МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт сервиса, туризма и дизайна (филиал) СКФУ в г. Пятигорске

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ СТУДЕНТОВ ПО ОРГАНИЗАЦИИ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ ПРОЕКТИРОВАНИЕ ИНФОРМАИЦОННЫХ СИСТЕМ

Направление подготовки	**09.03.02 Информационные системы и технологии**
Направленность (профиль)	**Информационные системы и технологии**
Квалификация выпускника	Бакалавр

РАЗРАБОТАНО:

Доцент кафедры СУиИТ

_____ Мартиросян К.В.

«__» _____ 2020 г.

Пятигорск, 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Методические рекомендации содержат перечень тем с вопросами для самостоятельной проработки, перечень лабораторных работ с вопросами для самостоятельной проработки, темы курсового проектирования, вопросы к экзамену.

Методические указания посвящены курсу «Проектирование информационных систем». Проектирование ИС 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 или технологии, а именно: топологии сети, конфигурации аппаратных средств, используемой архитектуры (файл-сервер или клиент-сервер), параллельной обработки, распределенной обработки данных и т.п.

Обычно выделяют следующие этапы создания ИС: формирование требований к системе, проектирование, реализация, тестирование, ввод в действие, эксплуатация и сопровождение/

Начальным этапом процесса создания ИС является моделирование бизнес-процессов, протекающих в организации и реализующих ее цели и задачи. Модель организации, описанная в терминах бизнес-процессов и бизнес-функций, позволяет сформулировать основные требования к ИС. Это фундаментальное положение методологии обеспечивает объективность в выработке требований к проектированию системы. Множество моделей описания требований к ИС затем преобразуется в систему моделей, описывающих концептуальный проект ИС. Формируются модели архитектуры ИС, требований к программному обеспечению (ПО) и информационному обеспечению (ИО). Затем формируется архитектура ПО и ИО, выделяются корпоративные БД и отдельные приложения, формируются модели требований к приложениям и проводится их разработка, тестирование и интеграция.

Целью начальных этапов создания ИС, выполняемых на стадии анализа деятельности организации, является формирование требований к ИС, корректно и точно отражающих цели и задачи организации-заказчика.

На этапе проектирования прежде всего формируются модели данных. Проектировщики в качестве исходной информации получают результаты анализа. Построение логической и физической моделей данных является основной частью проектирования базы данных. Полученная в процессе анализа информационная модель сначала преобразуется в логическую, а затем в физическую модель данных.

Параллельно с проектированием схемы базы данных выполняется проектирование процессов, чтобы получить спецификации (описания) всех модулей ИС. Оба эти процесса проектирования тесно связаны, поскольку часть бизнес-логики обычно реализуется в базе данных (ограничения, триггеры, хранимые процедуры). Главная цель проектирования процессов заключается в отображении функций, полученных на этапе анализа, в модули информационной системы. При проектировании модулей определяют интерфейсы программ: разметку меню, вид окон, горячие клавиши и связанные с ними вызовы.

Конечными продуктами этапа проектирования являются:

- схема базы данных (на основании ER-модели, разработанной на этапе анализа);
- набор спецификаций модулей системы (они строятся на базе моделей функций).

Кроме того, на этапе проектирования осуществляется также разработка архитектуры ИС, включающая в себя выбор платформы (платформ) и операционной системы (операционных систем).

1. ЦЕЛЬ И ЗАДАЧИ ИЗУЧЕНИЯ ДИСЦИПЛИНЫ

Целью освоения дисциплины «Проектирование информационных систем» является формирование набора профессиональных компетенций будущего бакалавра по направлению подготовки 09.03.02 «Информационные системы и технологии» для решения прикладных задач управления проектами.

Задачи освоения дисциплины: изучение основных понятий проектирования информационных систем, освоение технологий проектирования информационных систем, получение навыков работы с инструментами проектирования информационных систем.

2. ТЕМЫ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

№ темы	Наименование тем дисциплины, их краткое содержание	Обьем часов
	8 семестр	
	Раздел 1. Методы и средства проектирования информационных систем	
1	Тема 1. Концептуальное проектирование информационной системы. Лабораторная работа 1. Концептуальное проектирование информационной системы Содержание: Методы и средства проектирования информационной системы. Разработка концепции программного приложения. Техническое задание на проектирование информационной системы. Технико-экономическое обоснование проекта информационной системы.	10
2	Тема 2. Проектирование архитектуры информационной системы Лабораторная работа 2. Проектирование архитектуры информационной системы Содержание: Архитектура программного приложения. Двухуровневая и трехуровневая архитектура клиент-серверной информационной системы. Проектирование функциональных и технологических требований к информационной системе. Проектирование состава обеспечивающих подсистем информационной системы. Проектирование состава функциональных подсистем.	10
	Раздел 2. Инструментальные технологии проектирования информационных систем	
3	Тема 3. Проектирование базы данных информационной системы Лабораторная работа 3. Проектирование базы данных информационной системы. Содержание: Разработка информационного обеспечения программных приложений. Инфологическое моделирование базы данных информационной системы. Проектирование логической и физической структуры базы данных информационной системы.	10
4	Тема 4. Проектирование программного интерфейса информационной системы. Лабораторная работа 4. Проектирование программного интерфейса информационной системы. Содержание: Проектирование программного интерфейса информационной системы Разработка технологий ввода-вывода	12,75

	данных информационной системы. Разработка технологий взаимодействия базы данных и программного интерфейса ИС. Разработка структурной схемы модулей ИС.	
	Итого за 8 семестр	42,75
	Экзамен	20,25
	Итого	63

3. ТЕХНОЛОГИЧЕСКАЯ КАРТА САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩЕГОСЯ

Коды реализуемых компетенций	Вид деятельности студентов	Итоговый продукт самостоятельной работы	Средства и технологии оценки	Объем часов, в том числе		
				СРС	Контактная работа с преподавателем	Всего
ОПК-1, ПК-11, ПК-17	Подготовка к лекциям	Конспект	Собеседование	0,54	0,06	0,60
ОПК-1, ПК-11, ПК-17	Самостоятельное изучение литературы по темам 1,2,5,6	Конспект	Собеседование	25,70	2,86	28,55
ОПК-1, ПК-11, ПК-17	Подготовка к практическим занятиям	Индивидуальное задание	Отчет письменный	3,24	0,36	3,60
ОПК-1, ПК-11, ПК-17	Курсовое проектирование	Курсовой проект	Тематика курсового проектирования	27,00	3,00	30,00
			Экзамен	18,22	2,02	20,25
			Итого	47,70	8,30	83

4. РЕКОМЕНДАЦИИ ДЛЯ САМОПОДГОТОВКИ

4.1 Подготовка к лекциям. Самостоятельное изучение литературы

Тема 1. Концептуальное проектирование информационной системы

Базовый уровень

1. Рабочий проект информационной системы. Дисциплина обязательств.
2. Технический проект информационной системы
3. Управление проектом информационной системы
4. Календарный план разработки информационной системы.
5. Понятие архитектуры информационной системы
6. Открытая архитектура информационных систем
7. Системная инженерия: точка зрения и характеристики точек зрения
8. Множественность точек зрения при разработке информационной системы

9. Характеристики информационной системы. Интерактивность, usability

10. Применение инструментов моделирования информационных систем.

Повышенный уровень

1. Применение инструментов разработки информационных систем. Язык программирования, блок-схема, документирование, средства разработки алгоритмическая, эффективность алгоритма

2. Управление требованиями к информационной системе

3. Виды требований к информационной системе: функциональные требования, нефункциональные требования.

4. Свойства требований к информационной системе: ясность и недвусмысленность, полнота и непротиворечивость, необходимый уровень детализации, прослеживаемость, тестируемость и проверяемость, модифицируемость.

5. Формализация требований к информационной системе. Цикл работы с требованиями

6. Понятие конфигурационного управления проектом информационной системы

7. Управление версиями информационной системы. Понятие "ветки" проекта

8. Управление сборками при разработке информационной системы

9. Средства версионного контроля информационной системы

10. Единицы конфигурационного управления. Понятие baseline.

Тема 2. Проектирование архитектуры информационной системы

Базовый уровень

1. Принципы верификации и тестирования

2. Верификация и валидация программных продуктов

3. Понятие тестирования программных средств

4. Методы верификации объектно-ориентированных программ

5. Качество и надежность программного обеспечения

6. Профили открытых информационных систем

7. Функциональные и технологические стандарты разработки программных комплексов

8. Принципы организации проектирования и программных комплексов

9. Задачи обеспечения качества программных компонентов

10. Методы исследования качества программных компонентов

Повышенный уровень

1. Функциональные и технологические требования к программным комплексам

2. Архитектура программных комплексов для информатизации предприятий,

3. Стандарты проектирования программного обеспечения

4. Стандарты разработки программного обеспечения

5. Методы разработки программных комплексов для решения прикладных задач

6. Методы оценки сложности алгоритмов и программ

7. Методы тестирования и документирования программных комплексов

8. Основные составные информационной системы и их характеристика.

9. Многоуровневая архитектура информационных систем

10. Моделирование структуры информационных систем, виды моделей, их назначение

Тема 3. Проектирование базы данных информационной системы

Базовый уровень

1 Задачи обеспечения надежности программных компонентов

2. Методы исследования надежности программных компонентов

3. Экономико-правовые основы разработки программных продуктов

4. Объектно-ориентированное моделирование, диаграммы вариантов использования.

5. Многопользовательская информационная система с распределенной базой данных

6. Рабочий проект информационной системы. Дисциплина обязательств.

7. Технический проект информационной системы

8. Управление проектом информационной системы

9. Календарный план разработки информационной системы.

10. Понятие архитектуры информационной системы

Повышенный уровень

1. Метрики качества программного обеспечения

2. Стандартный метод оценки значений показателей качества

3. Управление качеством ПО

4. Оптимизация и реинжиниринг

5. Инновационные проекты.

6. Тестирование информационной системы

7. Стандартизация качества информационных систем

8. Методы обеспечения качества информационных систем

9. Понятие тестирования информационной системы. Тестирование черного ящика. Тестирование белого ящика

10. Инструменты тестирования информационных систем

Тема 4. Проектирование программного интерфейса информационной системы

Базовый уровень

1. Критерии тестирования информационных систем

2. Виды тестирования информационных систем

3. Процесс тестирования информационной системы. Работа с ошибками

4. Средства контроля ошибок (bug tracking systems) в ходе тестирования информационной системы

5. Диаграммные техники в работе со знаниями

6. Диаграммы использования. Случаи использования. Работа с требованиями. Случаи использования в управлении разработкой информационной системы

7. Итеративный цикл автор/рецензент. Карты памяти для проекта информационной системы

8. IT решение. Основные принципы MSF

9. Модель команды при работе над проектом информационной системы: основные принципы, ролевые кластеры

10. Масштабирование команды MSF. Модель процесса. Управление компромиссами при разработке информационной системы

Повышенный уровень

1. Разработка информационных систем. Понятие CMMI.

2. Уровни зрелости процессов по CMMI

3. Области усовершенствования в методологии CMMI.

4. Общее описание "гибких" методов разработки информационных систем

5. Верификация, валидация и аудит информационных систем

6. Extreme Programming: общее описание, основные принципы организации процесса

7. Разработка информационных систем. Scrum: общее описание, роли, практики

8. Обзор технологии Microsoft Visual Studio Team System (VSTS). Состав продукта: обзор, клиентская часть VSTS, серверная часть VSTS. Правила инсталляции. Пакет Team Explorer.

9. Управление элементами работ. Определение, свойства, жизненный цикл. Реквизиты.

10. Управление элементами работ. Средства использования (на примере элемента работы task). Доступ к элементам работы. Элементы работы при планировании. Элементы работы в дальнейшей разработке. Элементы работы в отчетах.

4.2 Подготовка к лабораторным работам

Лабораторная работа 1. . Концептуальное проектирование информационной системы

1. Что такое жизненный цикл программного продукта?
2. Дайте определение модели жизненного цикла ПО.
3. Приведите этапы разработки программного средства.
4. Какие этапы включает в себя модель ЖЦ ПС согласно ГОСТ 19.102-77?
5. Что включает в себя этап предпроектного исследования?
6. Перечислите функциональные требования к программному продукту.
7. Перечислите эксплуатационные требования к программному продукту.
8. Перечислите правила разработки технического задания.
9. Назовите основные разделы технического задания.
10. В каких отношениях находятся заказчик и разработчик при выработке требований к программному средству?

Лабораторная работа 2. Проектирование архитектуры информационной системы

1. Предложите, кто бы мог участвовать в формировании требований для университетской системы регистрации студентов. Объясните, почему почти неизбежно, что требования, сформулированные разными лицами, будут противоречивы.
2. Разрабатывается система ПО для автоматизации библиотечного каталога. Эта система будет содержать информацию относительно всех книг в библиотеке и будет полезна библиотечному персоналу, абонентам и читателям. Система должна иметь средства просмотра каталога, средства создания запросов и средства, позволяющие пользователям резервировать книги, находящиеся в данный момент на руках. Определите основные опорные точки зрения, которые необходимо учесть в спецификации системы, и покажите их взаимоотношения, используя диаграмму иерархии точек зрения.
3. Для трех точек зрения, определенных в системе библиотечного каталога, укажите сервисы и соответствующие данные, которые обеспечиваются этими точками зрения, и события, которые управляют этими сервисами.
4. Кто должен проводить обзор требований? Нарисуйте модель процесса обзора требований.
5. Ваша компания использует стандартный метод анализа требований. В процессе работы вы обнаружили, что этот метод не учитывает социальные факторы, важные для системы, которую вы анализируете. Ваш руководитель дал вам ясно понять, какому методу анализа нужно следовать. Обсудите, что вы должны делать в такой ситуации.

Лабораторная работа 3. Проектирование базы данных информационной системы

1. Перечислите основные объекты DFD, их описание и назначение.
2. Назовите базовые принципы инфологического моделирования.
3. В каких случаях целесообразно применять построение модели “как есть”, а в каких “как будет”?
4. Перечислите основные объекты нотации ERD, их описание и назначение.
5. В чём смысл использования связей в реляционных базах данных?
6. В чём отличия IDEF0 и DFD?
7. Когда целесообразней использовать IDEF0, а когда DFD?

Лабораторная работа 4. Проектирование программного интерфейса информационной системы

1. Дайте определение понятию «вариант использования».
2. Какие типы связи могут присутствовать на диаграмме вариантов использования?
3. Дайте определение понятию «действующее лицо».
4. Какие типы сообщений могут присутствовать на диаграммах взаимодействия?
5. Дайте определение понятию класс, объект класса.
6. Кем и для чего может быть использована диаграмма размещения?
7. Опишите процесс проектирования электронных форм ввода-вывода
8. Для чего применяется спецификация программных модулей?

4.3 Темы курсового проектирования

1. Разработка проекта прикладной информационной системы класса CRM
2. Разработка проекта прикладной информационной системы оперативного учета поступления товара на склад
3. Разработка проекта прикладной информационной системы «Недвижимость»
4. Разработка проекта прикладной информационной системы «Поставщики предприятия»
5. Разработка проекта прикладной информационной системы учета программных средств предприятия
6. Разработка проекта прикладной информационной системы тестирования знаний
7. Разработка проекта прикладной информационной системы «Абитуриент»
8. Разработка проекта прикладной информационной системы статистической обработки данных
9. Разработка проекта прикладной информационной системы «Библиотека»
10. Разработка проекта прикладной программы учета данных для информационной системы «Реклама»
11. Разработка проекта прикладной информационной системы учета успеваемости студентов
12. Разработка проекта прикладной информационной системы «Выпускник вуза»
13. Разработка проекта прикладной информационной системы «Отдел кадров»
14. Разработка проекта прикладной информационной системы учета продаж торгового предприятия
15. Разработка проекта прикладной программы учета ресурсов для информационной системы класса MRP
16. Разработка проекта прикладной обработки данных по производственной практике
17. Разработка проекта прикладной программы учета данных для информационной системы «Турагентство»
18. Разработка проекта прикладной информационной системы «Санаторно-курортная деятельность»
19. Разработка проекта прикладной программы вывода афиши для информационной системы «Афиша»
20. Разработка проекта прикладной информационной системы «Салон красоты»
21. Разработка проекта прикладной программы ввода-вывода данных для информационной системы «Аптека»
22. Разработка проекта прикладной информационной системы «Школа»
23. Разработка проекта прикладной информационной системы «Регистратура поликлиники»
24. Разработка проекта прикладной информационной системы «Гостиница»
25. Разработка проекта прикладной информационной системы «Авиабилеты»
26. Разработка проекта прикладной информационной системы «Кинотеатр»
27. Разработка проекта прикладной информационной системы «Книжный магазин»

28. Разработка проекта прикладной информационной системы «Компьютерный магазин»
29. Разработка проекта прикладной информационной системы «Учет курсовых проектов»
30. Разработка проекта прикладной информационной системы «Интернет-магазин»

4.4 Вопросы к экзамену

Базовый уровень

Знать
1. Рабочий проект информационной системы. Дисциплина обязательств.
2. Технический проект информационной системы
3. Управление проектом информационной системы
4. Календарный план разработки информационной системы.
5. Понятие архитектуры информационной системы
6. Открытая архитектура информационных систем
7. Системная инженерия: точка зрения и характеристики точек зрения
8. Множественность точек зрения при разработке ИС
9. Характеристики информационной системы. Интерактивность, usability
10. Применение инструментов моделирования информационных систем.

Уметь, Владеть
11. Метрики качества программного обеспечения
12. Стандартный метод оценки значений показателей качества
13. Управление качеством ПО
14. Оптимизация и реинжиниринг
15. Инновационные проекты.
16. Тестирование информационной системы
17. Стандартизация качества информационных систем
18. Методы обеспечения качества информационных систем
19. Понятие тестирования информационной системы
20. Инструменты тестирования информационных систем

Повышенный уровень

Знать
1. Принципы верификации и тестирования
2. Верификация и валидация программных продуктов
3. Понятие тестирования программных средств
4. Методы верификации объектно-ориентированных программ
5. Качество и надежность программного обеспечения
6. Профили открытых информационных систем
7. Функциональные и технологические стандарты разработки ПО
8. Принципы организации проектирования и программных комплексов
9. Задачи обеспечения качества программных компонентов
10. Методы исследования качества программных компонентов

Уметь, Владеть
11. Конфигурационное управление. Система контроля версий.
12. VSTS: конфигурационное управление. Отслеживание изменений
13. Тестирование. Система отслеживания ошибок
14. VSTS: тестирование. Связь изменений исходных текстов ПО и ошибок
15. Автоматическое тестирование Web-приложений.
16. Открытая архитектура информационных систем
17. Системная инженерия: точка зрения и характеристики точек зрения
18. Множественность точек зрения при разработке ИС
19. Характеристики информационной системы. Интерактивность, usability
20. Применение инструментов моделирования информационных систем.

4.5 Подготовка к выполнению самостоятельной работы

Для выполнения самостоятельной работы следует изучить теоретический материал.

5.ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

5.1 КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Организация процессов разработки проекта ИС отличается значительной сложностью. К причинам, обусловливающим сложность данных процессов, следует отнести прежде всего:

- масштабы разработки ИС;
- взаимосвязь различных по своей природе элементов проекта ИС (информационные, программные и технические средства обработки информации; экономико-математические модели; методы и средства проектирования; специалисты-разработчики; элементы проекта системы и др.);
- различные факторы старения указанных элементов;
- разный временной цикл существования и темпов обновления элементов;
- длительность процесса проектирования системы;
- индивидуальность проекта, обусловленную спецификой объекта проектирования;
- коллективный характер труда многих специалистов различной квалификации.

В области проектирования ИС появился самостоятельный рынок услуг по проектированию, покупке и установке вычислительной техники, разработке локальных сетей, прокладке сетевого оборудования и обучению пользователей, выполняемых компаниями, называемыми «системными интеграторами».

Под термином «системный интегратор» понимают компании, которые призваны выполнять комплексное решение задач заказчика при построении ИС, поскольку заказчик готов переложить детальную проработку и реализацию проекта на плечи системного интегратора, определив лишь исходные данные и задачи, которые должна решать реализуемая ИС. Такая компания выполняет, как правило, следующий набор функций:

- продажа (дистрибуция, поставка для проектов) аппаратного обеспечения;
- продажа (дистрибуция, поставка для проектов) программного обеспечения;
- консалтинг, проектные работы, сервис, техническая поддержка, обучение.

Объектами, с которыми работают специалисты фирм-интеграторов, являются:

- офисные и корпоративные сети;
- многоуровневые системы хранения информации;
- системы управления технологическими процессами;
- корпоративные автоматизированные информационные системы для крупных банков, нефтегазовых компаний, крупных химических комбинатов и т.д.

