бумага для мн. ап. Печать ризограф.
Усл. печ. стр. 0,56. Обл..-вид. Стр.. 0,56. Тираж 100. Зам. № .
ХГАК, 61003, Харьков-3, Бурсацкий спуск 4.
Напечатано в лаб. множ. Техники ХГАК