Реализуя интеграционные проекты, фирмы проводят следующие виды работ:

- проектируют топологию интегрированных вычислительных систем;
- выбирают эффективные технические решения;
- определяют оптимальный состав аппаратных и программных средств;
- осуществляют монтаж, сопровождение и гарантийное обслуживание техники в течение всего срока эксплуатации системы;
- осуществляют поставки готовых компонентов информационной системы, включая вычислительную и сетевую технику, различные программные продукты (сетевые средства, системы офисной автоматизации, банковские системы и прикладное программное обеспечение);
- разрабатывают собственное программное обеспечение на базе современных CASE-технологий.

Основные понятия и классификация процессов обработки данных

Под процессом обработки информации понимается определенный комплекс операций, выполняемых в строго регламентированной последовательности с использованием определенных методов обработки и инструментальных средств,

охватывающих все этапы обработки данных, начиная с регистрации первичных данных и заканчивая передачей результатной информации пользователю для выполнения функций управления.

Процессы обработки данных можно классифицировать по различным признакам (рис. 1.1), в частности по типу автоматизируемых процессов управления в ЭИС можно выделить:

- технологические процессы, выполняемые в системах обработки данных (СОД);
- технологические процессы аналитической обработки данных в системах подготовки принятия решений (СППР) и экспертных системах (ЭС);
- технологические процессы для разработки новых видов продукции и получения чертежной и технологической документации в системах автоматизированного проектирования (САПР);
- технологические процессы, выполняемые в системах электронного документооборота (СЭД).

Технологический процесс состоит из совокупности технологических операций.

Под технологической операцией будем понимать совокупность функционально связанных действий по преобразованию данных, выполняемых непрерывно на одном рабочем месте.

Основные технологические операции по выполняемой функции в технологическом процессе можно разделить: на рабочие операции и контрольные. В свою очередь, среди рабочих технологических операций по характеру обработки выделяют активные (связанные с логическим или арифметическим преобразованием информации) и пассивные (например, операции ввода-вывода).

Контрольные операции могут принадлежать к определенному методу организации контроля, которые, в свою очередь, объединяются в группы по следующим признакам:

- по времени выполнения: предварительный контроль, текущий контроль, заключительный контроль;
- по степени охвата контролем рабочих операций: пооперационный контроль и контурный контроль, охватывающий несколько рабочих операций;
- по принципам организации выделяют контроль, организованный по принципу дублирования работ (например, метод двойного файла, верификации и др.), принципу информационной избыточности (метод контрольных сумм, модульный метод и др.), принципу логической или арифметической увязки показателей (например, балансовый метод).

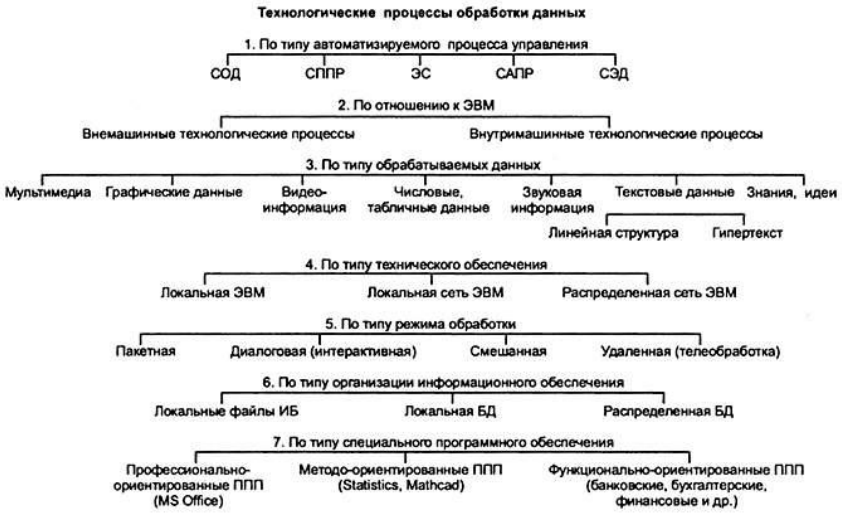


Рисунок 1.1 - Схема классификации технологических процессов обработки данных в ЭИС

Показатели оценки эффективности и выбор варианта организации технологических процессов обработки данных

В процессе проектирования системы обработки данных проектировщик может ориентироваться на несколько вариантов аппаратной платформы и разработать несколько вариантов технологических процессов, среди которых ему необходимо выбрать наилучший. К основным требованиям, предъявляемым к выбираемому технологическому процессу, относятся:

- обеспечение пользователя своевременной информацией;
- обеспечение высокой степени достоверности полученной информации;
- обеспечение минимальности трудовых и стоимостных затрат, связанных с обработкой данных.

При выборе варианта технологического процесса обработки экономической информации используют две группы показателей оценки эффективности: показатели достоверности получения и обработки информации и показатели трудовых и стоимостных затрат на проектирование системы и обработку информации.

Для обеспечения выполнения этих требований необходимо в первую очередь выбрать высокопроизводительную и надежную техническую базу, разработать состав основных операций и методы их реализации. Однако для достижения высокой достоверности обработки и получения результатной информации проектировщик должен помимо этого организовать систему контроля за достоверностью обработки информации. Для разработки такой системы проектировщик обязан проанализировать частоту возникновения ошибок по типам решаемых задач, по классам операций технологического процесса, по видам ошибок и по причинам их возникновения. С этой целью необходимо собрать статистику ошибок и получить распределение частоты их возникновения по следующим направлениям:

- по видам решаемых задач: например, аналитические, плановые, статистические, учетные;
- по классам операций технологического процесса;
- по видам ошибок, связанных с состоянием первичных документов, с переносом данных на машинные носители, с обработкой в ЭВМ, с контролем и выпуском результатных документов;
- по причинам возникновения ошибок: небрежность пользователей и плохое освоение операций по вводу информации в ЭВМ, вина исполнителя документов, ошибки в проекте (вина проектировщиков) и др.

Затем следует выбрать определенный метод контроля за каждой операцией или группой операций и выполнить оценку степени достоверности получаемой после обработки результатной информации.

Показатель достоверности обработки информации (D) может быть рассчитан по следующей формуле:

(1) $D = 1 – P$,

где D- величина достоверности процесса обработки;

P - вероятность появления ошибки, которую можно рассчитать по формуле

(2) $P = N / Q$,

где N - количество ошибочных действий, допущенных на множестве Q;

Q- общее количество действий.

Поскольку проектировщики, как правило, владеют ограниченной выборкой по величинам Q и N, то для оценки достоверности технологических процессов они используют показатель частоты появления ошибок (f), который рассчитывается по формуле (3):

(3) $f = \Delta N/\Delta Q$,

где f- частота возникновения ошибок;

ΔN - число ошибок, допущенных на множестве ΔQ

ΔQ - величина доступной выборки общего количества действий.

Для практической оценки степени достоверности вариантов технологических процессов разработано несколько методик, например, применяется методика с помощью оценки величины, обратной величине достоверности, - степени недостоверности технологического процесса, заданного для множества n-рабочих и m-контрольных операций некоторого технологического процесса и представленного в виде схемы (рис. 1.2).

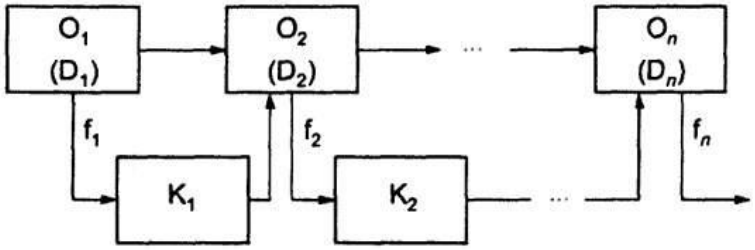


Рисунок 1.2 - Схема технологического процесса обработки данных

При выборе наилучшего технологического процесса обработки экономической информации, помимо использования показателей достоверности, применяют оценку, сравнение и выбор по соотношению уровня производительности того или иного варианта процесса и значению величин показателей трудовых и стоимостных затрат на проектирование и эксплуатацию этих процессов.

В этом комплексе рассчитывают абсолютные и относительные показатели оценки экономической эффективности технологических процессов.

К группе абсолютных показателей относят:

- показатели, оценивающие величину трудоемкости обработки информации за год по базовому (т.е. тому варианту, который берется за основу для сравнения) и предлагаемым вариантам (To) и (Tj);

- показатели, оценивающие величину эксплуатационных стоимостных затрат за год по базовому и предлагаемому вариантам (Co) и (Cj);

- показатель оценки снижения трудовых затрат за год (ΔT), который рассчитывается по формуле

(9) ΔT=To-Tj;

- показатель снижения стоимостных затрат за год (AC), который можно рассчитать по формуле

(10) ΔC=Co-Cj.

Группа относительных показателей оценки эффективности технологических процессов включает:

- коэффициент снижения трудовых затрат за год (Кт), показывающий, на какую долю или какой процент снижаются затраты предлагаемого варианта по сравнению с базовым, который рассчитывается по формуле

(11) Km = ΔT/To;

- индекс снижения трудовых затрат (Iт), показывающий, во сколько раз снижаются трудовые затраты предлагаемого j-го варианта по сравнению с базовым, и рассчитываемый по формуле

(12) Iт =To/Tj

- коэффициент снижения стоимостных затрат за год (Кс), который рассчитывается по формуле

(13) Кс = ΔC/Cj

- индекс снижения стоимостных затрат (Ic), рассчитываемый по формуле

(14) Ic = C0/Cj (14)

Расчетный коэффициент эффективности Е является обратной величиной сроку окупаемости и рассчитывается по формуле

(30) Ер=1/Ток.

По совокупности вышеприведенных показателей проектировщики выбирают наиболее эффективный вариант технологического процесса обработки информации. Обобщенная технологическая сеть выбора варианта организации технологического процесса обработки данных в ИС представлена на рис. 1.3.

Вначале осуществляются работы «Определение состава основных операций» (П1) и «Уточнение состава технических средств выполнения операций» (П2). Входными документами для выполнения этой работы служат материалы обследования, «Постановка задачи» (Д1.1), «Техническое задание» (Д1.2) и множество предварительно выбранных технических средств для операций технологического процесса (U2.1). В результате выполнения этих работ проектировщики получают перечень основных операций (Д1.3), описание технико-эксплуатационных характеристик выбранных технических средств (Д2.1) и методов работы с ними (Д2.2), которые поступают в качестве исходных данных на вход следующей операции.

На следующей операции выполняется «Выбор метода контроля и технических средств, осуществляющих контроль» (П3). На вход операции поступает универсум методов контроля (U3.1). В результате выполнения процедуры получают описание технических средств и методов выполнения контроля (Д3.1).

Далее осуществляется «Разработка вариантов схем технологического процесса обработки данных» (П4). Входными документами для данной операции являются перечни основных операций, технических характеристик средств и методик выполнения контроля (Д1.3, Д2.2, Д2.1, Д3.1). Целью выполнения данной работы является получение блок-схем нескольких вариантов технологических процессов (Д4.1).

Содержанием пятой операции является «Оценка технологических процессов по достоверности, трудовым и стоимостным показателям» (П5). Данная оценка производится на основе технического задания и методик расчета показателей (U5.1). Результатом выполнения работы является получение таблиц значений показателей (Д5.1).

Заключительной операцией служит «Выбор варианта технологического процесса и разработка технологической документации» (П6). Выполнение данной работы основывается на содержании технического задания, требовании гостов и остов на техно-рабочий проект (Д6.1). В результате получают совокупность технологических и инструкционных карт (Д6.2).

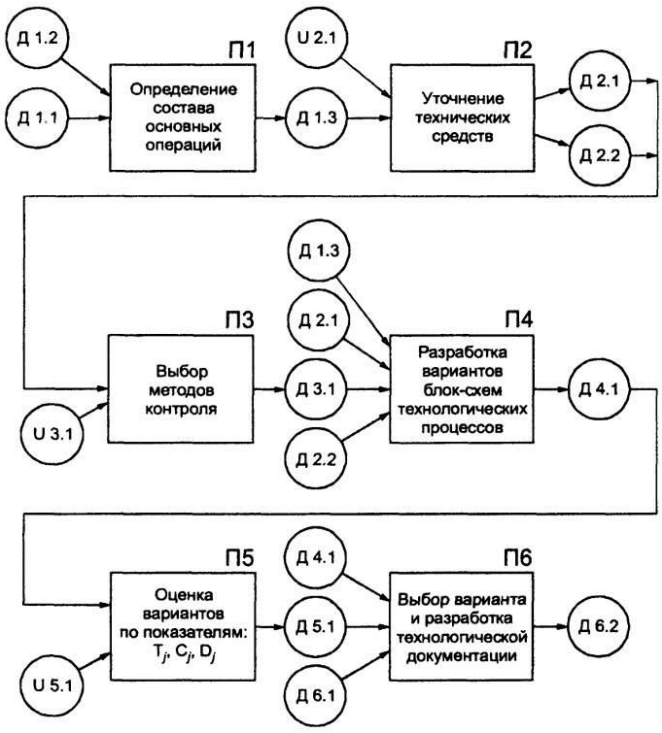


Рисунок 1.3 - Технологическая сеть выбора варианта технологического процесса обработки данных в ЭИС:

Д 1.1 - постановка задачи; Д 1.2 - состав основных операций; U 2.1 - универсум комплекса предварительно выбранных вариантов ТС; Д 2.1 - описание выбранного КТС; Д 2.2 - методы работы; U 3.1 - универсум методов контроля; Д 3.1 - описание методов контроля; Д 3.2 - уточненный вариант КТС; Д 4.1 - варианты схем технических процессов; U 5.1 - универсум методик оценки Tj, Cj, Dj;Д 5.1 - таблицы значений показателей; Д 6.1 - требования ТЗ; Д 6.2 - технологические и инструкционные карты

Проблемы, которые приходится решать специалистам в процессе создания программного обеспечения, очень сложны. Природа этих проблем не всегда ясна, особенно если разрабатываемая программная система инновационная. В частности, трудно чётко описать те действия, которые должна выполнять система. Описание функциональных возможностей и ограничений, накладываемых на систему, называется требованиями к этой системе, а сам процесс формирования, анализа, документирования и проверки этих функциональных возможностей и ограничений – разработкой требований.

Требования подразделяются на пользовательские и системные. Пользовательские требования – это описание на естественном языке (плюс поясняющие диаграммы) функций, выполняемых системой, и ограничений, накладываемых на неё. Системные требования – это описание особенностей системы (архитектура системы, требования к параметрам оборудования и т.д.), необходимых для эффективной реализации требований пользователя.

Разработка требований

Разработка требований — это процесс, включающий мероприятия, необходимые для создания и утверждения документа, содержащего спецификацию системных требований. Различают четыре основных этапа процесса разработки требований:

- анализ технической осуществимости создания системы,
- формирование и анализ требований,
- специфицирование требований и создание соответствующей документации,
- аттестация этих требований.

На рисунке 1.4 показаны взаимосвязи между этими этапами и результаты, сопровождающие каждый этап процесса разработки системных требований.

Но поскольку в процессе разработки системы в силу разнообразных причин требования могут меняться, управление требованиями, т.е. процесс управления изменениями системных требований, является необходимой составной частью деятельности по их разработке.

Формирование и анализ требований

Следующим этапом процесса разработки требований является формирование (определение) и анализ требований.

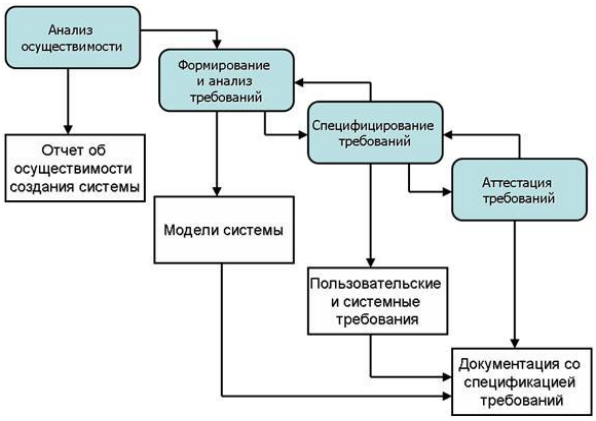


Рисунок 1.4 - Процесс разработки требований

Обобщенная модель процесса формирования и анализа требований показана на рисунке 1.5. Каждая организация использует собственный вариант этой модели, зависящий от “местных факторов”: опыта работы коллектива разработчиков, типа разрабатываемой системы, используемых стандартов и т.д.

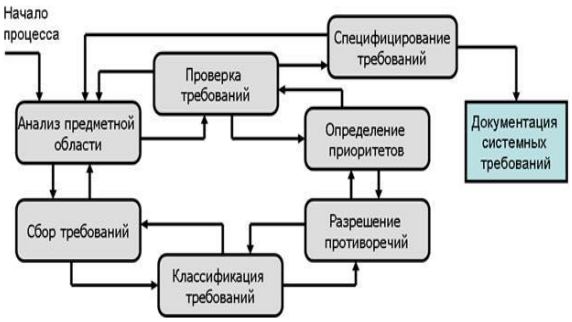


Рисунок 1.5 - Процесс формирования и анализа требований

Процесс формирования и анализа требований проходит через ряд этапов.

1. Анализ предметной области. Аналитики должны изучить предметную область, где будет эксплуатироваться система.

2. Сбор требований. Это процесс взаимодействия с лицами, формирующими требования. Во время этого процесса продолжается анализ предметной области.

3. Классификация требований. На этом этапе бесформенный набор требований преобразуется в логически связанные группы требований.

4. Разрешение противоречий. Без сомнения, требования многочисленных лиц, занятых в процессе формирования требований, будут противоречивыми. На этом этапе определяются и разрешаются противоречия различного рода.

5. Назначение приоритетов. В любом наборе требований одни из них будут более важны, чем другие. На этом этапе совместно с лицами, формирующими требования, определяются наиболее важные требования.

6. Проверка требований. На этом этапе определяется их полнота, последовательность и непротиворечивость.

Процесс формирования и анализа требований циклический, с обратной связью от одного этапа к другому. Цикл начинается с анализа предметной области и заканчивается проверкой требований. Понимание требований предметной области увеличивается в каждом цикле процесса формирования требований.

Рассмотрим три основных подхода к формированию требований: метод, основанный на множестве опорных точек зрения, сценарии и этнографический метод.

Опорные точки зрения

Подход с использованием различных опорных точек зрения к разработке требований признает различные (опорные) точки зрения на проблему и использует их в качестве основы построения и организации как процесса формирования требований, так и непосредственно самих требований.

Различные методы предлагают разные трактовки выражения "точка зрения". Точки зрения можно трактовать следующим образом.

1. Как источник информации о системных данных. В этом случае на основе опорных точек зрения строится модель создания и использования данных в системе. В процессе формирования требований отбираются все такие точки зрения (и на их основе определяются данные), которые будут созданы или использованы при работе системы, а также способы обработки этих данных.

2. Как структура представлений. В этом случае точки зрения рассматриваются как особая часть модели системы. Например, на основе различных точек зрения могут разрабатываться модели "сущность-связь", модели конечного автомата и т.д.

3. Как получатели системных сервисов. В этом случае точки зрения являются внешними (относительно системы) получателями системных сервисов. Точки зрения помогают определить данные, необходимые для выполнения системных сервисов или их управления.

Наиболее эффективным подходом к анализу таких систем является использование внешних опорных точек зрения. На основе этого подхода разработан метод VORD (Viewpoint-Oriented Requirements Definition — определение требований на основе точек зрения) для формирования и анализа требований. Основные этапы метода VORD показаны на рисунке 1.6.

1. Идентификация точек зрения, получающих системные сервисы, и идентификация сервисов, соответствующих каждой точке зрения.

2. Структурирование точек зрения — создание иерархии сгруппированных точек зрения. Общесистемные сервисы предоставляются более высоким уровням иерархии и наследуются точками зрения низшего уровня.

3. Документирование опорных точек зрения, которое заключается в точном описании идентифицированных точек зрения и сервисов.

4. Отображение системы точек зрения, которая показывает системные объекты, определенные на основе информации, заключенной в опорных точках зрения.

Рисунок 1.6 - Метод VORD

Пример. Рассмотрим использование метода VORD на первых трех шагах анализа требований для системы системы поддержки заказа и учета товаров в бакалейной лавке. В бакалейной лавке для каждого товара фиксируется место хранения (определенная полка), количество товара и его поставщик. Система поддержки заказа и учета товаров должна

обеспечивать добавление информации о новом товаре, изменение или удаление информации об имеющемся товаре, хранение (добавление, изменение и удаление) информации о поставщиках, включающей в себя название фирмы, ее адрес и телефон. При помощи системы составляются заказы поставщикам. Каждый заказ может содержать несколько позиций, в каждой позиции указываются наименование товара и его количество в заказе. Система по требованию пользователя формирует и выдает на печать следующую справочную информацию:

- список всех товаров;
- список товаров, имеющихся в наличии;
- список товаров, количество которых необходимо пополнить;
- список товаров, поставляемых данным поставщиком.

Первым шагом в формировании требований является идентификация опорных точек зрения. Во всех методах формирования требований, основанных на использовании точек зрения, начальная идентификация является наиболее трудной задачей. Один из подходов к идентификации точек зрения — метод "мозговой атаки", когда определяются потенциальные системные сервисы и организации, взаимодействующие с системой.

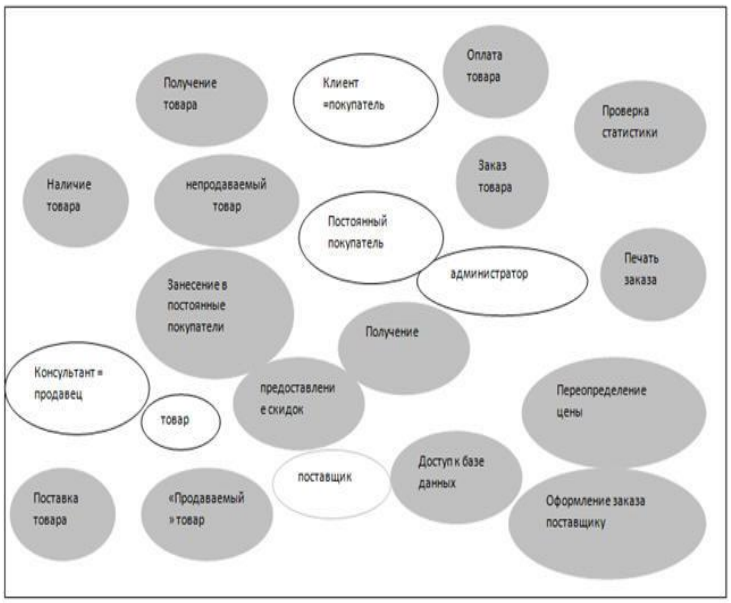


Рисунок 1.7 - Диаграмма идентификации точек зрения

Организуется встреча лиц, участвующих в формировании требований, которые предлагают свои точки зрения. Эти точки зрения представляются в виде диаграммы, состоящей из ряда круговых областей, отображающих возможные точки зрения (рис. 1.7). Во время "мозговой атаки" необходимо идентифицировать потенциальные опорные точки зрения, системные сервисы, входные данные, нефункциональные требования, управляющие события и исключительные ситуации.

Следующей стадией процесса формирования требований будет идентификация опорных точек зрения (на рисунке 1.7 показаны в виде темных круговых областей) и сервисов (показаны в виде затененных областей). Сервисы должны соответствовать опорным точкам зрения. Но могут быть сервисы, которые не поставлены им в соответствие. Это означает, что на начальном этапе "мозговой атаки" некоторые опорные точки зрения не были идентифицированы.

В таблице 1.1 показано распределение сервисов для некоторых идентифицированных на рисунке 1.7 точек зрения. Один и тот же сервис может быть соотнесен с несколькими точками зрения.

Таблица 1.1 - Сервисы, соотнесенные с точками зрения

клиент	**покупатель**	**постоянный покупатель**	**товар**	**поставщик**	**продавец**	**администратор**
Проверка наличия товара	Занесение в список клиентов	Получение скидки	Прием товара	Занесение в базу данных	Продажа товара	Доступ к базе данных
Покупка товара		Получение информации	Занесение в базу данных		Печать чека	Проверка статистики
Получение чека			Назначение цены		Доступ к каталогу	Переопределение цены
Заказ товара			Переопределение цены		Проверка наличия товара	Оформление заказа
Занесение покупателя и суммы покупки в базу данных			«Покупаемый» или «непокупаемый» товар		Оформление заказа покупателю	Печать заказа

Информация, извлеченная из точек зрения, используется для заполнения форм шаблонов точек зрения и организации точек зрения в иерархию наследования. Это позволяет увидеть общие точки зрения и повторно использовать информацию в иерархии наследования. Сервисы, данные и управляющая информация наследуются подмножеством точек зрения. На рисунке 1.8 показана часть иерархии точек зрения для системы поддержки заказа и учета товаров.

Рисунок 1.8 - Иерархия точек зрения

Аттестация требований

Аттестация должна продемонстрировать, что требования действительно определяют ту систему, которую хочет иметь заказчик. Проверка требований важна, так как ошибки в спецификации требований могут привести к переделке системы и большим затратам, если будут обнаружены во время процесса разработки системы или после введения ее в эксплуатацию. Стоимость внесения в систему изменений, необходимых для устранения ошибок в требованиях, намного выше, чем исправление ошибок

проектирования или кодирования. Причина в том, что изменение требований обычно влечет за собой значительные изменения в системе, после внесения которых она должна пройти повторное тестирование.

Во время процесса аттестации должны быть выполнены различные типы проверок требований.

1. Проверка правильности требований. Пользователь может считать, что система необходима для выполнения некоторых определенных функций. Однако дальнейшие размышления и анализ могут привести к необходимости введения дополнительных или новых функций. Системы предназначены для разных пользователей с различными потребностями, и поэтому набор требований будет представлять собой некоторый компромисс между требованиями пользователей системы.

2. Проверка на непротиворечивость. Спецификация требований не должна содержать противоречий. Это означает, что в требованиях не должно быть противоречащих друг другу ограничений или различных описаний одной и той же системной функции.

3. Проверка на полноту. Спецификация требований должна содержать требования, которые определяют все системные функции и ограничения, налагаемые на систему.

4. Проверка на выполнимость. На основе знания существующих технологий требования должны быть проверены на возможность их реального выполнения. Здесь также проверяются возможности финансирования и график разработки системы.

Существует ряд методов аттестации требований, которые можно использовать совместно или каждый в отдельности.

1. Обзор требований. Требования системно анализируются рецензентами.

2. Прототипирование. На этом этапе прототип системы демонстрируется конечным пользователям и заказчику. Они могут экспериментировать с этим прототипом, чтобы убедиться, что он отвечает их потребностям.

3. Генерация тестовых сценариев. В идеале требования должны быть такими, чтобы их реализацию можно было протестировать. Если тесты для требований разрабатываются как часть процесса аттестации, то часто это позволяет обнаружить проблемы в спецификации. Если такие тесты сложно или невозможно разработать, то обычно это означает, что требования трудно выполнить и поэтому необходимо их пересмотреть.

4. Автоматизированный анализ непротиворечивости. Если требования представлены в виде структурных или формальных системных моделей, можно использовать инструментальные CASE-средства для проверки непротиворечивости моделей. Для автоматизированной проверки непротиворечивости необходимо построить базу данных требований и затем проверить все требования в этой базе данных. Анализатор требований готовит отчет обо всех обнаруженных противоречиях.

Пользовательские и системные требования

На основании полученных моделей строятся пользовательские требования, т.е. как было сказано в начале описание на естественном языке функции, выполняемых системой, и ограничений, накладываемых на неё.

Пользовательские требования должны описывать внешнее поведение системы, основные функции и сервисы предоставляемые системой, её нефункциональные свойства. Необходимо выделить опорные точки зрения и сгруппировать требования в соответствии с ними. Пользовательские требования можно оформить как простым перечислением, так и используя нотацию вариантов использования.

Далее составляются системные требования. Они включают в себя:

1. Требования к архитектуре системы. Например, число и размещение хранилищ и серверов приложений.

2. Требования к параметрам оборудования. Например, частота процессоров серверов и клиентов, объём хранилищ, размер оперативной и видео памяти, пропускная способность канала и т.д.

3. Требования к параметрам системы. Например, время отклика на действие пользователя, максимальный размер передаваемого файла, максимальная скорость передачи данных, максимальное число одновременно работающих пользователей и т.д.

4. Требования к программному интерфейсу.

5. Требования к структуре системы. Например, Масштабируемость, распределённость, модульность, открытость.

- масштабируемость – возможность распространения системы на большое количество машин, не приводящая к потере работоспособности и эффективности, при этом способность системы наращивать свою мощность должна определяться только мощностью соответствующего аппаратного обеспечения.
- распределенность - система должна поддерживать распределённое хранение данных.
- модульность - система должна состоять из отдельных модулей, интегрированных между собой.
- открытость - наличие открытых интерфейсов для возможной доработки и интеграции с другими системами.

6. Требования по взаимодействию и интеграции с другими системами. Например, использование общей базы данных, возможность получения данных из баз данных определённых систем и т.д.

Техническое задание

Техническое задание представляет собой документ, в котором сформулированы основные цели разработки, требования к программному продукту, определены сроки и этапы разработки и регламентирован процесс приемо-сдаточных испытаний. В разработке технического задания участвуют как представители заказчика, так и представители исполнителя. В основе этого документа лежат исходные требования заказчика, анализ передовых достижений техники, результаты выполнения научно-исследовательских работ, предпроектных исследований, научного прогнозирования и т. п.

Порядок разработки технического задания

Разработка технического задания выполняется в следующей последовательности. Прежде всего устанавливают набор выполняемых функций, а также перечень и характеристики исходных данных. Затем определяют перечень результатов, их характеристики и способы представления.

Далее уточняют среду функционирования программного обеспечения: конкретную комплектацию и параметры технических средств, версию используемой операционной системы и, возможно, версии и параметры другого установленного программного обеспечения, с которым предстоит взаимодействовать будущему программному продукту.

В случаях, когда разрабатываемое программное обеспечение собирает и хранит некоторую информацию или включается в управление каким-либо техническим процессом, необходимо также четко регламентировать действия программы в случае сбоев оборудования и энергоснабжения.

1. Общие положения

1.1. Техническое задание оформляют в соответствии со стандартом ГОСТ 19.201-78, ГОСТ 34.602-89.

1.2. Для внесения изменений и дополнений в техническое задние на последующих стадиях разработки программы или программного изделия выпускают дополнение к нему. Согласование и утверждение дополнения к техническому заданию проводят в том же порядке, который установлен для технического задания.

1.3. Техническое задание должно содержать следующие разделы:

1) общие сведения;

2) назначение и цели создания (развития) системы;

3) характеристика объектов автоматизации;

4) требования к системе;

5) состав и содержание работ по созданию системы;

6) порядок контроля и приемки системы;

7) требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие;

8) требования к документированию;

9) источники разработки.

В зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них. При необходимости допускается в техническое задание включать приложения.

2. Содержание разделов

2.1. В разделе «Общие сведения» указывают:

1) полное наименование системы и ее условное обозначение;

2) шифр темы или шифр (номер) договора;

3) наименование предприятий (объединений) разработчика и заказчика (пользователя) системы и их реквизиты;

4) перечень документов, на основании которых создается система, кем и когда утверждены эти документы;

5) плановые сроки начала и окончания работы по созданию системы;

6) сведения об источниках и порядке финансирования работ;

7) порядок оформления и предъявления заказчику результатов работ по созданию системы (ее частей), по изготовлению и наладке отдельных средств (технических, программных, информационных) и программно-технических (программно-методических) комплексов системы.

2.2. Раздел «Назначение и цели создания (развития) системы» состоит из подразделов:

1) назначение системы;

2) цели создания системы.

2.2.1. В подразделе «Назначение системы» указывают вид автоматизируемой деятельности (управление, проектирование и т. п.) и перечень объектов автоматизации (объектов), на которых предполагается ее использовать.

2.2.2. В подразделе «Цели создания системы» приводят наименования и требуемые значения технических, технологических, производственно-экономических или других показателей объекта автоматизации, которые должны быть достигнуты в результате создания ИС, и указывают критерии оценки достижения целей создания системы.

2.3. В разделе «Характеристики объекта автоматизации» приводят:

1) краткие сведения об объекте автоматизации или ссылки на документы, содержащие такую информацию;

2) сведения об условиях эксплуатации объекта автоматизации и характеристиках окружающей среды.

Примечание: Для САПР в разделе дополнительно приводят основные параметры и характеристики объектов проектирования.

2.4. Раздел «Требования к системе» состоит из следующих подразделов:

1) требования к системе в целом;

2) требования к функциям (задачам), выполняемым системой;

3) требования к видам обеспечения.

Состав требований к системе, включаемых в данный раздел ТЗ на ИС, устанавливают в зависимости от вида, назначения, специфических особенностей и условий функционирования конкретной системы. В каждом подразделе приводят ссылки на действующие НТД, определяющие требования к системам соответствующего вида.

2.3.1. В подразделе «Требования к системе в целом» указывают:

- требования к структуре и функционированию системы;

- требования к численности и квалификации персонала системы и режиму его работы;

- показатели назначения;

- требования к надежности;

- требования безопасности;

- требования к эргономике и технической эстетике;

- требования к транспортабельности для подвижных АС;

- требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы;

- требования к защите информации от несанкционированного доступа;

- требования по сохранности информации при авариях;

- требования к защите от влияния внешних воздействий;

- требования к патентной чистоте;

- требования по стандартизации и унификации;

- дополнительные требования.

2.3.1.1. В требованиях к структуре и функционированию системы приводят:

1) перечень подсистем, их назначение и основные характеристики, требования к числу уровней иерархии и степени централизации системы;

2) требования к способам и средствам связи для информационного обмена между компонентами системы;

3) требования к характеристикам взаимосвязей создаваемой системы со смежными системами, требования к ее совместимости, в том числе указания о способах обмена информацией (автоматически, пересылкой документов, по телефону и т. п.);

4) требования к режимам функционирования системы;

5) требования по диагностированию системы;

6) перспективы развития, модернизации системы.

2.3.1.2. В требованиях к показателям назначения ИС приводят значения параметров, характеризующие степень соответствия системы ее назначению.

Для ИС управления указывают:

- допустимые пределы модернизации и развития системы;

- временные характеристики, при которых сохраняется назначение системы.

2.3.2. В подразделе «Требования к видам обеспечения» в зависимости от вида системы приводят требования к математическому, информационному, лингвистическому, программному, техническому, метрологическому, организационному, методическому и другие видам обеспечения системы.

2.3.2.1. Для математического обеспечения системы приводят требования к составу, области применения (ограничения) и способам, использования в системе математических методов и моделей, типовых алгоритмов и алгоритмов, подлежащих разработке.

2.3.2.2. Для информационного обеспечения системы приводят требования:

1) к составу, структуре и способам организации данных в системе;

2) к информационному обмену между компонентами системы;

3) к информационной совместимости со смежными системами;

4) по использованию общесоюзных и зарегистрированных республиканских, отраслевых классификаторов, унифицированных документов и классификаторов, действующих на данном предприятии;

5) по применению систем управления базами данных;

6) к структуре процесса сбора, обработки, передачи данных в системе и представлению данных;

7) к защите данных от разрушений при авариях и сбоях в электропитании системы;

8) к контролю, хранению, обновлению и восстановлению данных;

9) к процедуре придания юридической силы документам, продуцируемым техническими средствами ИС (в соответствии с ГОСТ 6.10.4).

2.3.2.3. Для лингвистического обеспечения системы приводят требования к применению в системе языков программирования высокого уровня, языков взаимодействия пользователей и технических средств системы, а также требования к кодированию и декодированию данных, к языкам ввода-вывода данных, языкам манипулирования данными, средствам описания предметной области (объекта автоматизации), к способам организации диалога.

2.3.2.4. Для программного обеспечения системы приводят перечень покупных программных средств, а также требования:

1) к независимости программных средств от используемых СВТ и операционной среды;

2) к качеству программных средств, а также к способам его обеспечения и контроля;

3) по необходимости согласования вновь разрабатываемых программных средств с фондом алгоритмов и программ.

2.3.2.5. Для технического обеспечения системы приводят требования:

1) к видам технических средств, в том числе к видам комплексов технических средств, программно-технических комплексов и других комплектующих изделий, допустимых к использованию в системе;

2) к функциональным, конструктивным и эксплуатационным характеристикам средств технического обеспечения системы.

2.3.2.6. Для организационного обеспечения приводят требования к структуре и функциям подразделений, участвующих в функционировании системы или обеспечивающих эксплуатацию;•

2.3.2.7. Для методического обеспечения ИС приводят требования к составу нормативно-технической документации системы (перечень применяемых при ее функционировании стандартов, нормативов, методик и т. п.).

2.4. Раздел «Состав и содержание работ по созданию (развитию) системы» должен содержать перечень стадий и этапов работ по созданию системы в соответствии с ГОСТ 24.601, сроки их выполнения, перечень организаций - исполнителей работ, ссылки на документы, подтверждающие согласие этих организаций на участие в создании системы, или запись, определяющую ответственного (заказчик или разработчик) за проведение этих работ.

2.5. В разделе «Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие» необходимо привести перечень основных мероприятий и их исполнителей, которые следует выполнить при подготовке объекта автоматизации к вводу ИС в действие.

2.6. В состав ТЗ на проектирование ИС при наличии утвержденных методик включают приложения, содержащие:

1) расчет ожидаемой эффективности системы;

2) оценку научно-технического уровня системы.

Приложения включают в состав ТЗ на проектирование ИС по согласованию между разработчиком и заказчиком системы.

5.2. ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Предпроектная стадия проектирования

В целях изучения взаимосвязанных приемов и методов канонического проектирования ИС перечисленные 7 стадий можно сгруппировать в часто используемые на практике четыре стадии процесса разработки ИС (рис. 2.1):

Вторая стадия «Техно-рабочее проектирование» выполняется в два этапа: техническое проектирование и рабочее проектирование.

На этапе «Техническое проектирование» выполняются работы по логической разработке и выбору наилучших вариантов проектных решений, в результате чего создается «Технический проект». Этап «Рабочее проектирование» связан с физической реализацией выбранного варианта проекта и получением документации «Рабочего проекта». При наличии опыта проектирования эти этапы иногда объединяются в один, в результате выполнения которого получают «Техно-рабочий проект» (ТРП) - Д2.1.

Третья стадия «Внедрение проекта» включает в себя три этапа: подготовка объекта к внедрению проекта; опытное внедрение проекта и сдача его в промышленную эксплуатацию.

На этапе «Подготовка объекта к внедрению проекта» осуществляется комплекс работ по подготовке предприятия к внедрению разработанного проекта ЭИС. На этапе «Опытное внедрение» осуществляют проверку правильности работы некоторых частей проекта и получают исправленную проектную документацию и «Акт о проведении опытного внедрения». На этапе «Сдача проекта в промышленную эксплуатацию» осуществляют комплексную системную проверку всех частей проекта, в результате которой получают доработанный «Техно-рабочий проект» (Д3.1) и «Акт приемки проекта в промышленную эксплуатацию» (Д3.2).

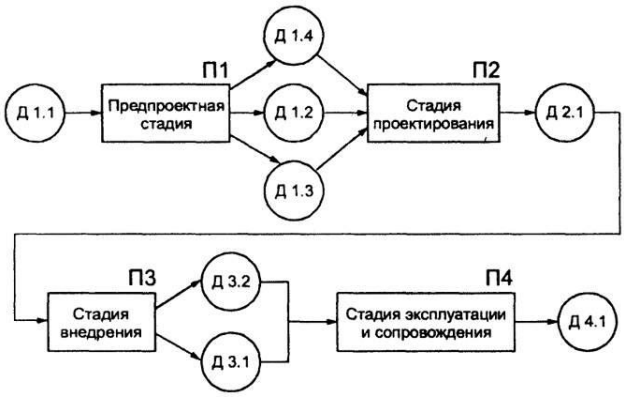


Рисунок 2.2 - ТСП стадий и этапов канонического проектирования ИС:

Д1.1 - предметная область; Д1.2 - материалы обследования; Д1.3 - ТЭО, ТЗ на проектирование; Д1.4 - эскизный проект; Д2.1 - техно-рабочий проект (ТРП); Д3.1 - исправленный ТРП, переданный в эксплуатацию; Д3.2 - акт о приемке проекта в промышленную эксплуатацию; Д4.1 - модернизированный ТРП

Традиционно этапы исследования предметной области - предприятия, обоснование проекта ИС для него и разработки технического задания объединяют термином «Предпроектная стадия» («Предпроектное обследование»), поскольку результаты выполнения работ на данных этапах не являются законченным проектным решением. Основное назначение «Предпроектной стадии» заключается в обосновании экономической целесообразности создания ИС и формулировании требований к ней.

Состав и содержание работ на предпроектной стадии создания ИС

При изучении существующей экономической системы разработчики должны уточнить границы изучения системы, определить круг пользователей будущей ИС различных уровней и выделить классы и типы объектов, подлежащих обследованию и последующей автоматизации.

Важнейшими объектами обследования могут являться:

- структурно-организационные звенья предприятия (например, отделы управления, цехи, участки, рабочие места);
- функциональная структура, состав хозяйственных процессов и процедур;
- стадии (техническая подготовка, снабжение, производство, сбыт) и элементы хозяйственного процесса (средства труда, предметы труда, ресурсы, продукция, финансы).

При каноническом проектировании основной единицей обработки данных является задача. Поэтому функциональная структура проблемной области на стадии предпроектного обследования изучается в разрезе решаемых задач и комплексов задач. При этом задача в содержательном аспекте рассматривается как совокупность операций преобразования некоторого набора исходных данных для получения результатной информации, необходимой для выполнения функции управления или принятия управленческого решения. В большинстве случаев исходные данные и результаты их преобразований представляются в форме экономических документов. Поэтому к числу объектов обследования относятся компоненты потоков информации (документы, показатели, файлы, сообщения). Кроме того, объектами обследования служат:

- технологии, методы и технические средства преобразования информации;
- материальные потоки и процессы их обработки.

Основной целью выполнения первого этапа предпроектного обследования «Сбор материалов» является:

- выявление основных параметров предметной области (например, предприятия или его части);
- установление условий, в которых будет функционировать проект ЭИС;
- выявление стоимостных и временных ограничений на процесс проектирования.

На этом этапе проектировщиками выполняется ряд технологических операций и решаются следующие задачи: предварительное изучение предметной области; выбор технологии проектирования; выбор метода проведения обследования; выбор метода сбора материалов обследования; разработка программы обследования; разработка плана-графика сбора материалов обследования; сбор и формализация материалов обследования. Технологическая сеть проектирования представлена на рис. 2.3.

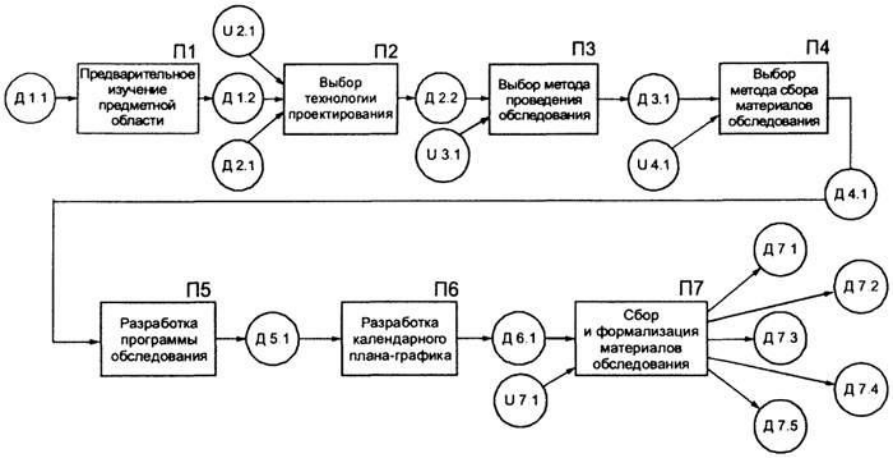


Рисунок 2.3 -ТСП работ, выполняемых на этапе «Сбор материалов обследования»:

Д 1.1 - общие сведения об объекте; Д 1.2 - примеры разработок проектов ЭИС для аналогичных систем; U 2.1 - универсум технологий проектирования; Д 2.1 - ресурсы; Д 2.2 - описание выбранной технологии, методов и средств проектирования; U 3.1 - универсум методов проведения обследования; Д 3.1 - описание выбранного метода; U 4.1 - универсум методов сбора материалов обследования; 4.1 - описание выбранного метода; Д 5.1 - программа обследования; Д 6.1 - план-график выполнения работ на предпроектной стадии; U 7.1 - универсум методов формализации; Д 7.1 - общие параметры (характеристики) экономической системы; Д 7.2 - методики методики управления (алгоритм расчета экономических показателей); Д 7.3 - организационная структура экономической системы;Д 7.4 - параметры информационных потоков; Д 7.5 - параметры материальных потоков

Состав и содержание работ на стадии техно-рабочего проектирования

Работы на стадии «Техно-рабочего проектирования» выполняются на основе утвержденного «Технического задания». Разрабатываются основные положения проектируемой системы, принципы ее функционирования и взаимодействия с другими системами; определяется структура системы; разрабатываются проектные решения по обеспечивающим частям системы.

На стадии «Техно-рабочего проектирования» выполняются два этапа работ: техническое и рабочее проектирование, технологическая сеть которых приведена на рис. 2.4. На первом из них - «Техническое проектирование» осуществляется логическая проработка функциональной и системной архитектуры ЭИС, в процессе которой строится несколько вариантов всех компонентов

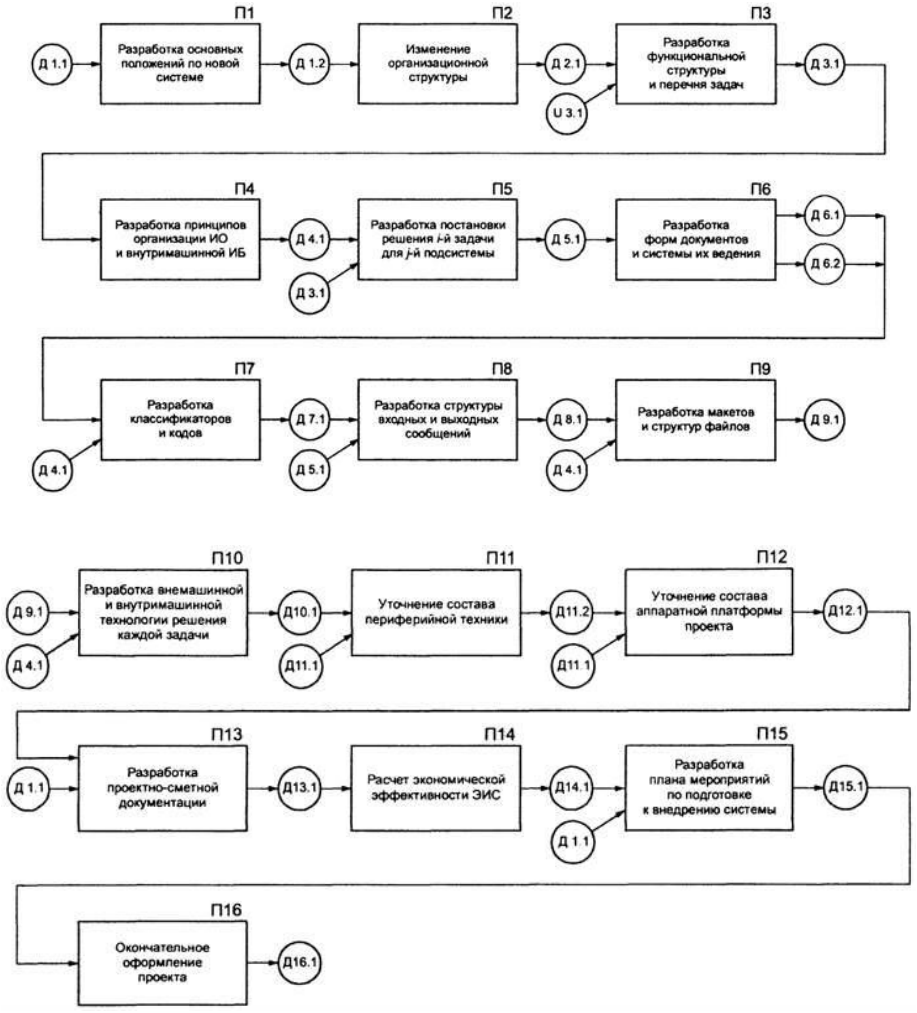


Рисунок 2.4 - ТСП выполнения работ на этапе технического проектирования:

Д 1.1 - ТЗ; Д 1.2 - основные положения по ЭИС; Д 2.1 - описание организационной структуры; Д 3.1 - описание функциональной структуры; Д 4.1 - принципы организации информационного обеспечения; Д 5.1 - постановка задачи; Д 6.1 - формы первичных и результатных документов; Д 6.2 - система ведения документов; Д 7.1 - классификаторы; Д 8.1 - структуры сообщений; Д 9.1 - описание макетов и структур файлов; Д 10.1 - системы технологических процессов обработки данных; Д 11.1 - ТЭО; Д 11.2 - описание состава и характеристика периферийной техники; Д 12.1 - АП; Д 13.1 - проектно-сметная документация; Д 14.1 - показатели экономической эффективности; Д 15.1 - план мероприятий по подготовке объекта к внедрению проекта ЭИС; Д 16.1 - технический проект системы; проводится оценка вариантов по показателям: стоимости, трудоемкости, достоверности получаемых результатов, и составляется «Технический проект» системы.

Все работы первого этапа можно разбить на две группы. К первой группе относится разработка общесистемных проектных решений, в том числе:

- разработка общесистемных положений по ЭИС (П1);
- изменение организационной структуры (П2);
- определение функциональной структуры (ПЗ);
- разработка проектно-сметной документации и расчет экономической эффективности системы (П13), (П14);
- разработка плана мероприятий по внедрению ЭИС (П15).

При разработке основных положений по системе (П1) уточняются цели создания ЭИС и выполняемые ею функции; устанавливается ее взаимосвязь с другими системами и формируется документ Д1.2 «Основные положения». Далее уточняется и изменяется организационная структура (П2) и получается описание организационной структуры (Д2.1).

Наиболее принципиальной в данном комплексе работ является разработка функциональной архитектуры ЭИС (ПЗ) Д3.1 на базе универсума U3.1 принципов выделения функциональных подсистем (модулей, контуров): предметного, функционального, смешанного (предметно-функционального) и проблемного.

Ко второй группе работ, выполняемых на этапе технического проектирования, относятся разработки локальных проектных решений, к числу которых относят следующие операции:

- разработка «Постановки задачи» для задач, входящих в состав каждой функциональной подсистемы (П5), включающей основные компоненты описания задачи и служащей основанием для разработки проектных решений по задаче;
- проектирование форм входных и выходных документов, системы ведения документов и макетов экранных форм документов (П6, П9);
- проектирование классификаторов экономической информации и системы ведения классификаторов (П7);
- разработка структуры входных и выходных сообщений (П8);
- проектирование состава и структур файлов информационной базы (П4);
- проектирование внемашинной и внутримашинной технологии решения каждой задачи (П10);
- уточнение состава технических средств (П11), (П12).

Основным компонентом локальных проектных решений, являющимся базой для разработки информационного, программного и технологического обеспечения для каждой задачи, является «Постановка задачи». Этот документ содержит три составные части (рис. 2.5):

- характеристику задачи;
- описание выходной информации;
- описание входной информации.

В состав раздела «Характеристика задачи» входят следующие компоненты: описание цели; назначение решения конкретной задачи; перечень функций и процессов, реализуемых решаемой задачей; характеристика организационной и технико-экономической сущности задачи; обоснование целесообразности автоматизации решения задачи; указание перечня объектов, для которых решается задача; описание процедур решения задачи; указание периодичности решения задачи и требований к организации сбора первичных данных; описание связей с другими задачами.

Под целью автоматизации решения задачи подразумевается получение определенных значений экономического эффекта в сфере управления какими-либо процессами системы или снижение стоимостных и трудовых затрат на обработку информации, улучшение качества и достоверности получаемой информации, повышение оперативности ее обработки и т.д., т.е. получение косвенного и прямого эффекта от внедрения данной задачи.

Под экономической сущностью решаемой задачи понимаются состав экономических показателей, рассчитываемых при ее решении, документы, в которые заносятся эти показатели, перечень исходных показателей, необходимых для получения результатных и наименования тех первичных документов, в которых они содержатся.

Организационная сущность задачи - это описание порядка решения задачи; организационной формы, применяемой для ее решения; режима решения; состава файлов с постоянной и переменной информацией; способа получения и ввода первичной информации в ЭВМ; формы выдачи результатной информации: на печать, на экран, на магнитный носитель или передача по каналам связи.

Описание алгоритма решения задачи включает формализованное описание входных и результатных показателей и перечень формул расчета результатных показателей в случае решения задачи прямым методом счета или описание математической модели, экономико-математического метода, применяемого для ее реализации, и перечня последовательных шагов выполнения расчетов.

Рисунок 2.5 - Схема структуры «Постановка задачи»

Далее указываются периодичность решения задачи и регламент выдачи результатных документов, требования к организации сбора исходных данных, т.е. к способу и техническим средствам съема, регистрации, сбора и передачи данных для обработки. Большое значение имеет описание связи задачи с другими задачами функциональной подсистемы, в которую она входит, а также с задачами других подсистем или с внешней средой.

Описание выходной информации включает в себя: перечень и описание выходных сообщений, документов; перечень структурных единиц информации; периодичность

возникновения и сроки получения информации; наименование; идентификатор по каждой форме документа.

Описание входной информации состоит из перечня входных сообщений; перечня структурных единиц информации; описания периодичности возникновения и сроков получения информации; наименования и идентификатора по каждой форме документа.

Далее для каждой задачи разрабатываются все компоненты информационного, технического, математического и лингвистического обеспечения, а также некоторые компоненты программного обеспечения.

Результатом работ на данной стадии является утвержденный «Технический проект», состав и содержание которого регламентируются стандартом (ГОСТ 34.201 - 89).

На втором этапе - «Рабочем проектировании» осуществляется техническая реализация выбранных наилучших вариантов и разрабатывается документация «Рабочий проект» (рис. 2.6). Наиболее ответственной работой, выполняемой на этом этапе, являются «Кодирование и составление программной документации» (П1). В ее состав входят следующие компоненты (Д 1.2):

- описание программ;
- спецификация программ;
- тексты программ;
- контрольные примеры;
- инструкции для системного программиста, оператора и пользователя.

Большую роль в деле эффективного использования разработанного проекта ЭИС играет качественная технологическая документация, входящая в состав «Рабочего проекта». Эта часть проекта разрабатывается на операции П2 и предназначена для использования специалистами в своей деятельности на каждом автоматизированном рабочем месте.

В состав технологической документации (Д2.1) входят: технологические карты, разрабатываемые на процессы обработки информации при решении задач каждого класса, и инструкционные карты, составляемые на каждую технологическую операцию.

Технологическая документация разрабатывается в соответствии с требованиями ГОСТ 3.11.09 - 82 «Система технологической документации. Термины и определения основных понятий», и составляет содержание технологического обеспечения ЭИС, которое можно разделить на несколько типов в соответствии с выделением следующих классов задач, решаемых в ИС:

- системы обработки данных (СОД);
- системы поддержки принятия решений (СППР);
- системы автоматизированного проектирования новой продукции (САПР) и т.д.

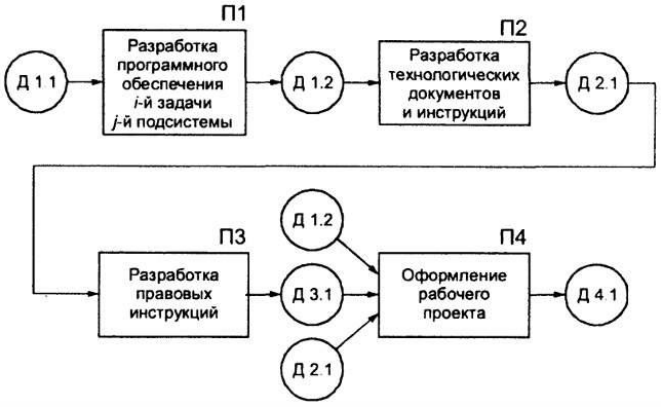


Рисунок 2.6 - ТСП работ, выполняемых на этапе рабочего проектирования:

Д 1.1 - технический проект; Д 1.2 -документы программного обеспечения; Д 2.1 - технические документы и инструкции; Д 3.1 - правовые инструкции; Д 4.1 - рабочий проект

К числу работ, выполняемых на этом этапе, относится «Разработка правовых инструкций»; (Д1.2) (П1), определяющих права и обязанности специалистов, работающих в условиях функционирования на предприятии компонентов ЭИС.

Заключительной операцией служит «Оформление документации Рабочего проекта» (Д4.2) согласно ГОСТам (Д4.1) на операции П4.

Проблемы разработки архитектуры программного приложения

Проблемы разработки архитектуры программного приложения впервые проявились в 60-х - начале 70-х годов, когда провалились многие большие проекты по разработке программных продуктов. Были зафиксированы задержки в создании ПО, оно было ненадежным, затраты на разработку в несколько раз превосходили первоначальные оценки, созданные программные системы часто имели низкие показатели производительности. Причины провалов коренились в тех подходах, которые использовались в управлении проектами. Применяемая методика была основана на опыте управления техническими проектами и оказалась неэффективной при разработке программного обеспечения.

Важно понимать разницу между профессиональной разработкой ПО и любительским программированием. Необходимость управления программными проектами вытекает из того факта, что процесс создания профессионального ПО всегда является субъектом бюджетной политики организации, где оно разрабатывается, и имеет временные ограничения. Работа руководителя программного проекта по большому счету заключается в том, чтобы гарантировать выполнение этих бюджетных и временных ограничений с учетом бизнес-целей организации относительно разрабатываемого ПО.

Руководители проектов призваны спланировать все этапы разработки программного продукта. Они также должны контролировать ход выполнения работ и соблюдения всех требуемых стандартов. Постоянный контроль за ходом выполнения работ необходим для того, чтобы процесс разработки не выходил за временные и бюджетные ограничения. Хорошее управление не гарантирует успешного завершения проекта, но плохое управление обязательно приведет к его провалу. Это может выразиться в задержке сроков сдачи готового ПО, в превышении сметной стоимости проекта и в несоответствии готового ПО спецификации требований.

Процесс разработки ПО существенно отличается от процессов реализации технических проектов, что порождает определенные сложности в управлении программными проектами:

1. Программный продукт нематериален. Программное обеспечение нематериально, его нельзя увидеть или потрогать. Руководитель программного проекта не видит процесс "роста" разрабатываемого ПО. Он может полагаться только на документацию, которая фиксирует процесс разработки программного продукта.

2. Не существует стандартных процессов разработки ПО. На сегодняшний день не существует четкой зависимости между процессом создания ПО и типом создаваемого программного продукта. Другие технические дисциплины имеют длительную историю, процессы разработки технических изделий многократно опробованы и проверены. Процессы создания большинства технических систем хорошо изучены. Изучением же процессов создания ПО специалисты занимаются только последнее время. Поэтому пока нельзя точно предсказать, на каком этапе процесса разработки ПО могут возникнуть проблемы, угрожающие всему программному проекту.

3. Большие программные проекты - это часто "одноразовые" проекты. Большие программные проекты, как правило, значительно отличаются от проектов, реализованных ранее. Поэтому, чтобы уменьшить неопределенность в планировании проекта, руководители проектов должны обладать очень большим практическим опытом. Но

постоянные технологические изменения в компьютерной технике и коммуникационном оборудовании обесценивают предыдущий опыт. Знания и навыки, накопленные опытом, могут не востребоваться в новом проекте.

Перечисленные отличия могут привести к тому, что реализация проекта выйдет из временного графика или превысит бюджетные ассигнования. Программные системы зачастую оказываются новинками как в "идеологическом", так и в техническом плане. Поэтому, предвидя возможные проблемы в реализации программного проекта, следует всегда помнить, что многим из них свойственно выходить за рамки временных и бюджетных ограничений.

Разработка архитектуры программного приложения

Невозможно описать и стандартизировать все работы, выполняемые в проекте по созданию ПО. Эти работы весьма существенно зависят от организации, где выполняется разработка ПО, и от типа создаваемого программного продукта. Но всегда можно выделить следующие:

- Написание предложений по созданию ПО.
- Планирование и составление графика работ по созданию ПО.
- Оценивание стоимости проекта.
- Подбор персонала.
- Контроль за ходом выполнения работ.
- Написание отчетов и представлений.

Первая стадия программного проекта может состоять из написания предложений по реализации этого проекта. Предложения должны содержать описание целей проектов и способов их достижения. Они также обычно включают в себя оценки финансовых и временных затрат на выполнение проекта. При необходимости здесь могут приводиться обоснования для передачи проекта на выполнение сторонней организации или команде разработчиков.

Написание предложений — очень ответственная работа, так как для многих организаций вопрос о том, будет ли проект выполняться самой организацией или разрабатываться по контракту сторонней компанией, является критическим. Не существует каких-либо рекомендаций по написанию предложений, многое здесь зависит от опыт.

На этапе планирования проекта определяются процессы, этапы и полученные на каждом из них результаты, которые должны привести к выполнению проекта. Реализация этого плана приведет к достижению целей проекта. Определение стоимости проекта напрямую связано с его планированием, поскольку здесь оцениваются ресурсы, требующиеся для выполнения плана.

Контроль за ходом выполнения работ (мониторинг проекта) — это непрерывный процесс, продолжающийся в течение всего срока реализации проекта. Руководитель должен постоянно отслеживать ход реализации проекта и сравнивать фактические и плановые показатели выполнения работ с их стоимостью. Хотя многие организации имеют механизмы формального мониторинга работ, опытный руководитель может составить ясную картину о стадии развитии проекта просто путем неформального общения с разработчиками.

Неформальный мониторинг часто помогает обнаружить потенциальные проблемы, которые в явном виде могут обнаружиться позднее. Например, ежедневное обсуждение хода выполнения работ может выявить отдельные недоработки в создаваемом программном продукте. Вместо ожидания отчетов, в которых будет отражен факт "пробуксовки" графика работ, можно обсудить со специалистами намечающиеся программистские проблемы и не допустить срыва графика работ.

В течение реализации проекта обычно происходит несколько формальных контрольных проверок хода выполнения работ по созданию ПО. Такие проверки должны

дать общую картину хода реализации проекта в целом и показать, насколько уже разработанная часть ПО соответствует целям проекта.

Время выполнения больших программных проектов может занимать несколько лет. В течение этого времени цели и намерения организации, заказавшей программный проект, могут существенно измениться. Может оказаться, что разрабатываемый программный продукт стал уже ненужным либо исходные требования к создаваемому ПО просто устарели и их необходимо кардинально менять. В такой ситуации руководство организации-разработчика может принять решение о прекращении разработки ПО или об изменении проекта в целом с тем, чтобы учесть изменившиеся цели и намерения организации-заказчика.

Руководители проектов обычно обязаны сами подбирать исполнителей для своих проектов. В идеальном случае профессиональный уровень исполнителей должен соответствовать той работе, которую они будут выполнять в ходе реализации проекта. Однако во многих случаях руководители должны полагаться на команду разработчиков, которая далека от идеальной. Такая ситуация может быть вызвана следующими причинами:

1. Бюджет проекта не позволяет привлечь высококвалифицированный персонал. В таком случае за меньшую плату привлекаются менее квалифицированные специалисты.

2. Бывают ситуации, когда невозможно найти специалистов необходимой квалификации как в самой организации-разработчике, так и вне ее. Например, в организации "лучшие люди" могут быть уже заняты в других проектах.

3. Организация хочет повысить профессиональный уровень своих работников. В этом случае она может привлечь к участию в проекте неопытных или недостаточно квалифицированных работников, чтобы они приобрели необходимый опыт и поучились у более опытных специалистов.

Таким образом, почти всегда подбор специалистов для выполнения проекта имеет определенные ограничения и не является свободным. Вместе с тем необходимо, чтобы хотя бы несколько членов группы разработчиков имели квалификацию и опыт, достаточные для работы над данным проектом. В противном случае невозможно избежать ошибок в разработке ПО.

Руководитель проекта обычно обязан посылать отчеты о ходе его выполнения как заказчику, так и подрядным организациям. Это должны быть краткие документы, основанные на информации, извлекаемой из подробных' отчетов о проекте. В этих отчетах должна быть та информация, которая позволяет четко оценить степень готовности создаваемого программного продукта.

В рамках курса «Системная инженерия» выделены следующие роли в группе по разработке ПО:

- Руководитель – общее руководство проектом, написание документации, общение с заказчиком ПО;
- Системный аналитик – разработка требований (составление технического задания, проекта программного обеспечения);
- Тестер – составление плана тестирования и аттестации готового ПО (продукта), составление сценария тестирования, базовый пример, проведение мероприятий по плану тестирования;
- Разработчик – моделирование компонент программного обеспечения, кодирование.

Планирование процесса разработки архитектуры программного приложения

Эффективное управление программным проектом напрямую зависит от правильного планирования работ, необходимых для его выполнения. План помогает руководителю предвидеть проблемы, которые могут возникнуть на каких-либо этапах создания ПО, и разработать превентивные меры для их предупреждения или решения. План, разработанный на начальном этапе проекта, рассматривается всеми его

участниками как руководящий документ, выполнение которого должно привести к успешному завершению проекта. Этот первоначальный план должен максимально подробно описывать все этапы реализации проекта.

Процесс планирования начинается, исходя из описания системы, с определения проектных ограничений (временные ограничения, возможности наличного персонала, бюджетные ограничения и т.д.). Эти ограничения должны определяться параллельно с оцениванием проектных параметров, таких как структура и размер проекта, а также распределением функций среди исполнителей. Затем определяются этапы разработки и то, какие результаты документация, прототипы, подсистемы или версии программного продукта) должны быть получены по окончании этих этапов. Далее начинается циклическая часть планирования. Сначала разрабатывается график работ по выполнению проекта или дается разрешение на продолжение использования ранее созданного графика. После этого проводится контроль выполнения работ и отмечаются расхождения между реальным и плановым ходом работ.

Далее, по мере поступления новой информации о ходе выполнения проекта, возможен пересмотр первоначальных оценок параметров проекта. Это, в свою очередь, может привести к изменению графика работ. Если в результате этих изменений нарушаются сроки завершения проекта, должны быть пересмотрены (и согласованы с заказчиком ПО) проектные ограничения.

Конечно, большинство руководителей проектов не думают, что реализация их проектов пройдет гладко, без всяких проблем. Желательно описать возможные проблемы еще до того, как они проявят себя в ходе выполнения проекта. Поэтому лучше составлять "пессимистические" графики работ, чем "оптимистические". Но, конечно, невозможно построить план, учитывающий все, в том числе случайные, проблемы и задержки выполнения проекта, поэтому и возникает необходимость периодического пересмотра проектных ограничений и этапов создания программного продукта.

План проекта должен четко показать ресурсы, необходимые для реализации проекта, разделение работ на этапы и временной график выполнения этих этапов. В некоторых организациях план проекта составляется как единый документ, содержащий все виды планов, описанных выше. В других случаях план проекта описывает только технологический процесс создания ПО. В таком плане обязательно присутствуют ссылки на планы других видов, но они разрабатываются отдельно от плана проекта.

Детализация планов проектов очень разнится в зависимости от типа разрабатываемого программного продукта и организации-разработчика. Но в любом случае большинство планов содержат следующие разделы.

1. Введение. Краткое описание целей проекта и проектных ограничений (бюджетных, временных и т.д.), которые важны для управления проектом.
2. Организация выполнения проекта. Описание способа подбора команды разработчиков и распределение обязанностей между членами команды.
3. Анализ рисков. Описание возможных проектных рисков, вероятности их проявления и стратегий, направленных на их уменьшение.
4. Аппаратные и программные ресурсы, необходимые для реализации проекта. Перечень аппаратных средств и программного обеспечения, необходимого для разработки программного продукта. Если аппаратные средства требуется закупать, приводится их стоимость совместно с графиком закупки и поставки.
5. Разбиение работ на этапы. Процесс реализации проекта разбивается на отдельные процессы, определяются этапы выполнения проекта, приводится описание результатов ("выходов") каждого этапа и контрольные отметки.
6. График работ. В этом графике отображаются зависимости между отдельными процессами (этапами) разработки ПО, оценки времени их выполнения и распределение членов команды разработчиков по отдельным этапам.

7. Механизмы мониторинга и контроля за ходом выполнения проекта. Описываются предоставляемые руководителем отчеты о ходе выполнения работ, сроки их предоставления, а также механизмы мониторинга всего проекта.

План должен регулярно пересматриваться в процессе реализации проекта. Одни части плана, например график работ, изменяются часто, другие более стабильны. Для внесения изменений в план требуется специальная организация документопотока, позволяющая отслеживать эти изменения.

Общие сведения о требованиях к архитектуре программного приложения

Проблемы, которые приходится решать специалистам в процессе создания программного обеспечения, очень сложны. Природа этих проблем не всегда ясна, особенно если разрабатываемая программная система инновационная. В частности, трудно чётко описать те действия, которые должна выполнять система. Описание функциональных возможностей и ограничений, накладываемых на систему, называется требованиями к этой системе, а сам процесс формирования, анализа, документирования и проверки этих функциональных возможностей и ограничений – разработкой требований.

Требования подразделяются на пользовательские и системные. Пользовательские требования – это описание на естественном языке (плюс поясняющие диаграммы) функций, выполняемых системой, и ограничений, накладываемых на неё. Системные требования – это описание особенностей системы (архитектура системы, требования к параметрам оборудования и т.д.), необходимых для эффективной реализации требований пользователя.

Разработка архитектуры программного приложения: анализ осуществимости

Разработка требований — это процесс, включающий мероприятия, необходимые для создания и утверждения документа, содержащего спецификацию системных требований. Для новых программных систем процесс разработки требований должен начинаться с анализа осуществимости. Началом такого анализа является общее описание системы и ее назначения, а результатом анализа — отчет, в котором должна быть четкая рекомендация, продолжать или нет процесс разработки требований проектируемой системы. Другими словами, анализ осуществимости должен осветить следующие вопросы.

1. Отвечает ли система общим и бизнес-целям организации-заказчика и организации-разработчика?
2. Можно ли реализовать систему, используя существующие на данный момент технологии и не выходя за пределы заданной стоимости?
3. Можно ли объединить систему с другими системами, которые уже эксплуатируются?

Критическим является вопрос, будет ли система соответствовать целям организации. Если система не соответствует этим целям, она не представляет никакой ценности для организации. В то же время многие организации разрабатывают системы, не соответствующие их целям, либо не совсем ясно понимая эти цели, либо под влиянием политических или общественных факторов.

Выполнение анализа осуществимости включает сбор и анализ информации о будущей системе и написание соответствующего отчета. Сначала следует определить, какая именно информация необходима, чтобы ответить на поставленные выше вопросы. Например, эту информацию можно получить, ответив на следующее:

1. Что произойдет с организацией, если система не будет введена в эксплуатацию?
2. Какие текущие проблемы существуют в организации и как новая система поможет их решить?
3. Каким образом система будет способствовать целям бизнеса?
4. Требует ли разработка системы технологии, которая до этого не использовалась в организации?

Далее необходимо определить источники информации. Это могут быть менеджеры отделов, где система будет использоваться, разработчики программного обеспечения, знакомые с типом будущей системы, технологи, конечные пользователи и т.д.

После обработки собранной информации готовится отчет по анализу осуществимости создания системы. В нем должны быть даны рекомендации относительно продолжения разработки системы. Могут быть предложены изменения бюджета и графика работ по созданию системы или предъявлены более высокие требования к системе.

5.3 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Базовые понятия ERD

Сущность (Entity) — множество экземпляров реальных или абстрактных объектов (людей, событий, состояний, идей, предметов и др.), обладающих общими атрибутами или характеристиками. Любой объект системы может быть представлен только одной сущностью, которая должна быть уникально идентифицирована. При этом имя сущности должно отражать тип или класс объекта, а не его конкретный экземпляр (например, АЭРОПОРТ, а не ВНУКОВО).

Каждая сущность должна обладать уникальным идентификатором. Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа сущности. Каждая сущность должна обладать некоторыми свойствами:

- иметь уникальное имя; к одному и тому же имени должна всегда применяться одна и та же интерпретация; одна и та же интерпретация не может применяться к различным именам, если только они не являются псевдонимами;
- иметь один или несколько атрибутов, которые либо принадлежат сущности, либо наследуются через связь;
- иметь один или несколько атрибутов, которые однозначно идентифицируют каждый экземпляр сущности.

Каждая сущность может обладать любым количеством связей с другими сущностями модели.

Связь (Relationship) — поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Связь — это ассоциация между сущностями, при которой каждый экземпляр одной сущности ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности, и наоборот.

Атрибут (Attribute) — любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Атрибут представляет тип характеристик или свойств, ассоциированных с множеством реальных или абстрактных объектов (людей, мест, событий, состояний, идей, предметов и т.д.). Экземпляр атрибута — это определенная характеристика отдельного элемента множества. Экземпляр атрибута определяется типом характеристики и ее значением, называемым значением атрибута. На диаграмме "сущность-связь" атрибуты ассоциируются с конкретными сущностями. Таким образом, экземпляр сущности должен обладать единственным определенным значением для ассоциированного атрибута.

Метод IDEFI

Наиболее распространенными методами для построения ERD-диаграмм являются метод Баркера и метод IDEFI.

Метод Баркера основан на нотации, предложенной автором, и используется в case-средстве Oracle Designer.

Метод IDEFI основан на подходе Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме. На основе совершенствования метода IDEFI создана его новая версия — метод IDEFIX, разработанный с учетом таких требований, как простота для изучения и возможность автоматизации. IDEFIX-диаграммы используются в ряде распространенных CASE-средств (в частности, ERwin, Design/IDEF).

В методе IDEFIX сущность является независимой от идентификаторов или просто независимой, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями. Сущность называется зависимой от идентификаторов или просто зависимой, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности.

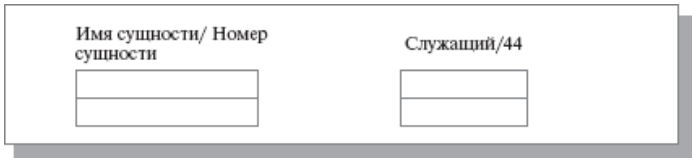


Рисунок 3.1 - Независимые от идентификации сущности

Рисунок 3.2 - Зависимые от идентификации сущности

Каждой сущности присваиваются уникальные имя и номер, разделяемые косой чертой "/" и помещаемые над блоком.

Связь может дополнительно определяться с помощью указания степени или мощности (количества экземпляров сущности-потомка, которое может порождать каждый экземпляр сущности-родителя). В IDEFIX могут быть выражены следующие мощности связей:

- каждый экземпляр сущности-родителя может иметь ноль, один или более одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не менее одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не более одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя связан с некоторым фиксированным числом экземпляров сущности-потомка.

Если экземпляр сущности-потомка однозначно определяется своей связью с сущностью-родителем, то связь называется идентифицирующей, в противном случае — неидентифицирующей.

Связь изображается линией, проводимой между сущностью-родителем и сущностью-потомком, с точкой на конце линии у сущности-потомка. Мощность связей может принимать следующие значения: N — ноль, один или более, Z — ноль или один, P — один или более. По умолчанию мощность связей принимается равной N.

Рисунок 3.3 - Графическое изображение мощности связи

Идентифицирующая связь между сущностью-родителем и сущностью-потомком изображается сплошной линией. Сущность-потомок в идентифицирующей связи является зависимой от идентификатора сущностью. Сущность-родитель в идентифицирующей связи может быть как независимой, так и зависимой от идентификатора сущностью (это определяется ее связями с другими сущностями).

Пунктирная линия изображает неидентифицирующую связь. Сущность-потомок в неидентифицирующей связи будет не зависимой от идентификатора, если она не является также сущностью-потомком в какой-либо идентифицирующей связи.

Атрибуты изображаются в виде списка имен внутри блока сущности. Атрибуты, определяющие первичный ключ, размещаются наверху списка и отделяются от других атрибутов горизонтальной чертой.

Сущности могут иметь также внешние ключи (Foreign Key), которые могут использоваться в качестве части или целого первичного ключа или неключевого атрибута. Для обозначения внешнего ключа внутрь блока сущности помещают имена атрибутов, после которых следуют буквы FK в скобках.

Рисунок 3.4 - Неидентифицирующая связь

Документирование модели

Многие СУБД имеют ограничение на именование объектов (например, ограничение на длину имени таблицы или запрет использования специальных символов — пробела и т. п.). Зачастую разработчики ИС имеют дело с нелокализованными версиями СУБД. Это означает, что объекты БД могут называться короткими словами, только латинскими символами и без использования специальных символов (т. е. нельзя назвать таблицу, используя предложение — ее можно назвать только одним словом). Кроме того, проектировщики БД нередко злоупотребляют "техническими" наименованиями, в результате таблица и колонки получают наименования типа RTD_324 или CUST_A12 и т.д. Полученную в результате структуру могут понять только специалисты (а чаще всего — только авторы модели), ее невозможно обсуждать с экспертами предметной области. Разделение модели на логическую и физическую позволяет решить эту проблему. На физическом уровне объекты БД могут называться так, как того требуют ограничения СУБД. На логическом уровне можно этим объектам дать синонимы — имена более понятные неспециалистам, в том числе на кириллице и с использованием специальных символов. Например, таблице CUST_A12 может соответствовать сущность Постоянный клиент. Такое соответствие позволяет лучше документировать модель и дает возможность обсуждать структуру данных с экспертами предметной области.

Уровни логической модели

Различают три уровня логической модели, отличающихся по глубине представления информации о данных:

- диаграмма сущность-связь (Entity Relationship Diagram, ERD);
- модель данных, основанная на ключах (Key Based model, KB);
- полная атрибутивная модель (Fully Attributed model, FA).

Диаграмма сущность-связь представляет собой модель данных верхнего уровня. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Такая диаграмма не слишком детализирована, в нее включаются основные сущности и связи между ними, которые удовлетворяют основным требованиям, предъявляемым к ИС. Диаграмма сущность-связь может включать связи "многие-ко-многим" и не включать описание ключей. Как правило, ERD используется для презентаций и обсуждения структуры данных с экспертами предметной области.

Модель данных, основанная на ключах, — более подробное представление данных. Она включает описание всех сущностей и первичных ключей и предназначена для представления структуры данных и ключей, которые соответствуют предметной области.

Полная атрибутивная модель — наиболее детальное представление структуры данных: представляет данные в третьей нормальной форме и включает все сущности, атрибуты и связи.

Создание физической модели данных

Физическая модель содержит всю информацию, необходимую для реализации конкретной БД. Различают два уровня физической модели:

- трансформационную модель;
- модель СУБД.

Трансформационная модель содержит информацию для реализации отдельного проекта, который может быть частью общей ИС и описывать подмножество предметной области. Данная модель позволяет проектировщикам и администраторам БД лучше представить, какие объекты БД хранятся в словаре данных, и проверить, насколько физическая модель удовлетворяет требованиям к ИС.

Модель СУБД автоматически генерируется из трансформационной модели и является точным отображением системного каталога СУБД.

Физический уровень представления модели зависит от выбранного сервера. ERwin поддерживает более 20 реляционных и нереляционных БД.

Модель сущность-связь

Модель была предложена Петером Пин-Шен Ченом в 1976 г. На использовании разновидностей ER-модели основано большинство современных подходов к проектированию баз данных (главным образом, реляционных). Моделирование предметной области базируется на использовании графических диаграмм, включающих небольшое число разнородных компонентов. В связи с наглядностью представления концептуальных схем баз данных ER-модели получили широкое распространение в CASE-системах, поддерживающих автоматизированное проектирование реляционных баз данных. Базовыми понятиями ER-модели являются **сущность, связь** и **атрибут.**

Сущность - это реальный или воображаемый объект, информация о котором представляет интерес. В диаграммах ER-модели сущность пред-ставляется в виде прямоугольника, содержащего имя сущности. При этом имя сущности - это имя типа, а не конкретного объекта - экземпляра этого типа. Каждый экземпляр сущности должен быть отличим от любого другого экземпляра той же сущности.

Связь - это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация всегда является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь). В любой связи выделяются два конца (в соответствии с парой связываемых сущностей), на

каждом из которых указывается имя конца связи, степень конца связи (сколько экземпляров данной сущности связывается), обязательность связи (т. е. любой ли экземпляр данной сущности должен участвовать в данной связи).

Связь представляется в виде линии, связывающей две сущности или ведущей от сущности к ней же самой. При этом в месте "стыковки" связи с сущностью используются трехточечный вход в прямоугольник сущности, если для этой сущности в связи могут использоваться много экземпляров сущности, и одноточечный вход, если в связи может участвовать только один экземпляр сущности. Обязательный конец связи изображается сплошной линией, а необязательный - прерывистой линией.

Как и сущность, связь - это типовое понятие, все экземпляры обеих пар связываемых сущностей подчиняются правилам связывания.

На рис. приведен пример изображения сущностей и связи между ними.

Пример связи между сущностями

Данная диаграмма может быть интерпретирована следующим образом: Каждый СТУДЕНТ учится только в одной ГРУППЕ; Любая ГРУППА состоит из одного или более СТУДЕНТОВ. На следующем рисунке изображена сущность ЧЕЛОВЕК с рекурсивной связью, связывающей ее с ней же самой.

Пример рекурсивной связи

Лаконичной устной трактовкой изображенной диаграммы является следующая:

Каждый ЧЕЛОВЕК является сыном одного и только одного ЧЕЛОВЕКА;

Каждый ЧЕЛОВЕК может являться отцом для одного или более ЛЮДЕЙ ("ЧЕЛОВЕК").

Атрибутом сущности является любая деталь, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности. Имена атрибутов заносятся в прямоугольник, изображающий сущность, под именем сущности и изображаются малыми буквами. Например:

Изображение сущности с ее атрибутами

Уникальным идентификатором сущности является атрибут, комбинация атрибутов, комбинация связей или комбинация связей и атрибутов, уникально отличающая любой экземпляр сущности от других экземпляров сущности того же типа.

Как и в реляционных схемах баз данных, в ER-схемах вводится понятие нормальных форм, причем их смысл очень близко соответствует смыслу реляционных нормальных форм. Заметим, что формулировки нормальных форм ER-схем делают более понятным смысл нормализации реляционных схем. Мы рассмотрим только очень краткие и неформальные определения трех первых нормальных форм.

В *первой нормальной форме* ER-схемы устраняются повторяющиеся атрибуты или группы атрибутов, т. е. производится выявление неявных сущностей, "замаскированных" под атрибуты.

Во *второй нормальной форме* устраняются атрибуты, зависящие только от части уникального идентификатора. Эта часть уникального идентификатора определяет отдельную сущность.

В *третьей нормальной форме* устраняются атрибуты, зависящие от атрибутов, не входящих в уникальный идентификатор. Эти атрибуты являются основой отдельной сущности.Мы остановились только на самых важных понятиях ER-модели данных. К числу более сложных элементов модели относятся следующие:

Подтипы и супертипы сущностей. ER-модель позволяет задавать отношение IS-A между типами. При этом если T_1 IS-A T_2 (где T_1 и T_2 - типы сущностей), то T_1 называется подтипом T_2 а T_2- супертипом T_1. Т.о., существует возможность наследования типа сущности, исходя из одного или нескольких супертипов.

Связи "многие-со-многими". Иногда бывает необходимо связывать сущности таким образом, что с обоих концов связи могут присутствовать несколько экземпляров сущности (например, все члены кооператива сообща владеют имуществом кооператива). Для этого вводится разновидность связи "многие-со-многими".

Уточняемые степени связи. Иногда бывает полезно определить возможное количество экземпляров сущности, участвующих в данной связи (например, служащему разрешается участвовать не более чем в трех проектах одновременно). Для выражения этого семантического ограничения разрешается указывать на конце связи ее максимальную или обязательную степень.

Каскадные удаления экземпляров сущностей. Некоторые связи бывают настолько сильными (конечно, в случае связи "один-ко-многим"), что при удалении опорного экземпляра сущности (соответствующего концу связи "один") нужно удалить и все экземпляры сущности, соответствующие концу связи "многие". Соответствующее требование "каскадного удаления" можно сформулировать при определении сущности.

Домены. Как и в случае реляционной модели данных, бывает полезна возможность определения потенциально допустимого множества значений атрибута сущности (домена).

Эти и другие, более сложные элементы модели данных "Сущность-Связь", делают ее более мощной, но одновременно несколько усложняют ее использование. Конечно, при реальном использовании ER-диаграмм для проектирования баз данных необходимо ознакомиться со всеми возможностями.

Программные средства проектирования реляционных баз данных

BPwin - инструмент моделирования с возможностью анализа, документирования и корректирования бизнес процессов. Он поможет устранить лишние или неэффективные операции, уменьшить издержки, повысить гибкость и улучшить уровень обслуживания заказчика. В модели BPwin, Вы можете четко задокументировать важные позиции, такие как необходимые операции, проследить, как они выполняются и какие необходимы для этого ресурсы. Модель BPwin обеспечивает интегрированное изображение того, как работает ваша организация. Это изображение, в свою очередь, состоит из подмоделей отделов, пример которых приведен ниже в общей диаграмме дерева узлов. BPwin поддерживает двунаправленные линии связи с программой ERwin.

ERwin - инструмент моделирования данных для разработки базы данных, который поддерживает методологии IDEF1X и IE . Вы можете совместно использовать BPwin и ERwin как для моделирования процесса, так и для моделирования данных и Вы можете обмениваться объектами и названиями атрибута между двумя программами. Эта возможность особенно полезна для пользователей, кто разрабатывает модель бизнес процесса и модель базы данных одновременно. Любые изменения, которые Вы проводите в объекте и названии атрибута в любой модели, могут быть внесены в другую модель. Вы можете также создать двухсторонние линии связи между моделью BPwin и связанной с ней моделью ERwin, что выполняется, автоматически каждый раз, когда Вы открываете одну из связанных диаграмм. Это проводится для синхронизации моделей.

Средство автоматизированного проектирования баз данных ERwin

ERwin - CASE-средство проектирования баз данных фирмы Platinum. ERwin сочетает графический интерфейс Windows, инструменты для построения ER-диаграмм, редакторы для создания логического и физического описания модели данных и прозрачную поддержку ведущих реляционных СУБД. Для удобства изложения материала здесь и далее использована оригинальная терминология, принятая в ERwin.

ERwin не привязан к технологии какой-либо конкретной фирмы, поставляющей СУБД или средства разработки. Он поддерживает различные серверы баз данных и настольные СУБД, а также может обращаться к базе данных через интерфейс ODBC.

ERwin можно использовать совместно с некоторыми популярными средствами разработки клиентских частей приложений: PowerBuilder, Visual Basic, Delphi. Кроме того, ERwin поддерживает работу в среде групповой разработки ModelMart, являющейся продуктом той же Platinum.

Процесс моделирования в ERwin базируется на методологии проектирования реляционных баз данных IDEF1X. Данная методология была разработана для ВВС США и теперь широко используется в правительственных учреждениях и частных компаниях как в самих США, так и далеко за их пределами. Она определяет стандарты терминологии и графического изображения типовых элементов на ER-диаграммах.

Структура процесса моделирования в ERwin

В ERwin используются два уровня представления модели данных: логический и физический (что соответствует концептуальному и логическому уровню, принятым в теории БД). На логическом уровне не рассматривается использование конкретной СУБД, не определяются типы данных (например, целое или вещественное число) и не определяются индексы для таблиц. Целевая СУБД, имена объектов и типы данных, индексы составляют второй (физический) уровень модели ERwin.

ERwin предоставляет возможности создавать и управлять этими двумя различными уровнями представления одной диаграммы (модели), равно как и иметь много вариантов отображения на каждом уровне.

Процесс построения информационной модели состоит из следующих этапов:

1. Создание логической модели данных:
   - определение сущностей;
   - определение зависимостей между сущностями;
   - задание первичных и альтернативных ключей;
   - определение неключевых атрибутов сущностей.
2. Переход к физическому описанию модели:
   - назначение соответствий имя сущности - имя таблицы, атрибут сущности - атрибут таблицы;
   - задание триггеров, хранимых процедур и ограничений.
3. Генерация базы данных.

8.3. Создание логической модели базы данных

С точки зрения пользователя ERwin, процесс создания логической модели данных заключается в визуальном редактировании ER-диаграммы. Диаграмма ERwin строится из трех основных блоков: сущностей, атрибутов и связей.

На диаграмме сущность изображается прямоугольником. В зависимости от режима представления диаграммы прямоугольник может содержать имя сущности, ее описание, список ее атрибутов и другие сведения. Основная информация, описывающая сущность, включает:

- атрибуты, составляющие первичный ключ;
- неключевые атрибуты;
- тип сущности (независимая/зависимая).

Первичный ключ - это атрибут или набор атрибутов, уникально идентифицирующий экземпляр сущности. Если несколько наборов атрибутов могут уникально

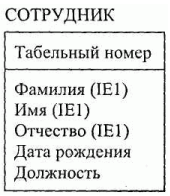
идентифицировать сущность, то выбор одного из них осуществляется разработчиком на основании анализа предметной области и учета следующих требований к первичному ключу.

- первичный ключ не должен принимать пустые (NULL) значения;
- первичный ключ не должен изменяться в течение времени;
- размер первичного ключа должен быть как можно меньшим.

При этом если разработчик считает, что какой-либо из оставшихся наборов будет часто использоваться для доступа к сущности, то он может объявить его *альтернативным ключом.*

В ERwin можно также составлять группы атрибутов, которые не идентифицируют уникально экземпляры сущности, но часто используются для доступа к данным. Они получили название *инверсных входов.* Одни и те же атрибуты сущности могут входить в несколько различных групп ключей.

Рассмотрим выше сказанное на примере сущности СОТРУДНИК (рис. 2.5).



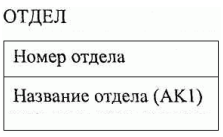
Пример сущности

Среди всех атрибутов данной сущности на роль первичного ключа могут претендовать "табельный номер" и группа атрибутов "фамилия", "имя","отчество", "дата рождения" (последний необходим, т. к. на предприятии могут работать полные тезки). Очевидно, что по соображению размера в качестве первичного ключа следует выбрать первый из вариантов.

На диаграмме атрибуты, составляющие первичный ключ, располагаются в верхней части прямоугольника и отделяются от прочих (не входящих в первичных ключ) горизонтальной линией.

Группа атрибутов "фамилия", "имя", "отчество", "дата рождения" может являться альтернативным ключом. Однако вряд ли кто-либо, пытающийся найти информацию о сотруднике, будет знать дату его рождения. А вот группа атрибутов "фамилия", "имя", "отчество", вполне возможно, будет достаточно часто использоваться для этих целей. Поэтому на основе этих атрибутов было бы логично создать инверсный вход. Инверсный вход обозначается на диаграмме символами IEn, заключенными в скобки.

Пример использования альтернативного ключа приведен на рис. 2.6. Альтернативный ключ обозначается на диаграмме символами АКп, заключенными в скобки.



Пример альтернативного ключа

Если экземпляры сущности могут быть уникально идентифицированы без определения ее связей с другими сущностями, она называется *независимой.* В противном случае сущность называют *зависимой.* Зависимая сущность отображается в ERwin прямоугольником с закругленными углами.

Связь в ERwin трактуется как функциональная зависимость между двумя сущностями (в частности, возможна связь сущности с самой собой).

Если рассматривать диаграмму как графическое представление правил предметной области, то сущности являются существительными, а связи -глаголами. Например, между сущностями ОТДЕЛ и СОТРУДНИК существует связь "состоит из" (ОТДЕЛ состоит из СОТРУДНИКОВ).

В ERwin связи представлены пятью основными элементами информации: тип связи;

- родительская и дочерняя (зависимая) сущности;• мощность связи;
- допустимость пустых (null) значений;
- требования по обеспечению ссылочной целостности.

ERwin поддерживает следующие основные типы связей: идентифицирующая, неидентифицирующая, полная категория, неполная категория, многие-ко-многим.

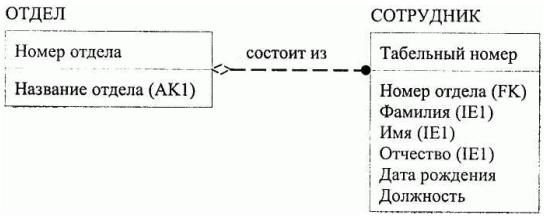
Связь называется *идентифицирующей,* если экземпляр дочерней сущности идентифицируется через ее связь с родительской сущностью. Атрибуты, составляющие первичный ключ родительской сущности, при этом входят в первичный ключ дочерней сущности. Дочерняя сущность при идентифицирующей связи всегда является зависимой.

Связь называется *неидентифицирующей,* если экземпляр дочерней сущности идентифицируется иначе, чем через связь с родительской сущностью. Атрибуты, составляющие первичный ключ родительской сущности, при этом входят в состав неключевых атрибутов дочерней сущности.

Идентифицирующая связь изображается сплошной линией; неидентифицирующая - пунктирной линией. Линии заканчиваются точкой со стороны дочерней сущности.

При определении связи происходит миграция атрибутов первичного ключа родительской сущности в соответствующую область атрибутов дочерней сущности. Поэтому такие атрибуты не вводятся вручную.

На рис. приведен пример неидентифицируюшей связи. Первичный ключ сущности ОТДЕЛ "номер отдела" мигрировал в область неключевых атрибутов (поскольку связь неидентифицируюшая) сущности СОТРУДНИК. На диаграмме атрибуты, наследованные от родительской сущности, помечаются символами FK, заключенными в скобки.



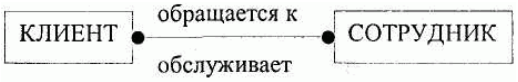
Пример неидентифицирующей связи

Зависимая сущность может наследовать один и тот же атрибут от более чем одной родительской сущности или от одной и той же родительской сущности через несколько связей.

Поскольку атрибуты первичного ключа родительской сущности по умолчанию мигрируют со своими именами, ERwin считает, что в зависимой сущности атрибуты внешнего ключа появляются только один раз. Чтобы избежать этого ограничения, ERwin позволяет ввести для них роли, т. е. новые имена, под которыми мигрирующие атрибуты будут представлены в дочерней сущности. В случае неоднократной миграции атрибута такое переименование необходимо. Например, при создании модели сделки по обмену валюты сущность СДЕЛКА (рис. 2.8) должна иметь два различных атрибута для кодов проданной и купленной валюты. В данном случае первичный ключ сущности ВАЛЮТА ("код валюты") имеет две роли в дочерней сущности.

Пример использования ролей

Ситуация, когда экземпляру одной сущности соответствует один или несколько экземпляров второй сущности, а экземпляру второй сущности соответствует один или несколько экземпляров первой сущности, отражается в логической модели связью *многие-ко-мпогим* между данными сущностями. На диаграмме связь изображается сплошной линией с точками на концах. Например, для заключения сделки в некоторой фирме клиент обращается к любому из свободных сотрудников этой фирмы. В то же время сотрудник фирмы может обслуживать нескольких клиентов. Поэтому тип связи между сущностями КЛИЕНТ и СОТРУДНИК должен быть мно-гие-ко-многим (рис. 3.9).



Пример связи многие-ко-мпогим

Заметим, что связь типа многие-ко-многим возможна только на логическом уровне. Преобразование связи данного типа на физическом уровне будет рассмотрена в следующем пункте. Однако добавим, что связи многие-ко-многим рекомендуется избегать. В рассмотренном примере этого можно добиться, если ввести дополнительную сущность СДЕЛКА.

Пример устранения связи многие-ко-многим

Некоторые сущности определяют целую категорию объектов одного типа. В ERwin в таком случае создается сущность для определения категории и для каждого элемента категории, а затем вводится для них *связь категоризации.* Родительская сущность категории называется *супертипом,* а дочерние - *подтипом.*

Различная часть (например, данные почасовой оплаты для временных работников или данные о зарплате и отпуске для штатных работников) помещается в сущности-подтипы. В сущности-супертипе вводится атрибут-дискриминатор, позволяющий различать конкретные экземпляры сущности-подтипа.

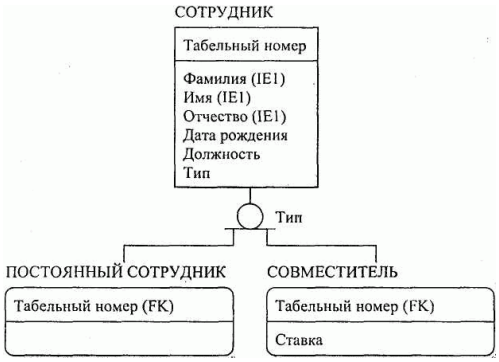
В зависимости от того, все ли возможные сущности-подтипы включены в модель, категорийная связь является *полной* или *неполной.* В ERwin полная категория изображается окружностью с двумя подчеркиваниями, а неполная - окружностью с одним подчеркиванием.

На рис. категорийная связь между сущностью СОТРУДНИК и сущностями ПОСТОЯННЫЙ СОТРУДНИК и СОВМЕСТИТЕЛЬ является неполной, если предположить, что существует еще один тип сотрудников - консультант.

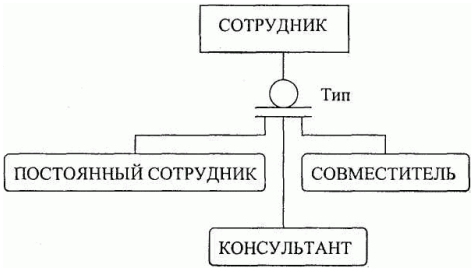
Включение в модель сущности КОНСУЛЬТАНТ приводит к тому, что категорийная связь становится полной.

Мощность связи представляет собой отношение количества экземпляров родительской сущности к соответствующему количеству экземпляров дочерней сущности. Мощность связи определяется только для идентифицирующих и неидентифицирующих связей и записывается как l:n. ERwin, в соответствии с

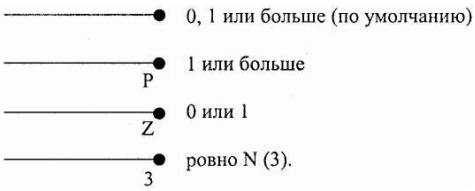
методологией IDEF1X, предоставляет 4 варианта для п, которые изображаются дополнительным символом у дочерней сущности (рис.



Пример неполной категории



Пример полной категории



Обозначение кратности связи

Допустимость пустых (NULL) значений в неидентифицирующих связях ERwin изображает пустым ромбиком на дуге связи со стороны родительской сущности.

В целях контроля ссылочной целостности (под *ссылочной целостностью* в ERwin понимается обеспечение требования, чтобы значения внешнего ключа экземпляра дочерней сущности соответствовали значениям первичного ключа в родительской сущности) для каждой связи могут быть заданы требования по обработке операций INSERT/UPDATE/DELETE для родительской и дочерней сущности. ERwin представляет следующие варианты обработки этих событий:

- отсутствие проверки;
- проверка допустимости;
- запрет операции;
- каскадное выполнение операции (DELETE/UPDATE);
- установка пустого (NULL-значения) или заданного значения по умолчанию.

Постановка задачи «Оперативный анализ прибыли и убытков по товарам в супермаркете»

Постановка задачи пользователем требует от него выполнения комплексов операций в последовательности, определяемой логикой их внутренней взаимосвязи, что отражает технологию этого процесса. Рассмотрим пример постановки задачи «Оперативный анализ прибыли и убытков по товарам в супермаркете».

Комплекс № 1 «Организационно-экономическая сущность задачи». В данном комплексе осуществляются операции по определению назначения задачи, ее цели, периодичности и сроков выполнения. В этом же комплексе отражаются информационные взаимосвязи подразделений объекта, и при этом обращается внимание на внешние и внутренние связи подразделения, в котором решается задача. Затем раскрывается информационная взаимосвязь входной и выходной информации.

Назначение задачи уточняет область ее применения, что отражается в конкретизации объекта, в котором осуществляется автоматизация информационных процессов. В рассматриваемом примере задача предназначена для торгового предприятия типа супермаркета.

Цель отражает четкое, но достаточно общее описание результата, который ожидается получить в итоге постановки задачи и ее последующей реализации с помощью технических и программных средств. Цель рассматриваемой задачи заключается в своевременном получении информации для принятия решения относительно эффективности торговли и необходимости закупки новой партии товаров.

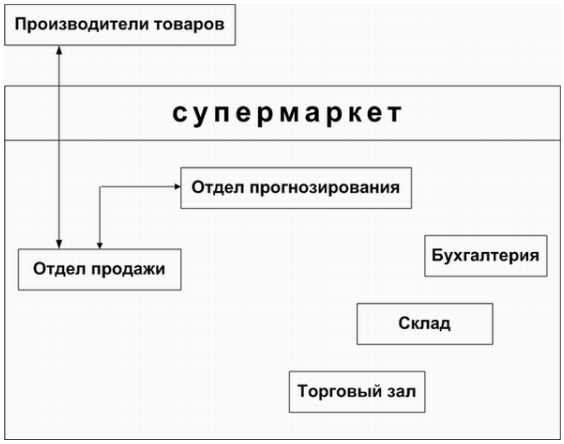


Рисунок 3.10 - Информационная взаимосвязь подразделений супермаркета

Периодичность и сроки решения задачи конкретизируют частоту потребности работника управления в информации (например, один раз в год, ежемесячно, по мере необходимости и т.п.). При этом оговариваются дата (число, месяц, год) и время дня суток (например, к десяти часам ежедневно). Данная задача решается в реальном времени, при котором обеспечивается доступ к базе данных по мере необходимости.

Информационная взаимосвязь подразделений данного экономического объекта позволяет определить состав взаимосвязанных подразделений объекта и место подразделения, для функционирования которого необходимо решение данной задачи. Пример отражения информационной взаимосвязи подразделений супермаркета и выделение конкретного подразделения (в частности, отдела продаж) приведен на рисунке 3.10.

Рисунок 3.11 - Внешние и внутренние информационные связи отдела продажи

При изучении внешних и внутренних информационных связей подразделения раскрывается его структура и указывается конкретная информация, которая должна поступать на входе данного подразделения и на выходе. Пример отражения внешних и внутренних информационных связей подразделения представлен на рис. 12.6.

Заключительной операцией в этом комплексе является отражение информационной взаимосвязи входной и выходной информации. Операция акцентирует внимание на уровнях детализации и обобщения информации. Пример взаимосвязи информации представлен на рис.3.11.

Рисунок 3.12 - Информационная взаимосвязь входной и выходной информации

Комплекс № 2 «Описание выходной информации». В данном комплексе осуществляются операции по определению состава реквизитов выходной информации, расположению реквизитов выходной информации с отражением контрольного примера, описанию полей (реквизитов) выходного документа.

Определение *состава реквизитов выходной информации* зависит от поставленной перед задачей цели; состав реквизитов должен быть необходимым и достаточным для организации работы специалиста подразделения.

Последовательность *расположения реквизитов* определяется правилами распределения реквизитов по частям документа (заголовочной, содержательной, оформительской) и отдельным зонам. Внутри зон реквизиты также располагаются по

установленным правилам (удобство работы пользователя, специфика отражения итогов, акцентирование внимания на отдельных реквизитах и т.п.). В результате этой операции создается эскиз выходного документа с отображением *контрольного примера.* В контрольном примере дается логика расчета, при этом используются числа, легко подсчитываемые вручную.

Заключительной операцией этого комплекса является *описание полей (реквизитов) выходного документа,* или иначе - представление структуры выходного документа. По рассматриваемой задаче структура выходного документа представлена в табл. 12.1. В таблице идентификация отражает короткое, легко запоминающееся название поля в латинском алфавите. Тип данных подчеркивает текстовую или числовую основу данных. В данном примере представлен только числовой тип данных. Разрядность по каждому реквизиту указывается максимальная.

В комплексе 2 при проектировании выходного документа учитывается также влияние программных и технических средств (информационная емкость экрана, ширина печатающего устройства, возможность получения нескольких экземпляров и т.п.). В этом же комплексе обобщается специфика выходной информации: рассматриваются состав потребителей информации, способы передачи, объемно-временные характеристики, особенности контроля данных.

Данный комплекс конкретизирует ответ на вопрос: «Что требуется получить в результате постановки задачи и ее реализации на персональном компьютере?», т.е. уточняет первоначально поставленную цель решения задачи.

Комплекс 3 «Описание входной информации» отвечает на вопрос, на основании какой информации может быть получена выходная информация. Под входной информацией понимается вся информация, необходимая для решения задачи и расположенная на различных носителях: первичных документах, машинных носителях, в памяти персонального компьютера. С этой целью составляются перечень входной информации и состав реквизитов каждого вида входной информации, расположение реквизитов входной информации, описание полей (реквизитов) входных документов.

При определении *перечня входной информации* описываются вид информации (текущая переменная, нормативно-справочная), источники информации, специфика сбора, хранения информации, способы поступления, а также объемно-временные характеристики и способы контроля.

Таблица 3.1 - Структура выходного документа

Наименование поля (реквизита)	Идентификация	Тип данных	Количество разрядов
1. Код группы товара	GRUP	Числовой	2
2. Код товара	TOV	Числовой	6
3. Количество товаров — продано, шт.	KPROD	Числовой	3
4. Цена покупки, руб.	PGEN	Числовой	3
5. Цена продажи, руб.	PPROD	Числовой	3
6. Объем реализации по закупочным ценам, руб.	VRP	Числовой	4
7. Объем реализации по ценам продажи, руб.	VRPP	Числовой	4
8. Наличие на складе — количество, шт.	KCKL	Числовой	3
9. Наличие на складе по ценам покупки, руб.	SCKL	Числовой	4
10. Прибыль или убыток, руб.	PRIB	Числовой	4

Состав реквизитов входной информации зависит от особенностей входной информации. Он должен быть необходимым и достаточным для организации дальнейшей обработки. *Расположение реквизитов* осуществляется в соответствии с существующими правилами ее проектирования. *Описание полей (реквизитов)* выполняется по отношению ко всем видам входной информации и осуществляется аналогично подобной операции для выходной информации (см. табл. 3.1).

В этом же комплексе обобщаются особенности входной информации, которые конкретизируют вид информации (текущая, нормативно-справочная), источники возникновения информации, специфику ее сбора, способы поступления, объемно-временные характеристики, особенности контроля данных.

Комплекс 4 «Алгоритмы решения задами» отвечает на вопрос: «Каким образом, т.е. на основе каких алгоритмов расчета входная информация преобразуется в выходную информацию?» Разработка алгоритмов решения задачи связана с выполнением неформализованного и формализованного моделирования.

При *неформализованном моделировании* алгоритмы расчетов представляются в описательном виде. Например, в данной задаче «Оперативный анализ прибыли и убытков по товарам в супермаркете» используются алгоритмы:

1) Умножение *Количества товаров - продано* на *Цену покупки* для получения *Объема реализации по ценам покупки.*

2) Умножение *Количества товаров — продано* на *Цену продажи* для получения *Объема реализации по ценам продажи.*

3) Умножение *Количества товаров на складе* на *Цену покупки* для получения *Наличия товаров на складе в стоимостном выражении.*

4) Вычитание из *Объема реализации по ценам продажи Объема реализации по ценам покупки* и *Наличия товаров на складе в стоимостном выражении для* получения *Прибыли (или Убытка)* по *Коду товара* с указанием *Кода группы товара.*

5) Суммирование *Прибыли и Убытков* по *Коду товара* внутри *Кода группы товара с* целью получения *Прибыли (или Убытка)* по *Коду группы товара.*

Результат взаимодействия показателей по изложенным алгоритмам желательно отразить в виде неформализованной модели, которая может быть представлена как схема взаимодействия различных показателей по их наименованиям или идентификаторам.

Формализованное моделирование осуществляется по определенным правилам. Согласно правилам по каждому экономическому показателю выявляются реквизиты-признаки и реквизиты-основания. Им присваиваются условные обозначения: реквизитам-основаниям заглавные буквы, реквизитам-признакам строчные буквы. Экономический показатель выражается в виде совокупности обозначений. Взаимосвязи показателей представляются в виде формул. Совокупность формул отражает инфологическую модель решения задачи.

Инфологическая модель задачи «Оперативный анализ прибыли и убытков по товарам в супермаркете» представлена на рис. 3.13.

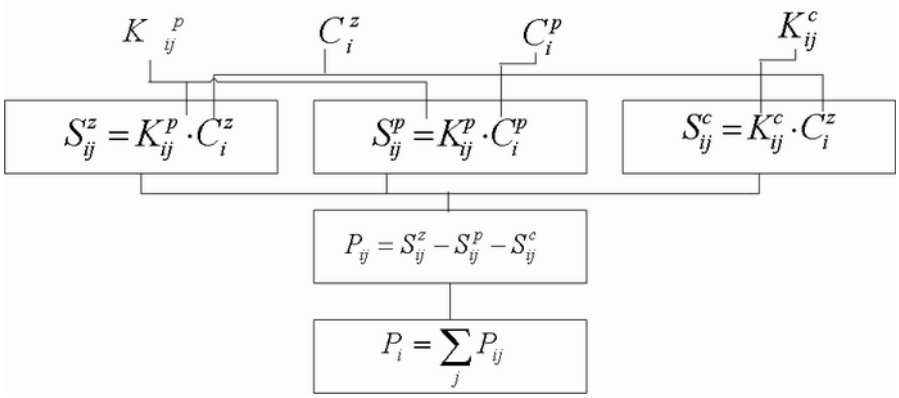


Рисунок 3.13 - Инфологическая модель задачи "Оперативный анализ прибыли и убытков по товарам в супермаркете"

Инфологическая модель не только позволяет четко выразить логику расчета, но и служит основой для реализации других видов моделей: матричной, функциональной зависимости, графосхем. Это позволяет проектировать базы данных по задачам, комплексам задач, функциональным подсистемам и системе в целом. Созданием инфологической модели заканчивается технология постановки задачи.

Технология постановки задачи находит продолжение в технологии ее реализации на персональном компьютере и полностью зависит от используемых программных и технических средств.

5.4. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ИНТЕРФЕЙСА ИНФОРМАЦИОННОЙ СИСТЕМЫ

Этапы проектирования форм электронных документов

Проектирование форм электронных документов, т.е. создание шаблона формы с помощью программного обеспечения проектирования форм, обычно включает в себя выполнение следующих шагов:

первый шаг - создание структуры ЭД, который заключается в рисовании линий, создании графических элементов (например, логотипов), т. е. подготовке внешнего вида с помощью графических средств проектирования;

второй шаг - определение содержания формы ЭД, т.е. выбор способов, которыми будут заполняться поля. Поля могут быть заполнены вручную или посредством выбора значений из какого-либо списка, меню, базы данных. В последнем случае дизайнер форм должен связать форму с базой данных.

Технологическая сеть процесса проектирования макетов экранных форм документов приведена на рис. 4.1.

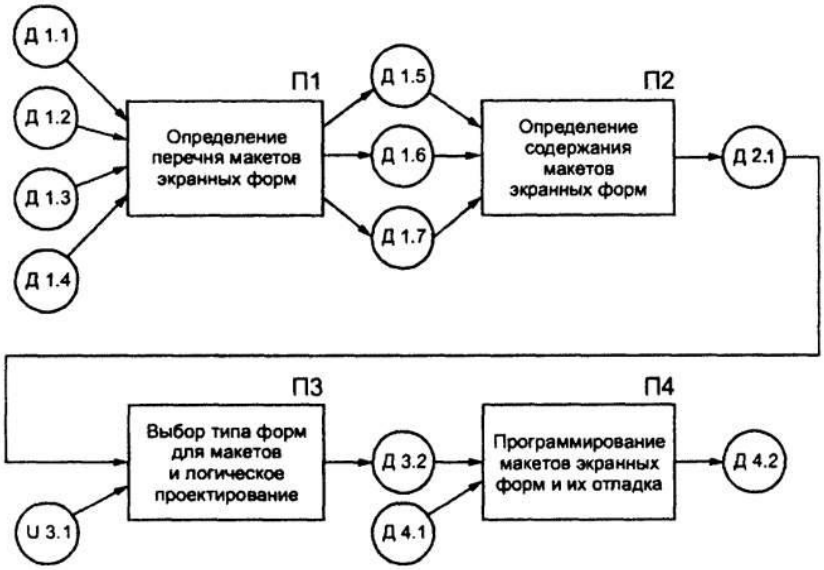


Рисунок 4.1 - Технологическая сеть процесса проектирования макетов экранных форм документов:

Д 1.1 - постановка задачи; Д 1.2 - документы с оперативной информацией; Д 1.3 - документы с постоянной информацией; Д 1.4 - документы с результатной информацией; Д 1.5 - перечни макетов с оперативной информацией; Д 1.6 - перечни макетов с постоянной информацией; Д 1.7 - перечни макетов с результатной информацией; Д 2.1 - содержание макетов - перечни полей; U 3.1 - универсум типов форм; Д 3.2 - логические структуры макетов; Д 4.1 - язык программирования; Д 4.2 - программы для ввода или вывода информации

Для определения перечня макетов экранных форм по каждой задаче проектировщик анализирует (операция П1) «постановку» каждой задачи (Д1.1), в которой приводятся перечни используемых входных документов с оперативной и постоянной информацией (Д1.2,1.3) и документов с результатной информацией (Д 1.4). В процессе анализа определяется, будут ли создаваться макеты под каждый документ или будет осуществляться интеграция полей нескольких входных документов в один макет. В результате получается перечень макетов экранных форм входных и результатных документов (Д1.5 -Д1.7).

Содержание макетов (операция П2) определяется на основе анализа состава реквизитов первичных документов с постоянной и оперативной информацией и результатных документов. Содержание макетов (Д2.1) - это перечни полей, значения которых должны находиться в файлах с оперативной и результатной информацией, и типы форматов этих полей.

При выполнении третьей операции ПЗ осуществляются выбор типа формы для каждого макета и проектирование их логической структуры (Д3.2). Под логическим проектированием макетов подразумеваются распределение полей по зонам выбранной формы документа и определение последовательности полей в каждой зоне. На входе операции используется универсум типов форм документов (U3.1).

При построении структур макетов для первичных документов с оперативной информацией используют комбинированную форму документа, максимально приближенную к той, которая была использована для построения самого документа. Расположение полей должно быть в последовательности, соответствующей логической структуре документа и файлов с оперативной информацией, сокращающей трудоемкость операции загрузки информации в информационную базу.

При построении макетов для документов с постоянной информацией следует иметь в виду, что эти макеты используются для ввода и актуализации записей информационной базы, поэтому для их проектирования применяют, как правило, анкетную форму расположения реквизитов, удобную для выполнения этих операций.

Макеты, предназначенные для вывода на экран результатной информации, строятся по методике проектирования результатных документов, т.е. на основе использования комбинированной формы с трехзонным расположением реквизитов и многострочной содержательной частью.

В основе выбора формы макета лежат принципы минимальной трудоемкости и стоимости ввода информации в ЭВМ, максимальной степени читабельности результатной информации, выводимой на экран, и максимальной надежности и достоверности выполнения этих операций.

Работа заканчивается выполнением операции «Программирование разработанных макетов экранных форм и их отладка» (операция П4) с использованием выбранного языка программирования (Д4.1) и апробацией их работы.

В процессе проектирования и программирования макетов проектировщик должен делить экранное поле на две части: информационную, предназначенную для собственно самого макета, и служебную для дополнительной информации.

Требования к форме электронных документов

Информационная часть должна отвечать следующим требованиям:

иметь хороший обзор;

не должна быть перегружена справочными реквизитами, значения которых следует выдавать на экран в виде списков для просмотра при наборе значений группировочных признаков;

значения группировочных признаков также следует выдавать на экран из справочников при переходе указателя в данное поле или при наборе неправильных значений этих признаков;

каждое поле должно быть снабжено подсказкой, которую следует выдавать на экран при неправильных действиях пользователя;

должна быть обеспечена возможность исправления ошибок в наборе;

продвижение указателя должно быть обеспечено в прямом и обратном направлении по вертикали и по горизонтали с возможностью экранной прокрутки всего документа;

текущее время и дата должны проставляться автоматически;

общий цвет информационной части должен быть спокойных тонов, не вызывающих усталости пользователя при многочасовой работе с ним;

цвет полей, подлежащих вводу с клавиатуры, должен отличаться от цвета информационной части;

цвет активного поля должен отличаться от основного цвета информационной части и от цвета этого поля в пассивном состоянии.

Служебная часть макета, как правило, помещается в нижней части экрана и должна быть отделена от информационной части графически и цветом. Она предназначается для включения подсказок об использовании тех кнопок, с помощью которых пользователь может работать с этим макетом:

производить откат на одно поле назад;

отказываться от ввода;

производить загрузку введенной записи в базу данных;

выдавать на печать и т.д.

Кроме того, каждый макет должен иметь в этой части экрана инструкционную часть для пользователя со справочной информацией о порядке заполнения макетов и всех видах ошибок, которые могут возникнуть при работе с ними, и способами их исправления.

Общие сведения об объектном моделировании программных приложений

Существует множество технологий и инструментальных средств, с помощью которых можно реализовать в некотором смысле оптимальный проект ИС, начиная с этапа анализа и заканчивая созданием программного кода системы. В большинстве случаев эти технологии предъявляют весьма жесткие требования к процессу разработки и используемым ресурсам, а попытки трансформировать их под конкретные проекты оказываются безуспешными. Эти технологии представлены CASE-средствами верхнего уровня или CASE-средствами полного жизненного цикла (upper CASE tools или full life-cycle CASE tools). Они не позволяют оптимизировать деятельность на уровне отдельных элементов проекта, и, как следствие, многие разработчики перешли на так называемые CASE-средства нижнего уровня (lower CASE tools). Однако они столкнулись с новой проблемой — проблемой организации взаимодействия между различными командами, реализующими проект.

Унифицированный язык объектно-ориентированного моделирования

Унифицированный язык объектно-ориентированного моделирования Unified Modeling Language (UML) явился средством достижения компромисса между этими подходами. Существует достаточное количество инструментальных средств, поддерживающих с помощью UML жизненный цикл информационных систем, и, одновременно, UML является достаточно гибким для настройки и поддержки специфики деятельности различных команд разработчиков.

Создание UML началось в октябре 1994 г., когда Джим Рамбо и Гради Буч из Rational Software Corporation стали работать над объединением своих методов OMT и Booch. В настоящее время консорциум пользователей UML Partners включает в себя представителей таких грандов информационных технологий, как Rational Software, Microsoft, IBM, Hewlett-Packard, Oracle, DEC, Unisys, IntelliCorp, Platinum Technology.

UML представляет собой объектно-ориентированный язык моделирования, обладающий следующими основными характеристиками:

- является языком визуального моделирования, который обеспечивает разработку репрезентативных моделей для организации взаимодействия заказчика и разработчика ИС, различных групп разработчиков ИС;

- содержит механизмы расширения и специализации базовых концепций языка.

UML — это стандартная нотация визуального моделирования программных систем, принятая консорциумом Object Managing Group (OMG) осенью 1997 г., и на сегодняшний день она поддерживается многими объектно-ориентированными CASE-продуктами.

UML включает внутренний набор средств моделирования, которые сейчас приняты во многих методах и средствах моделирования. Эти концепции необходимы в большинстве прикладных задач, хотя не каждая концепция необходима в каждой части каждого приложения. Пользователям языка предоставлены возможности:

- строить модели на основе средств ядра, без использования механизмов расширения для большинства типовых приложений;

- добавлять при необходимости новые элементы и условные обозначения, если они не входят в ядро, или специализировать компоненты, систему условных обозначений (нотацию) и ограничения для конкретных предметных областей.

Язык UML

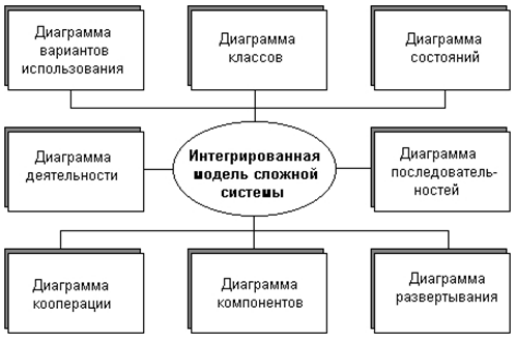


Рисунок 4.1 - Интегрированная модель сложной системы в нотации языка UML

Стандарт UML предлагает следующий набор диаграмм для моделирования:

1. диаграммы вариантов использования (use case diagrams) – для моделирования бизнес-процессов организации и требований к создаваемой системе);

2 диаграммы классов (class diagrams) – для моделирования статической структуры классов системы и связей между ними;

3. диаграммы поведения системы (behavior diagrams):

3.1 диаграммы взаимодействия (interaction diagrams):

3.1.1 диаграммы последовательности (sequence diagrams) и

3.1.2 кооперативные диаграммы (collaboration diagrams) – для моделирования процесса обмена сообщениями между объектами;

3.2 диаграммы состояний (statechart diagrams) – для моделирования поведения объектов системы при переходе из одного состояния в другое;

3.3 диаграммы деятельностей (activity diagrams) – для моделирования поведения системы в рамках различных вариантов использования, или моделирования деятельностей;

4. диаграммы реализации (implementation diagrams):

4.1 диаграммы компонентов (component diagrams) – для моделирования иерархии компонентов (подсистем) системы;

4.2 диаграммы развертывания (deployment diagrams) – для моделирования физической архитектуры системы.

Диаграммы вариантов использования

Понятие варианта использования (use case) впервые ввел Ивар Якобсон и придал ему такую значимость, что в настоящее время вариант использования превратился в основной элемент разработки и планирования проекта.

Вариант использования представляет собой последовательность действий (транзакций), выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом). Вариант использования описывает типичное взаимодействие между пользователем и системой. В простейшем случае вариант использования определяется в процессе обсуждения с пользователем тех функций, которые он хотел бы реализовать. На языке UML вариант использования изображают следующим образом:

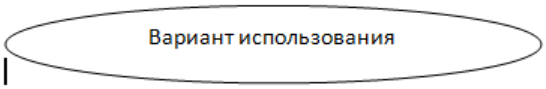


Рисунок 4.2 - Вариант использования

Действующее лицо (actor) – это роль, которую пользователь играет по отношению к системе. Действующие лица представляют собой роли, а не конкретных людей или наименования работ. Несмотря на то, что на диаграммах вариантов использования они изображаются в виде стилизованных человеческих фигурок, действующее лицо может также быть внешней системой, которой необходима некоторая информация от данной системы. Показывать на диаграмме действующих лиц следует только в том случае, когда им действительно необходимы некоторые варианты использования. На языке UML действующие лица представляют в виде фигур:

Рисунок 4.3 - Действующее лицо (актер)

Действующие лица делятся на три основных типа:
- пользователи;
- системы;
- другие системы, взаимодействующие с данной;
- время.

Время становится действующим лицом, если от него зависит запуск каких-либо событий в системе.

Связи между вариантами использования и действующими лицами

В языке UML на диаграммах вариантов использования поддерживается несколько типов связей между элементами диаграммы. Это связи коммуникации (communication), включения (include), расширения (extend) и обобщения (generalization).

Связь коммуникации – это связь между вариантом использования и действующим лицом. На языке UML связи коммуникации показывают с помощью однонаправленной ассоциации (сплошной линии).

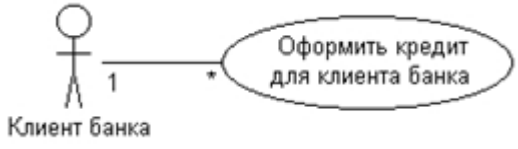


Рисунок 4.4 - Пример связи коммуникации

Связь включения применяется в тех ситуациях, когда имеется какой-либо фрагмент поведения системы, который повторяется более чем в одном варианте использования. С помощью таких связей обычно моделируют многократно используемую функциональность.

Связь расширения применяется при описании изменений в нормальном поведении системы. Она позволяет варианту использования только при необходимости использовать функциональные возможности другого.

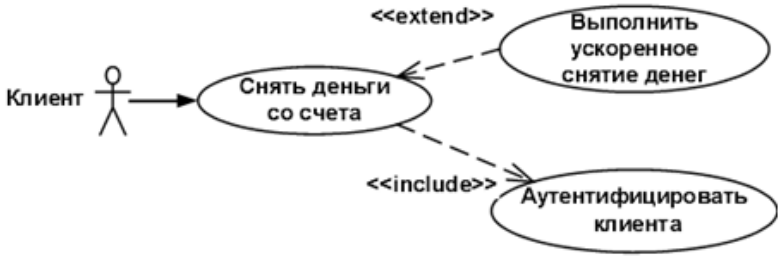


Рисунок 4.5 - Пример связи включения и расширения

С помощью связи обобщения показывают, что у нескольких действующих лиц имеются общие черты.

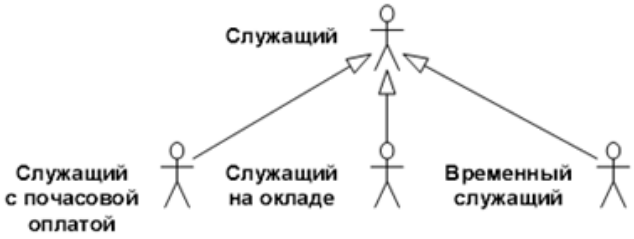


Рисунок 4.6 - Пример связи обобщения

Диаграммы взаимодействия (interaction diagrams)

Диаграммы взаимодействия (interaction diagrams) описывают поведение взаимодействующих групп объектов. Как правило, диаграмма взаимодействия охватывает поведение объектов в рамках только одного варианта использования. На такой диаграмме отображается ряд объектов и те сообщения, которыми они обмениваются между собой.

Сообщение (message) – это средство, с помощью которого объект-отправитель запрашивает у объекта получателя выполнение одной из его операций.

Информационное (informative) сообщение – это сообщение, снабжающее объект-получатель некоторой информацией для обновления его состояния.

Сообщение-запрос (interrogative) – это сообщение, запрашивающее выдачу некоторой информации об объекте-получателе.

Императивное (imperative) сообщение – это сообщение, запрашивающее у объекта-получателя выполнение некоторых действий.

Существует два вида диаграмм взаимодействия: диаграммы последовательности (sequence diagrams) и кооперативные диаграммы (collaboration diagrams).

Диаграмма последовательности (sequence diagrams)

Диаграмма последовательности отражает поток событий, происходящих в рамках варианта использования.

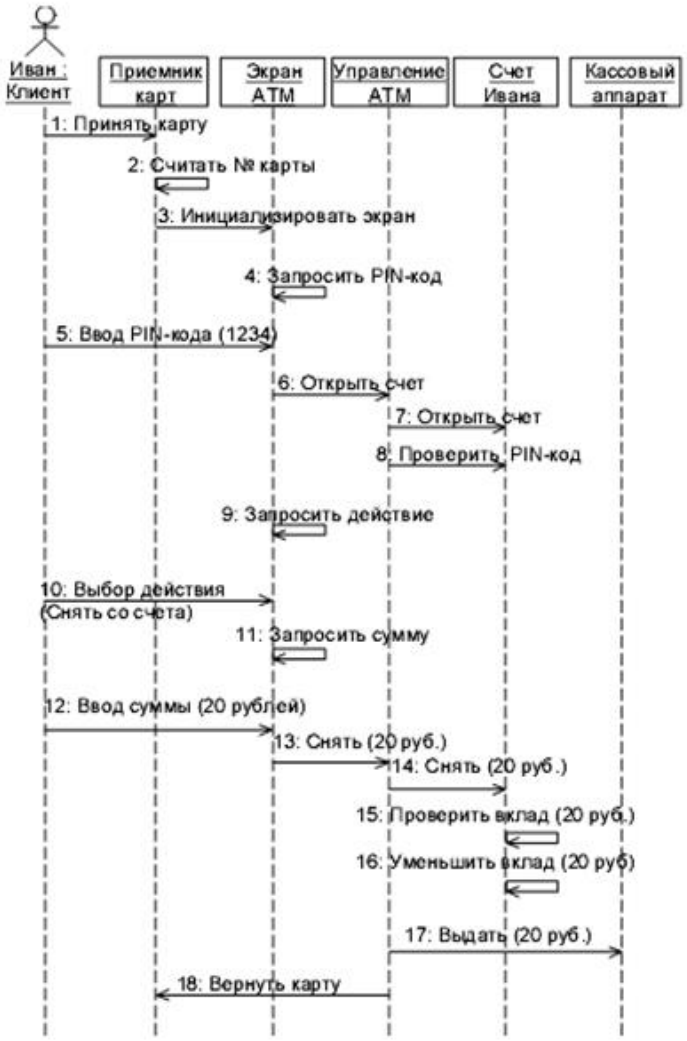


Рисунок 4.7 - Пример диаграммы последовательности

Все действующие лица показаны в верхней части диаграммы. Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектом или между объектами для выполнения требуемых функций.

На диаграмме последовательности объект изображается в виде прямоугольника, от которого вниз проведена пунктирная вертикальная линия. Эта линия называется линией жизни (lifeline) объекта. Она представляет собой фрагмент жизненного цикла объекта в процессе взаимодействия.

Каждое сообщение представляется в виде стрелки между линиями жизни двух объектов. Сообщения появляются в том порядке, как они показаны на странице сверху вниз. Каждое сообщение помечается как минимум именем сообщения. При желании можно добавить также аргументы и некоторую управляющую информацию. Можно показать самоделегирование (self-delegation) – сообщение, которое объект посылает самому себе, при этом стрелка сообщения указывает на ту же самую линию жизни.

Диаграмма кооперации (collaboration diagram)

Диаграммы кооперации отображают поток событий через конкретный сценарий варианта использования, упорядочены по времени, а кооперативные диаграммы больше внимания заостряют на связях между объектами.

На диаграмме кооперации представлена вся та информация, которая есть и на диаграмме последовательности, но кооперативная диаграмма по-другому описывает

поток событий. Из нее легче понять связи между объектами, однако, труднее уяснить последовательность событий.

На кооперативной диаграмме так же, как и на диаграмме последовательности, стрелки обозначают сообщения, обмен которыми осуществляется в рамках данного варианта использования. Их временная последовательность указывается путем нумерации сообщений.

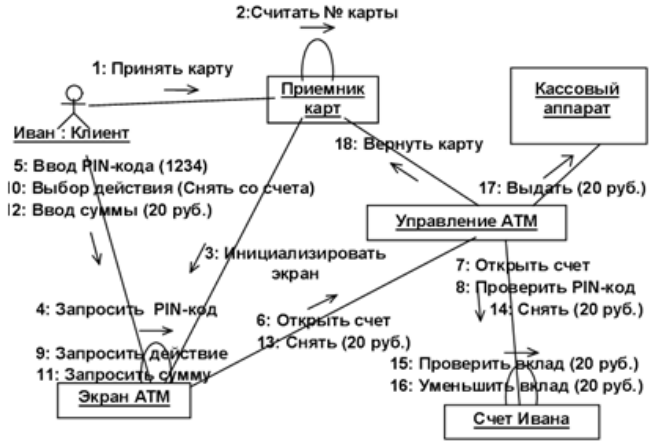


Рисунок 4.8 - Пример диаграммы кооперации

Диаграммы классов. Общие сведения

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

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

На диаграмме классов изображаются следующие элементы:

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

Класс

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

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

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

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

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

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

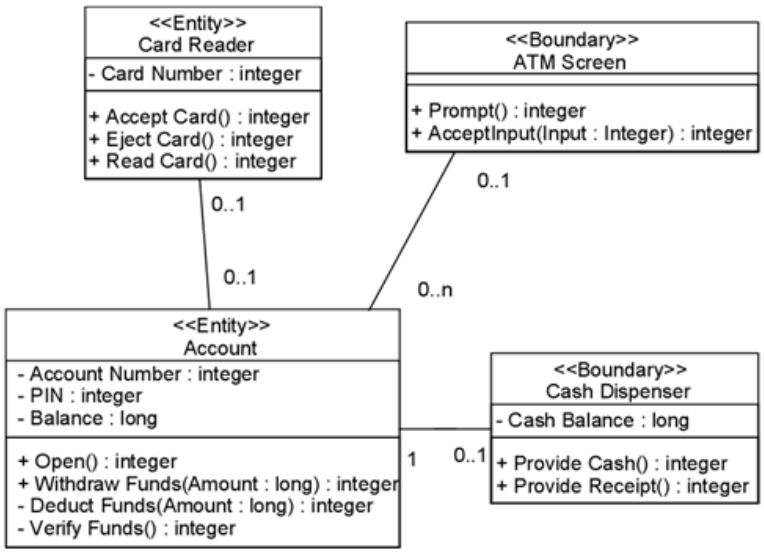


Рисунок 4.9 - Пример диаграммы классов

Стереотипы классов

Стереотипы классов – это механизм, позволяющий разделять классы на категории.

В языке UML определены три основных стереотипа классов:

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

Граничные классы

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

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

Классы-сущности

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

Управляющие классы

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

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

Помимо упомянутых выше стереотипов можно создавать и свои собственные.

Атрибуты

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

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

У атрибута можно определить четыре возможных значения этого параметра:

- Public (общий, открытый). Это значение видимости предполагает, что атрибут будет виден всеми остальными классами. Любой класс может просмотреть или изменить значение атрибута. В соответствии с нотацией UML общему атрибуту предшествует знак « + ».
- Private (закрытый, секретный). Соответствующий атрибут не виден никаким другим классом. Закрытый атрибут обозначается знаком « – » в соответствии с нотацией UML.
- Protected (защищенный). Такой атрибут доступен только самому классу и его потомкам. Нотация UML для защищенного атрибута – это знак « # ».
- Package or Implementation (пакетный). Предполагает, что данный атрибут является общим, но только в пределах его пакета. Этот тип видимости не обозначается никаким специальным значком.

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

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

Операции

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

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

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

В языке UML операции имеют следующую нотацию:

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

Следует рассмотреть четыре различных типа операций:

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

Операции реализации

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

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

Операции управления

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

Операции доступа

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

Вспомогательные операции

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

Чтобы идентифицировать операции, выполните следующие действия:

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

Рассмотрите управляющие операции. Может потребоваться добавить конструкторы и деструкторы.

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

Связи

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

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

Ассоциации

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

Рисунок 4.10 - Связь ассоциация

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

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

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

Зависимости

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

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

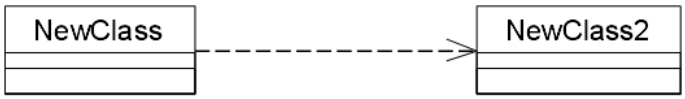


Рисунок 4.11 - Связь зависимость

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

Агрегации

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

Рисунок 4.12 - Связь агрегация

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

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

Обобщения (Наследование)

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

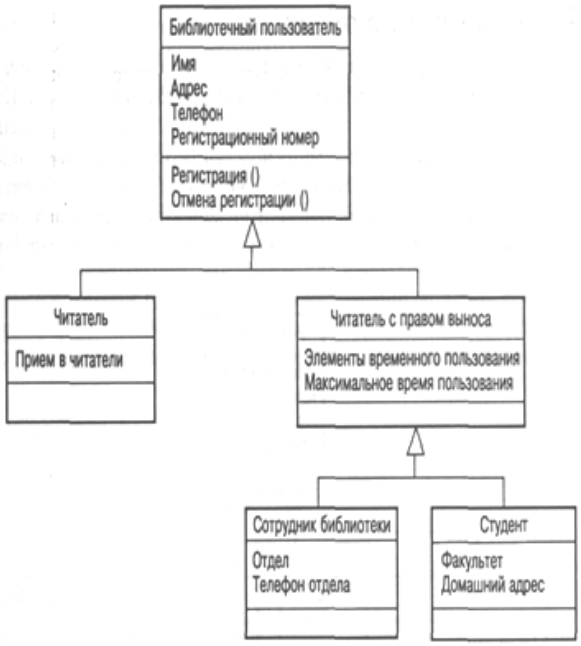


Рисунок 4.13 - Пример связи наследование

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

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

Множественность

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

Например, при разработке системы регистрации курсов в университете можно определить классы Course (курс) и Student (студент). Между ними установлена связь: у курсов могут быть студенты, а у студентов – курсы. Вопросы, на который должен

ответить параметр множественности: «Сколько курсов студент может посещать в данный момент? Сколько студентов может за раз посещать один курс?»

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

В языке UML приняты определенные нотации для обозначения множественности.

Таблица 4.1 - Обозначения множественности связей в UML

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

Имена связей

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

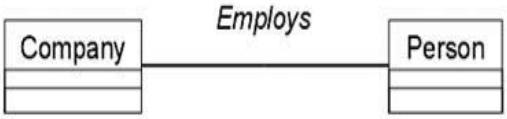


Рисунок 4.14 - Пример имен связей

Роли

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

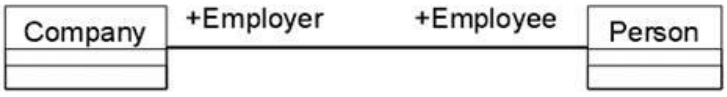


Рисунок 4.15 - Пример ролей связей

Пакет. Механизм пакетов

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

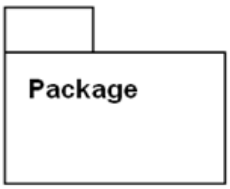


Рисунок 4.16 - Обозначение пакета в UML

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

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

Существует несколько наиболее распространенных подходов к группировке.

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

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

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

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

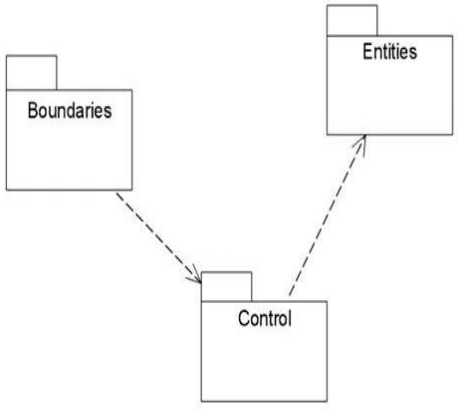


Рисунок 4.17 - Пример диаграммы пакетов

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

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

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

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

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

Диаграммы состояний

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

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

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

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

Деятельность

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

Входное действие

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

Выходное действие

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

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

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

Соответствующая строка на диаграмме выглядит как

Do: ^Цель.Событие (Аргументы)

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

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

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

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

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

События

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

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

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

Ограждающие условия

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

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

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

Действие

Действием (action), как уже говорилось, является непрерываемое поведение, осуществляющееся как часть перехода. Входные и выходные действия показывают внутри состояний, поскольку они определяют, что происходит, когда объект входит или выходит из него. Большую часть действий, однако, изображают вдоль линии перехода, так как они не должны осуществляться при входе или выходе из состояния.

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

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

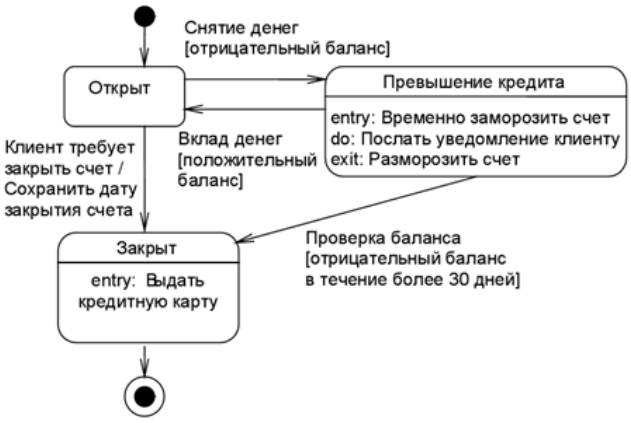


Рисунок 4.18 - Пример диаграммы состояний

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

Диаграммы размещения

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

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

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

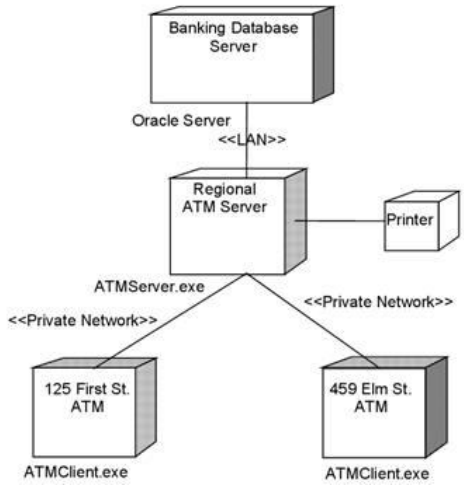


Рисунок 4.19 - Пример диаграммы размещения

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

Диаграммы компонентов

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

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

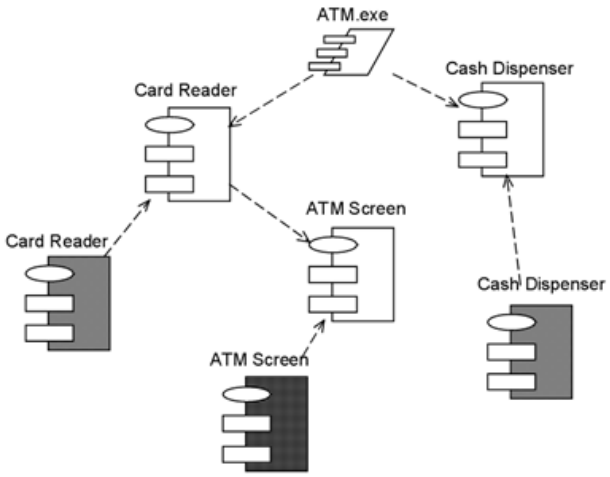


Рисунок 4.20 -. Пример диаграммы компонентов

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

Объединение диаграмм компонентов и развертывания

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

6. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

6.1. Перечень основной и дополнительной литературы, необходимой для освоения дисциплины

6.1.1. Перечень основной литературы

1. Заботина, Н. Н. Проектирование информационных систем: учеб. пособие/ Н.Н. Заботина. – М.: ИНФРА-М, 2017. – 329 с.

2. Коваленко, В. В. Проектирование информационных систем: учеб. пособие для вузов/ В.В. Коваленко. – М.: Форум, 2016. – 319 с.

6.1.2. Перечень дополнительной литературы

1. Емельянова, Н. З. Проектирование информационных систем: учебное пособие/ Н. З. Емельянова, Т. Л. Партыка, И. И. Попов. – М.: ФОРУМ, 2017. – 432 с.

2.Гвоздева, Т. В. Проектирование информационных систем: учеб. пособие/ Т. В. Гвоздева, Б. А. Баллод. – Ростов-на-Дону: Феникс, 2016. – 508 с.

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

1. Методические указания по выполнению лабораторных работ по дисциплине «Проектирование информационных систем».
2. Методические рекомендации для студентов по организации самостоятельной работы по дисциплине «Проектирование информационных систем».
3. Методические указания по выполнению курсового проекта по дисциплине «Проектирование информационных систем».

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

1. Национальный Открытый Университет. Интуит. http://www.intuit.ru.
2. Федеральный портал «Российское образование. http://www.edu.ru.
3. Российская государственная библиотека. http://www.rsl.ru.